Distributed Certificate Authority Reference Material

Relevant Topics and Technologies

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Mon, 30 Jun 2014 00:40:10 UTC

Contents

Articles

Bitcoin 1
Bitcoin network 17
Certificate authority 25
Chord (peer-to-peer) 30
Complex network 34
Computer network 39
Cryptographic hash function 57
Cryptographic nonce 65
Cryptography 66
Decentralization 80
Digital signature 100
Distributed hash table 107
Graph (mathematics) 112
Hash table 120
Hypercube 132
Hypercube graph 138
Kademlia 141
Merkle tree 148
Peer-to-peer 150
Public key certificate 161
Public-key cryptography 166
Route poisoning 178
Sybil attack 179
Topological space 181
Web of trust 186
X.509 190

References

Article Sources and Contributors 200
Image Sources, Licenses and Contributors 205

Article Licenses

License 208

Bitcoin

Bitcoin	
A common logo from the bitcoin reference client	
Date of introduction	3 January 2009
User(s)	Worldwide
Production	25 bitcoins per block (approximately every ten minutes) until mid 2016, and then afterwards 12.5 bitcoins per block for 4 years until next halving. This halving continues until 2110-2140.
Source	Number of bitcoins in circulation [1]
Method	Increase in the supply
Subunit	
10^{-8}	satoshi
Symbol	BTC, XBT, ฿

Bitcoin is a payment system introduced as open-source software in 2009 by developer Satoshi Nakamoto. The payments in the system are recorded in a public ledger using its own unit of account, which is also called **bitcoin**.[2] The WSJ and The Chronicle of Higher Education advocate use of lowercase *bitcoin* in all cases, however.[3] This article follows the latter convention.</ref> Payments work peer-to-peer without a central repository or single administrator, which has led the US Treasury to call bitcoin a decentralized virtual currency. Although its status as a currency is disputed, media reports often refer to bitcoin as a cryptocurrency or digital currency.

Bitcoins are created as a reward for payment processing work in which users offer their computing power to verify and record payments into the public ledger. Called mining, individuals or companies engage in this activity in exchange for transaction fees and newly created bitcoins. Besides mining, bitcoins can be obtained in exchange for fiat money, products, and services. Users can send and receive bitcoins electronically for an optional transaction fee using wallet software on a personal computer, mobile device, or a web application.

Bitcoin as a form of payment for products and services has seen growth, and merchants have an incentive to accept the digital currency because fees are lower than the 2–3% typically imposed by credit card processors. The European Banking Authority has warned that bitcoin lacks consumer protections. Unlike credit cards, any fees are paid by the purchaser not the vendor. Bitcoins can be stolen and chargebacks are impossible.[4] Commercial use of bitcoin is currently small compared to its use by speculators, which has fueled price volatility.

Bitcoin has been a subject of scrutiny amid concerns that it can be used for illegal activities. In October 2013 the US FBI shut down the Silk Road online black market and seized 144,000 bitcoins worth US$28.5 million at the time. The US is considered bitcoin-friendly compared to other governments. In China, buying bitcoins with yuan is subject to restrictions, and bitcoin exchanges are not allowed to hold bank accounts.

Overview

The most important part of the bitcoin system is a public ledger that records financial transactions in bitcoins. Recording transactions is accomplished without the intermediation of any single, central authority. Instead, multiple intermediaries exist in the form of computer servers running bitcoin software. By connecting over the Internet, these servers form a network that anyone can join. Transactions of the form *payer X wants to send Y bitcoins to payee Z* are broadcast to this network using readily available software applications. Bitcoin servers can validate these transactions, add them to their copy of the ledger, and then broadcast these ledger additions to other servers.

The block chain ledger

All bitcoin transfers are recorded in a computer file that acts as a public ledger called the block chain, which everyone can examine. Where a conventional ledger records the transfer of actual bills or promissory notes that exist apart from it, bitcoins are simply entries in the block chain and do not exist outside of it.

Mining

Obsolete bitcoin mining hardware common in early and mid-2013. Called USB Erupter, each can calculate 333 megahashes per second.

Maintaining the block chain is called mining, and those who do are rewarded with newly created bitcoins and transaction fees. Miners may be located anywhere in the world; they process payments by verifying each transaction as valid and adding it to the block chain. As of 2014 payment processing is rewarded with 25 newly created bitcoins per block added to the block chain. To claim the reward, a special transaction called a coinbase is included with the processed payments. All bitcoins in circulation can be traced back to such coinbase transactions. The bitcoin protocol specifies that the reward for adding a block will be halved approximately every four years. Eventually, the reward will be removed entirely when an arbitrary limit of 21 million bitcoins is reached c. 2140, and transaction processing will then be rewarded by transaction fees solely. Paying a transaction fee is optional, but may speed up confirmation of the transaction. Payers have an incentive to include such fees because doing so means their transaction will likely be added to the block chain sooner; miners can choose which transactions to process and prefer to include those that pay fees.

As of 2013 mining has become quite competitive, and the process has been compared to an arms race as ever more specialized technology is utilized. The most efficient mining hardware makes use of custom designed application-specific integrated circuits, which outperform general purpose CPUs and use less power as well. Without access to these purpose built machines, a bitcoin miner is unlikely to earn enough to even cover the cost of the electricity used in his or her efforts.

Mining pools

The individual odds of winning the reward for adding a block to the block chain decrease alongside an increase in the number of miners. Many miners participate, but the reward for each block can only go to a single bitcoin address. As of 2014 it has become common for miners to join organized mining pools, which split the work and the reward among all participants and make mining a less risky endeavor. Even for those who join pools, the cost of the electricity necessary to mine may outweigh the bitcoin rewards from doing so.

Anonymity

The public nature of bitcoin means that, while those who use it are not identified by name, transactions can be linked to individuals and companies. All transactions are recorded into a public ledger, the block chain, and are viewable by everyone. Additionally, many jurisdictions require exchanges, where people can buy and sell bitcoins for cash, to collect personal information. The privacy concerns of some who use bitcoins are sufficient to cause them to take additional steps to cover their tracks. In order to obfuscate the link between individual and transaction, a different bitcoin address for each transaction can be used, and others rely on so-called mixing services that allow users to trade bitcoins whose transaction history implicates them for coins with different transaction histories.[5]

It has further been suggested that bitcoin payments should not be considered more anonymous than credit card payments.

Buying and selling

Bitcoins can be bought and sold with many different currencies from individuals and companies. Bitcoins may be purchased in person or at a bitcoin ATM in exchange for cash currency.[6] Participants in online exchanges offer bitcoin buy and sell bids. Using an online exchange to obtain bitcoins entails some risk, and according to one study, 45% of exchanges fail and take client bitcoins with them. Since bitcoin transactions are irreversible, sellers of bitcoins must take extra measures to ensure they have received traditional funds from the buyer.

Wallets

See also: Digital wallet and Online wallet

Example of *Casascius physical bitcoins*

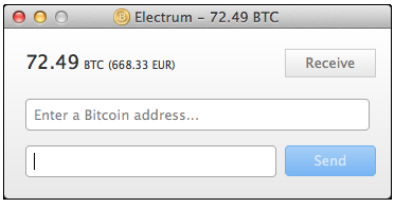
A paper wallet with QR codes

While wallets are often described as being a place to hold or store bitcoins,[7] due to the nature of the system, bitcoins are inseparable from the block chain transaction ledger. Perhaps a better way to define a wallet is something "that stores the digital credentials for your bitcoin holdings" and allows you to access (and spend) them. Bitcoin uses public-key cryptography, in which two cryptographic keys, one public and one private, are generated. The public key can be thought of as an account number or name and the private key, ownership credentials. At its most basic, a wallet is a collection of these keys. Most bitcoin software also includes the ability to make transactions, however.

Perhaps better termed physical wallets, physical bitcoins are ubiquitous in media coverage and combine a novelty coin with a private key printed on paper, metal, wood, or plastic. Physical bitcoins aren't widely seen outside of pictures in news articleWikipedia:Citation needed, but for those serious about security, storing private keys on paper printouts or in offline data storage devices is the best option.

Software

Main article: Bitcoin network



Electrum – sample bitcoin client

Bitcoin client software called a bitcoin wallet allows a user to transact bitcoins. A wallet program generates and stores private keys, and communicates with peers on the bitcoin network. The first wallet program called Bitcoin-Qt was released in 2009 by Satoshi Nakamoto as open source code. It can be used as a desktop wallet for payments or as a server utility for merchants and other payment services. Bitcoin-Qt, also called *Satoshi client* is sometimes referred to as the reference client because it serves to define the bitcoin protocol and acts as a standard for other implementations. As of version 0.9, Bitcoin-Qt has been renamed Bitcoin Core to more accurately describe its role in the network.[8] When making a purchase with a mobile device, QR codes are used ubiquitously to simplify transactions. Several server software implementations of the bitcoin protocol exist. So-called *full nodes* on the network validate transactions and blocks they receive, and relay them to connected peers.

Ownership

The ownership of bitcoins associated with a certain bitcoin address can be demonstrated with knowledge of the private key belonging to the address. For the owner, it is important to protect the private key from loss or theft. If a private key is lost, the user cannot prove ownership by other means. The coins are then lost and cannot be recovered. Because anyone with knowledge of the private key can take ownership of any associated bitcoins, theft can occur when a private key is revealed or stolen.

Security, theft, and loss

Integral to bitcoin security is the prevention of unauthorized transactions from an individual's wallet. A bitcoin transaction permanently transfers ownership to a new address, a string having the form of random letters and numbers derived from public keys by application of a hash function and encoding scheme. The corresponding private keys act as a safeguard for the owner; a valid payment message from an address must contain the associated public key and a digital signature proving possession of the associated private key. Because anyone with a private key can spend all of the bitcoins associated with the corresponding address, protection of private keys is quite important. Loss of a private key may result in theft, which has occurred on numerous occasions. The practical day-to-day security of bitcoin wallets is an ongoing concern. Risk of theft can be reduced by generating keys offline on an uncompromised computer and saving them on external storage or paper printouts.

Bitcoins can be lost. In 2013 one user said he lost 7,500 bitcoins, worth $7.5m at the time, when he discarded a hard drive containing his private key. Bitcoins can also be found. In March 2014, former bitcoin exchange Mt. Gox reported it found an "old wallet, which was used before June 2011 [that] held about 200,000 bitcoins".

History

Main article: History of Bitcoin

Bitcoin was first mentioned in a 2008 research paper published under the name Satoshi Nakamoto. It is unknown who Satoshi Nakamoto is. In 2009, an exploit in an early bitcoin client was found that allowed large numbers of bitcoins to be created.

In March 2013, a technical glitch caused a fork in the block chain, with one half of the network adding blocks to one version of the chain and the other half adding to another. For six hours two bitcoin networks operated at the same time, each with its own version of the transaction history. The core developers called for a temporary halt to transactions, sparking a sharp sell-off. Normal operation was restored when the majority of the network downgraded to version 0.7 of the bitcoin software.

Some mainstream websites began accepting bitcoins c. 2013. WordPress started in November 2012 followed by OKCupid in April 2013, Atomic Mall in November 2013, TigerDirect and Overstock.com in January 2014 and Expedia in June 2014. Certain non-profit or advocacy groups such as the Electronic Frontier Foundation allow bitcoin donations. (Although this organization stopped accepting bitcoins in 2011 and began again in 2013.[9])

The first law enforcement events occurred in May 2013. Assets belonging to the Mt. Gox exchange were seized by Department of Homeland Security, and the Silk Road drug market website was shut down by the FBI.

In October 2013, Chinese internet giant Baidu had allowed clients of website security services to pay with bitcoins. During November 2013, the China-based bitcoin exchange BTC China overtook the Japan-based Mt. Gox and the Europe-based Bitstamp to become the largest bitcoin trading exchange by trade volume. On 19 November 2013, the value of a bitcoin on the Mt. Gox exchange soared to a peak of US$900 after a United States Senate committee hearing was told that virtual currencies were a legitimate financial service. On the same day, one bitcoin traded for over RMB¥6780 (US$1100) in China. On 5 December 2013, the People's Bank of China prohibited Chinese financial institutions from using bitcoins. After the announcement, the value of bitcoins dropped and Baidu no longer accepted bitcoins for certain services. Buying real-world goods with any virtual currency had been illegal in China since at least 2009.

The first bitcoin ATM was installed in October 2013 in Vancouver, British Columbia, Canada.

With roughly 12 million existing bitcoins as of November 2013, the new price increased the market cap for bitcoin to at least US$7.2 billion. By 23 November 2013, the total market capitalization of bitcoin exceeded US$10 billion for the first time.

In the US two men were arrested in January 2014 on charges of money-laundering using bitcoins including Charlie Shrem, the head of defunct bitcoin exchange BitInstant and a vice chairman of the Bitcoin Foundation. Shrem allegedly allowed the other arrested party to purchase large quantities of bitcoins for use on black-market websites.

In early February 2014, one of the largest bitcoin exchanges, Mt. Gox, suspended withdrawals citing technical issues.[10] By the end of the month, Mt. Gox had filed for bankruptcy protection in Japan amid reports that 744,000 bitcoins had been stolen. Originally a site for trading Magic: The Gathering cards, Mt. Gox once was the dominant bitcoin exchange although prior to the collapse its popularity had waned.[11]

Economics

Classification as money

Bitcoin is often referred to as a currency, but it does not conform to widely used definitions of money. Economists generally agree that to qualify as money, something must be a store of value, a medium of exchange, and a unit of account. Bitcoin has some way to go if it wants to meet these criteria. It does best as a medium of exchange. (About 1,000 bricks and mortar businesses were willing to accept payment in bitcoins as of November 2013 in addition to more than 35,000 online merchants.[12]) The bitcoin market currently suffers from volatility, limiting the ability of bitcoins to act as a stable store of value, and, although bitcoins are the unit of account for the block chain, bitcoin does not see use as a unit of account outside of it. Where people are allowed to buy with bitcoins, prices are not denominated in bitcoins. As there are some exceptions to the above, it could be argued that bitcoin is a store of value (although a very poor one) and that it is a unit of account (although a very rarely used one).Wikipedia:Citation needed

Others feel being a widely accepted or common medium of exchange is sufficient for something to be considered money.[13] Still more believe money must be both a medium of exchange and a unit of account. Others think money has four characteristics. It must be: a medium of exchange, a unit of account, a store of value, and a standard of deferred payments.

The People's Bank of China has stated that bitcoin "is fundamentally not a currency". According to the director of the Institute for Money, Technology and Financial Inclusion at the University of California-Irvine there is "an unsettled debate about whether bitcoin is a currency or payment protocol". Despite this, in the media bitcoin is commonly referred to with terms that include the word currency such as digital currency, digital cash, virtual currency, electronic money, or cryptocurrency. Some media outlets do make a distinction between "real" money and bitcoins, however.

Price and volatility

To improve access to price information and increase transparency, on 30 April 2014 Bloomberg LP announced plans to list prices from bitcoin companies Kraken and Coinbase on its 320.000 subscription financial data terminals.

According to Mark T. Williams of Boston University, the volatility of bitcoin is over seven times that of gold and over eight times that of the S&P 500. The Bitcoin Foundation contends that high volatility is due to insufficient liquidity while a Forbes journalist claims that it is related to the uncertainty of its long-term value. As of 2014, pro-bitcoin venture capitalists argue the greatly increased trading volume that planned high-frequency trading exchanges are hoped to bring will decrease price volatility. Volatility has little effect on the utility of bitcoin as a payment processing system.

The price of bitcoins has gone through various cycles of appreciation and depreciation referred to by some as bubbles and busts. In 2011, the value of one bitcoin rapidly rose from about US$0.30 to US$32 before returning to US$2. In the latter half of 2012 and during the 2012-2013 Cypriot Financial Crisis, the bitcoin price began to rise, reaching a peak of US$266 on 10 April 2013, before crashing to around US$50. At the end of 2013, the cost of one bitcoin rose to the all-round peak of US$1135, but fell to the price of US$693 three days later. In 2014 the price fell sharply, and as of April remained depressed at little more than half that of 2013.

Alternative to national currencies

Bitcoins are used by some Argentinians as an alternative to the official currency,[14] which is stymied by inflation and strict capital controls, to protect their savings against inflation or the possibility that governments could confiscate savings accounts. It's been suggested that during the 2012–2013 Cypriot financial crisis bitcoin purchases rose due to fears that savings accounts would be confiscated or taxed.

Bitcoins are accepted in this café in the Netherlands as of 2013

Speculative bubble

Bitcoin has been labelled a speculative bubble by many including Former Federal Reserve Chairman Alan Greenspan and economist John Quiggin. Two lead software developers of bitcoin, Gavin Andresen and Mike Hearn, have warned that bubbles may occur.[15] Nobel Laureate Robert Shiller said that bitcoin "exhibited many of the characteristics of a speculative bubble." Others reject the label and see bitcoin's quick rise in price as nothing more than normal economic forces at work.

As investment

One way of investing in bitcoins is to buy and hold them as a long-term, high-risk investment. FINRA, a United States self-regulatory organization, warns that investing in bitcoins carries significant risks. The European Banking Authority warns that the risks of investment go beyond a potential fall in the value of bitcoins. Bitcoins may be of limited value to unsophisticated investors. Risk hasn't deterred some such as the Winklevoss twins, who made a US$1.5 million personal investment and attempted to launch a bitcoin ETF. Other investors, like Peter Thiel's Founders Fund, which invested US$3 million, don't purchase bitcoins themselves instead funding bitcoin infrastructure like companies that provide payment systems to merchants, exchanges, and wallet services, etc. Investors also invest in bitcoin mining.

Supply

Growth of the bitcoin supply is predefined by the bitcoin protocol. Currently there are over twelve million bitcoins in circulation with an approximate creation rate of 25 every ten minutes. The total supply is capped at an arbitrary limit of 21 million, and every four years the creation rate is halved. This means new bitcoins will continue to be released for more than a hundred years.

Value forecasts

Financial journalists and analysts, economists, and investors have attempted to predict the possible future value of bitcoin. Economist John Quiggin stated, "bitcoins will attain their true value of zero sooner or later, but it is impossible to say when." In 2013, Bank of America FX and Rate Strategist David Woo forecast a maximum fair value per bitcoin of $1,300. Bitcoin investor Cameron Winklevoss stated in 2013 that the "[s]mall bull case scenario for bitcoin is... 40,000 USD a coin". In late 2013, finance professor Mark Williams forecast a bitcoin would be worth less than ten US dollars by July 2014.

On its political economy

It has been claimed that Bitcoin is a currency that reflects a new type of capitalism, named “distributed” capitalism . This new iteration of capitalism conforms to the characteristics of the network era and utilizes the peer-to-peer infrastructures to achieve capital accumulation . Bitcoin is designed to allow multiple users, though in a competitive framework. It might appear as though it exists outside the financial system, but by promoting scarcity and competition this project aggravates the over-accumulation of capital and exacerbates the social inequalities that it is supposed to combat According to some scholars , Bitcoin should be viewed like a new technology, not just a currency: It has paved the way for new types of currencies that utilize new technological infrastructures and whose dynamics should not be ignored.

Reception

Some economists have responded positively to bitcoin, including François R. Velde, Senior Economist at the Chicago Fed, who described it as "an elegant solution to the problem of creating a digital currency." Paul Krugman and Brad DeLong have found fault with bitcoin questioning why it should act as a reasonably stable store of value or whether there is a floor on their value. Economist John Quiggin has criticized bitcoin as "the final refutation of the efficient-market hypothesis".

David Andolfatto, a Vice President at the Federal Reserve Bank of St. Louis, stated that bitcoin is a threat to the establishment, which he argues is a good thing for the Federal Reserve System and other central banks because it prompts these institutions to operate sound policies.

Free software movement activist Richard Stallman has criticized the lack of anonymity and called for reformed development. PayPal President David A. Marcus calls bitcoin a "great place to put assets" but claims it will not be a

currency until price volatility is reduced. As bitcoins proved popular, they have been increasingly covered by comics around the world.

Acceptance by merchants

Established firms that accept bitcoins include Atomic Mall, Clearly Canadian, Dish Network, Overstock.com, the Sacramento Kings, TigerDirect, Virgin Galactic, and Zynga.

In late 2013 the University of Nicosia became the first university in the world to accept it.

Financial institutions

As of 2014, bitcoin companies have had difficulty opening traditional bank accounts because lenders have been leery of bitcoin's links to illicit activity. According to a co-founder of one such company, BitPay, "banks are scared to deal with bitcoin companies, even if they really want to". Yet, some financial institutions have been bullish on bitcoin. In a 2013 report, Bank of America Merrill Lynch stated that "we believe bitcoin can become a major means of payment for e-commerce and may emerge as a serious competitor to traditional money-transfer providers. As a medium of exchange, bitcoin has clear potential for growth and that in a long-term fair-value analysis maximum market capitalization for bitcoins could be $15 billion." In June 2014, the first bank that converts deposits in currencies instantly to Bitcoin without any fees, for further transactions, was opened in Boston.

Concurrent with Bloomberg LP, 33% owned by Merrill Lynch launching pricing information is the development of high-frequency trading firms by Atlas ATS in New York and Hong Kong and one from London-based Coinfloor, claiming to be the first auditable bitcoin exchange, and a SecondMarket project of an exchange for institutional investors.

Legal and journalistic opinions

Bitcoins have been evaluated and treated in various ways around the world. While it may appear that a lively debate over the nature of bitcoins is occurring, the different classifications are likely more reflective of what rules apply to bitcoin in a specific jurisdiction than what bitcoin really is. Magistrate Judge Amos Maazant of a Texas court classified bitcoins as currency. A German court found bitcoin to be a unit of account. The Finnish Government judged it to be a commodity. The People's Bank of China has stated that bitcoin "is fundamentally not a currency".

A WSJ journalist declared bitcoins a commodity in December 2013. A Forbes journalist referred to bitcoins as digital collectible. Two University of Amsterdam computer scientists proposed the term 'money-like informational commodity' in order "to allow for a systematic discussion of its development through all stages including an initial stage and a possible demise without being constrained by the implications of it being a money or a near-money".

Legal status and regulation

Main article: Legal status of Bitcoin

Few governments have moved to regulate bitcoin and similar private currencies. According to the European Central Bank, traditional financial sector regulation is not applicable because bitcoin does not involve traditional financial actors. Under other regimes, existing rules have been extended to include bitcoin and bitcoin companies. Steven Strauss, a Harvard public policy professor, suggested in April 2013 governments could outlaw bitcoin, a possibility that was mentioned in a 2013 SEC filing made by a bitcoin investment vehicle. A detailed survey of forty foreign jurisdictions and the European Union is maintained by the US Library of Congress.

India

Digital or virtual currencies such as Bitcoin have gained widespread acceptance in India despite a natural scepticism to assets not backed by tangible entities such as land. After the RBI warning in December 2013, a number of Bitcoin operators shut shop.Wikipedia:Citation needed The actions of the ED (enforcement directorate) and the I-T (income-tax) department have sent tremors throughout the mainstream Bitcoin community in India, if only for the reason that there is still no official regulation on how companies involved in dealing with digital currencies should comply with anti-money laundering and financial transaction laws.

Canada

The Canadian government announced in February 2014 that it was going to regulate bitcoin under existing anti-money laundering and counter-terrorist financing legislation.[16] In Quebec, The Financial Markets Authority stated in regards to bitcoin ATMs, that it would prosecute any violation of the Securities Act, the Derivatives Act, or the Money Services Business Act.[17]

China

China restricted bitcoin exchange for local currency in December 2013. On 10 April 2014 the People's Bank of China ordered banks and all third-party payment services to stop dealing with anyone in the bitcoin business. The ruling de-funds all Chinese bitcoin trading websites, as they will no longer have bank accounts in China.

Cyprus

The use of bitcoins is not regulated in Cyprus. On 11 December 2013, the Central Bank of Cyprus issued a statement on bitcoins, stating that "it considers the use of any kind of virtual money as particularly dangerous, given that it is not under any regulatory system and its operation is unchecked."

Hong Kong

Pre-existing Hong Kong law covers acts of fraud and money laundering involving virtual commodities.

Indonesia

A spokesman for Bank Indonesia reportedly issued a statement on Bitcoin in December 2013, saying that "bitcoin is a potential payment method, but it's different than ordinary currency... It is not regulated by the central bank so there are risks... At the moment, we're studying bitcoin and we have no plan to issue a regulation on it."

Japan

No laws in Japan regulate the use of bitcoins. Haruhiko Kuroda, governor of the Bank of Japan (BOJ), stated in December, 2013, that BOJ was "researching issues of bitcoins, but I have nothing to say regarding bitcoins at the moment."[18]

Russian Federation

On 27 January 2014, the Central Bank of the Russian Federation issued a statement entitled "On Using Virtual Currencies, Specifically Bitcoin, in Transactions." According to the statement, the Central Bank views the services of Russian legal entities aimed at assisting in the exchange of bitcoins for goods, services, or currencies as a "dubious activity" associated with money laundering and terrorism financing, and recommends that Russian individuals and legal entities refrain from transactions involving bitcoins.

Singapore

The Monetary Authority of Singapore may require bitcoin intermediaries to collect personal details of their customers and report suspicious activity similar to what it requires from money changers.

USA

In the US the first step of regulation occurred in July 2011, when the US Department of Treasury's Financial Crimes Enforcement Network added "other value that substitutes for currency" to its definition of Money services businesses. In 2013 the Treasury issued new rules regarding virtual currencies, whereby exchanges (but not users) are considered money transmitters and must comply with rules to prevent money laundering and terrorist financing. Besides obtaining personal details of clients, bitcoin exchanges must verify that their customers are not on the Office of Foreign Asset Control's Specially Designated Nationals list. In April 2014, the Treasury confirmed that bitcoin cloud mining and escrow services are not classified as money transmitters.

The US Government Accountability Office reviewed virtual currencies upon the request of the Senate Finance Committee and in May 2013 recommended that the IRS formulate tax guidance for bitcoin businesses. On 25 March 2014, in time for 2013 tax filing, the IRS issued a guidance that virtual currency is treated as property for US federal tax purposes and that "an individual who 'mines' virtual currency as a trade or business [is] subject to self-employment tax."

The US Commodity Futures Trading Commission stated in March 2014 it was considering regulation of digital currencies. As of April 2014[19], there are no new rules at the state level although the New York State Department of Financial Services intends to propose regulations no later than the end of the second quarter of 2014. As of 11 March 2014[19], it has officially invited bitcoin exchanges to apply with them.

In January 2014, the US Securities and Exchange Commission (SEC) was “very focused” on whether Bitcoin-denominated stock exchanges were illegal, per its enforcement administrator and was inquiring into the gambling site SatoshiDice listing shares on Bitcoin exchange MPEx. In May it warned investors that "both fraudsters and promoters of high-risk investment schemes may target Bitcoin users." The SEC charged and settled with the former owner of SatoshiDice in June 2014 for selling securities without registering with the SEC.

The IRS classified bitcoins as a capital asset end of March 2014 and subject to taxes on capital gains.

On 8 May 2014, the US Federal Election Commission issued draft guidance to US politicians who want to receive Bitcoin donations. The Commission declined to declare bitcoins currency, opting to deem them items "of value."

In May 2014, Brett Stapper, the co-founder of Falcon Global Capital, registered to lobby members of Congress and federal agencies on issues related to Bitcoin.

In June 2014, in California, Assemblyman Roger Dickinson (D-Sacramento) wrote a law (Assembly Bill 129) [20] to legalize Bitcoin and all other alternative and digital currency, such as Litecoin, Dogecoin, Starbucks Stars, and Amazon Coins. However, Dickinson "thinks the federal government should regulate the cryptocurrency" and said "I saw this legislation as a ways of cleaning up the code in California to conform to reality".

International guidance

The 2013 G7's Financial Action Task Force published guidance for Internet-based payment services that defines "exchangers buying or selling digital currency for cash (or other digital currencies) [...] as a virtual bureau de change" and warns that "Internet-based payment services that allow third party funding from anonymous sources may face an increased risk of [money laundering/terrorist financing]" concluding that this may "pose challenges to countries in [anti-money laundering/counter terrorist financing] regulation and supervision."

Criminal activity

Bitcoins have been associated with online criminal behavior and so-called cybercriminals. Used to obfuscate online transactions, bitcoins are seized when deep web black markets are shut by authorities.[21] Criminal activities have stigmatized the currency and attracted the attention of financial regulators, legislative bodies, and law enforcement.[22] CNN has referred to bitcoin as a "shady online currency [that is] starting to gain legitimacy in certain parts of the world," and *The Washington Post* calls it "the currency of choice for seedy online activities." The FBI stated in a 2012 report that "bitcoin will likely continue to attract cyber-criminals who view it as a means to move or steal funds." Criminal activity involving bitcoin has largely centered around theft, money laundering, the use of botnets for mining, and the use of bitcoins in exchange for illegal items or services. "Like cash, it can be used for ill as well as for good." Certain nation states may feel that its use in circumventing capital controls is also undesirable. Despite claims made by non-profit Bitcoin Foundation that "cryptography is the reason no one can steal bitcoins," theft is widespread.[23]

Black markets

In 2012, it was estimated that 4.5% to 9% of all transactions of all bitcoin exchanges in the world were for drug trades on a single deep web drugs market, Silk Road. The bulk of bitcoin purchases during the time were speculative in nature, so drugs must have constituted a greater percentage of the actual goods purchased with bitcoins c. 2012. Silk Road was shut by US law enforcement in October 2013 leading to a short-term fall in the value of bitcoin.[24] Alternative sites were soon available, and in early 2014 the Australian Broadcasting Corporation reported that the closure of the Silk Road had little impact on the number of Australians selling drugs online, which had actually increased.

Several news outlets have asserted that the popularity of bitcoins hinges on the ability to use them to purchase illegal goods. Non-drug transactions were thought to be far less than the number involved in the purchase of drugs, and roughly one half of all transactions made using bitcoin c. 2013 were bets placed at a single online gambling website. One source stated online gun dealers use bitcoin to sell arms without background checks. The bitcoin community branded one site, Sheep Marketplace, as a scam when it prevented withdrawals and shut down after an alleged bitcoins theft. In a separate case, escrow accounts with bitcoins belonging to patrons of a different black market were hacked in early 2014.

Money laundering

Bitcoins may not be ideal for money laundering because all transactions are public. Authorities have expressed concerns, however. The European Banking Authority and the FBI have both stated that bitcoin may be used for money laundering.[25] In early 2014, an operator of a US bitcoin exchange was arrested for money laundering.

Ponzi scheme

Various journalists, US economist Nouriel Roubini, and the head of the Estonian central bank have voiced concerns that bitcoin may be "only a little more than a Ponzi scheme".[26] Bitcoin supporters disagree. A 2012 report by the European Central Bank states, "it [is not] easy to assess whether or not the bitcoin system actually works like a pyramid or Ponzi scheme."

In an alleged Ponzi scheme that utilized bitcoins, The Bitcoin Savings and Trust promised investors up to 7 percent weekly interest, and raised at least 700,000 bitcoins from 2011 to 2012. The SEC charged the company and its founder in 2013 "with defrauding investors in a Ponzi scheme involving bitcoin...".

Thefts

A theft is an unauthorized transfer from a bitcoin address using the private key to unlock the address. Because transactions are irreversible and the identity of users difficult to unmask, it is rare that stolen bitcoins are recovered and returned. Theft occurs on a regular basis despite claims made by the Bitcoin Foundation that theft is impossible. Generating and storing keys offline mitigates the risk of theft. Most large-scale thefts occur at exchanges or online wallet services that store the private keys of many users. The thief hacks an online wallet service by finding a bug in its website or spreading malware to computers holding the private keys

Many high-profile thefts have been reported. In late November 2013, an estimated $100 million in bitcoins were stolen from the online illicit goods marketplace Sheep Marketplace, which immediately closed. Users tracked the coins as they were processed and converted to cash, but no funds were recovered and no culprits identified. A different black market, Silk Road 2, stated that during a February 2014 hack bitcoins valued at $2.7 million were taken from escrow accounts. In late February 2014 Mt. Gox, one of the largest virtual currency exchanges, filed for bankruptcy in Tokyo after its computer system was hacked and approximately $477 million in bitcoins were stolen. Flexcoin, a bitcoin storage specialist based in Alberta, Canada, shut down on March 2014 after saying it discovered a theft of about $650,000 in bitcoins. Poloniex, a digital currency exchange, reported on March 2014 that it lost bitcoins valued at around $50,000.

Malware

Bitcoin-related malware includes software that steals bitcoins from users using a variety of techniques, software that uses infected computers to mine bitcoins, and different types of ransomware, which disable computers or prevent files from being accessed until some payment is made. Security company Dell SecureWorks said in February 2014 that it had identified 146 types of bitcoin malware; about half of it undetectable with standard antivirus scanners.

Unauthorized mining

In June 2011, Symantec warned about the possibility that botnets could mine covertly for bitcoins. Malware used the parallel processing capabilities of GPUs built into many modern video cards. Although the average PC with an integrated graphics processor is virtually useless for bitcoin mining, tens of thousands of PCs laden with mining malware could produce some results.

Several reports of employees or students using university or research computers to mine bitcoins have been published.

Botnet cases

In mid-August 2011, bitcoin mining botnets were detected, and less than three months later, bitcoin mining trojans had infected Mac OS X.

In April 2013, electronic sports organization E-Sports Entertainment was accused of hijacking 14,000 computers to mine bitcoins; the company later settled the case with the State of New Jersey.

German police arrested two people in December 2013 who customized existing botnet software to perform bitcoin mining, which police said had been used to mine at least $950,000 worth of bitcoins.

For four days in December 2013 and January 2014, Yahoo Europe hosted an ad containing bitcoin mining malware that infected an estimated two million computers. The software, called Sefnit, was first detected in mid-2013 and has been bundled with many software packages. Microsoft has been removing the malware through its Microsoft Security Essentials and other security software since January 2014.

Malware stealing bitcoins

Some malware can steal private keys for bitcoin wallets allowing the bitcoins themselves to be stolen. The most common type searches computers for cryptocurrency wallets to upload to a remote server where they can be cracked and their coins stolen. Many of these also log keystrokes to record passwords, often avoiding the need to crack the keys. A different approach detects when a bitcoin address is copied to a clipboard and quickly replaces it with a different address, tricking people into sending bitcoins to the wrong address. This method is effective because bitcoin transactions are irreversible.

Cases of theft

One virus, spread through the Pony botnet, was reported in February 2014 to have stolen up to $220,000 in cryptocurrencies including 335 bitcoins from 85 wallets. Security company Trustwave, which tracked the malware, reports that its latest version was able to steal 30 types of digital currency.

A type Mac malware active in August 2013, Bitvanity posed as a vanity wallet address generator and stole addresses and private keys from other bitcoin client software. A different trojan for Mac OS X, called CoinThief was reported in February 2014 to be responsible for multiple bitcoin thefts, including one user who lost 20 bitcoins. The software was hidden in versions of some cryptocurrency apps on Download.com and MacUpdate.

Ransomware

Another type of bitcoin-related malware is ransomware. One program called Cryptolocker, typically spread through legitimate-looking email attachments, encrypts the hard drive of an infected computer, then displays a countdown timer and demands a ransom, usually two bitcoins, to decrypt it. Police in Massachusetts said they paid a 2 bitcoin ransom in November 2013, worth more than $1,300 at the time, to decrypt one of their hard drives. Linkup, a combination ransomware and bitcoin mining program that surfaced in February 2014, disables internet access and demands credit card information to restore it, while secretly mining bitcoins.

Security

Further information: Bitcoin network

There are two main ways the blockchain ledger can be corrupted to steal bitcoins: by fraudulently adding to or modifying it. The bitcoin system protects the blockchain against both using a combination of digital signatures and cryptographic hashes.

The Addition Attack and digital signatures

Payers and payees are identified in the blockchain by their public cryptographic keys: most bitcoin transfers are from one public key to a different public key. (Actually, hashes of these keys are used in the blockchain, and are called "bitcoin addresses".) In principle, an attacker Eve could steal money from Alice and Bob by simply adding transactions to the blockchain ledger like *Alice pays Eve 100 bitcoins*, *Bob pays Eve 100 bitcoins*, and so on, using of course these people's bitcoin addresses instead of their names. The bitcoin protocol prevents this kind of theft by requiring every transfer to be digitally signed with the payer's private key; only signed transfers can be added to the blockchain ledger. Since Eve cannot forge Alice's signature, Eve cannot defraud Alice by adding an entry to the blockchain equivalent to *Alice pays Eve 100 bitcoins*. At the same time, anyone can verify Alice's signature using her public key, and therefore that she has authorized any transaction in the blockchain where she is the payer.

The Modification Attack and mining

The other principal way to steal bitcoins would be to modify blockchain ledger entries. Eve could buy something from Alice, like a sofa, by adding a signed entry to the blockchain ledger equivalent to *Eve pays Alice 100 bitcoins*. Later, after receiving the sofa, Eve could modify that blockchain ledger entry to read instead: *Eve pays Alice 1 bitcoin*, or even delete the entry. Digital signatures cannot prevent this attack: Eve can simply sign her entry again after modifying it.

To prevent modification attacks, the bitcoin system first requires entries be added to the blockchain not one at a time, but in groups or *blocks*. More importantly, each block must be accompanied by a cryptographic hash of three things: the hash of the previous block, the block itself, and a number called a *nonce*. A hash of only the first two items will, like any cryptographic hash, always have a fixed number of bits (e.g. 256 for SHA-256). The nonce is a number which, when included, yields a hash with a specified number of leading zero bits. Because cryptographic hashes are essentially random, in the sense that their output cannot be predicted from their inputs, there is only one known way to find the nonce: to try out integers one after the other, e.g. 1, then 2, then 3, and so on. This process is called mining. The larger the number of leading zeros, the longer on average it will take to find a requisite nonce. The bitcoin system constantly adjusts the number of leading zeros so that the average time to find a nonce is about ten minutes. That way, as computer hardware gets faster over the years, the bitcoin protocol will simply require more leading zero bits to make mining always last about ten minutes.

This system prevents modification attacks in part because an attacker has to recalculate all the hashes of the blocks after the modified one. In the example above, if Eve wants to change *100 bitcoins* to *1 bitcoin*, she will not only have to recompute the hash of the block that transaction is in, but of all the blocks that come after it; she will have to recreate the chain of blocks. She can do this, but it will take her time, about ten minutes on average per block. However, during that time the network will continue to add blocks, and it will do so much faster than Eve alone can mine. Eve would have to recalculate all the blocks before the network could add a new one, or at least catch up with or overtake the network's miners. To do this, she would have to have roughly as much computing power as much of the existing bitcoin miners combined. This would be very expensive and, if the bitcoin network is large enough, likely infeasible. Furthermore, because of financial incentives to mine described below, it will make more financial sense for Eve to devote her resources to normal bitcoin mining instead. Thus the system protects against fraudulent blockchain modifications by making them expensive and, if the attacker is rational, unappealing because they make less financial sense than becoming a miner. The more miners there are, the more expensive and less feasible such attacks become, making the whole system even more secure.

Double-spending

Bitcoin system is based on an innovative solution of a problem common to all digital currency and payment schemes: that of so-called double-spending. With paper money or physical coins, when the payer transfers money to the payee, the payer cannot keep a copy of that dollar bill or coin. With digital money, which is just a computer file, this is not the case, and the payer could in principle spend the same money again and again, copying the file over and over. With bitcoin, when Eve offers to pay Alice some bitcoins, Alice can always first check the blockchain ledger to verify that Eve actually owns that many bitcoins. Of course, Eve could try to pay many people simultaneously; but bitcoin can defend against that. If Eve offers to pay Alice some bitcoins in exchange for goods, Alice can stipulate that she will not deliver the goods until Eve's payment to Alice appears in the blockchain, which typically involves waiting about ten minutes.

Types of attacks

Race attack

If the transaction has no confirmations, shops and services which accept payment can be exposed to a so-called *race attack*. For example, two transactions are created for the same funds to be sent to different shops/services. System rules ensure that only one of those transactions can be added to the block chain.

Shops can take numerous precautions to reduce this type of attack. It is always good to consider whether you should accept transactions without any confirmation.

Finney attack

Another type of attack. Shops or services which accept transactions without any confirmation are affected. A *Finney attack* is an attack which requires the participation of a miner to premine a block sending the money to be defrauded back to the fraudster. The risk of such an attack cannot be reduced to nothing regardless of the preventative measures taken by shops or services, but it does require the participation of a miner and an ideal combination of contributing factors. It is no mean feat, the miner risks a potential loss of the block reward. Just as with the other type of attack, the shop or service must seriously consider its policies concerning transactions without any confirmation.Wikipedia:Citation needed

Vector76 attack

Also called an *attack with confirmation*, this is a combination of the 2 aforementioned attacks which gives the perpetrator the ability to spend funds twice simply with a confirmation.Wikipedia:Citation needed

Brute force attack

This attack is possible even if the shop or service is expecting several transaction confirmations. It requires the attacker to be in possession of relatively high-performance hardware (hash frequency).

The perpetrator sends a transaction to the shop paying for a product/service and at the same time continues looking for a connection in the block chain (block chain fork) which recognizes this transaction. After a certain number of confirmations, the shop sends the product. If the perpetrator has found more than n blocks at this point, he breaks his block chain fork and regains his money, but if the perpetrator has not succeeded in doing this, the attack can be deemed a failure and the funds are sent to the shop, as should be the case.

The success of this attack depends on the speed (hash frequency) of the attacker and the number of confirmations for the shop/service. For example, if the attacker possesses 10% of the calculation power of the bitcoin network and the shop expects 6 confirmations for a successful transaction, the probability of success of such an attack will be 0.1%.Wikipedia:Citation needed

>50% attack

If the perpetrator controls more than 50% of the bitcoin network power, the probability of success of the aforementioned attack will be 100%. By virtue of the fact that the perpetrator can generate blocks more often than the other part of the network, he can create his own block chain until it becomes longer than the "integral" part of the network.Wikipedia:Citation needed

In the media

A bitcoin documentary film called *The Rise and Rise of Bitcoin* made its debut at the Tribeca Film Festival in New York on 23 April 2014, chronicling bitcoin's origins to its explosive growth in 2013.[27]

Undergraduate students at the Massachusetts Institute of Technology will receive $100 in bitcoins "to better understand this emerging technology". A student had the idea of a Bitcoin Club and raised more than half a million

dollars from a high frequency trader.

Some US political candidates, including New York City Democratic Congressional candidate Jeff Kurzon have said they would accept campaign donations in bitcoin.

Notes

[1] https://blockchain.info/charts/total-bitcoins?timespan=all&showDataPoints=false&daysAverageString=1&show_header=true&scale=0&address=

[2] There is no uniform convention for bitcoin capitalization. Some sources use *Bitcoin*, capitalized, to refer to the technology and network and *bitcoin*, lowercase, to refer to the unit of account.<ref>

[3] For WSJ, see

- For Chronicle of Higher Ed, see

[4] For theft, see

- For lack of chargebacks, see

[5] For use of different address for each transaction, see

- For mixing services, see

[6] For ATMs, see

- For buying in person, see

[7] For wallets holding bitcoins, see

- For wallets storing bitcoins, see

[8] http://www.coindesk.com/bitcoin-version-0-9-0-brings-transaction-malleability-fixes-branding-change/.

[9] For 2011 stoppage, see

- For EFF accepting bitcoins as of 2013, see

[10] For Mt. Gox being a large exchange, see

- For suspended withdrawals, see

[11] For Magic: The Gathering, see

- For waning popularity, see

[12] To obtain 35,000 figure, 16,000 merchants signed up with Bitcoin payment processor Coinbase are added to 20,000 merchants signed to BitPay.

- For 16,000 Coinbase merchants, see
- For 20,000 BitPay merchants, see

[13] For widely accepted medium of exchange, see

- For common medium of exchange, see

[14] .Blogs.ft.com (16 April 2013). Retrieved 20 April 2013.

[15] For Andresen, see

- For Hearn, see

[16] Duhaime Law

[17] Duhaime Law

[18] Summary of Bank of Japan Press Conference, at 10 (http://www.boj.or.jp/announcements/press/kaiken_2013/kk1312c.pdf)

[19] http://en.wikipedia.org/w/index.php?title=Bitcoin&action=edit

[20] http://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201320140AB129&search_keywords=

[21] For obfuscation of transactions and seizure of bitcoins from black market Utopia, see

- For seizure of bitcoins from black market Silk Road, see

[22] For Bitcoin enthusiasts worries that the currency may be stigmatized as "drug barter tokens", see Chen, Adrian (1 June 2011). . Gawker.

- For attention by law enforcement and regulatory bodies, see

[23] For claim, see

- For widespread theft, see

[24] For law enforcement action, see

- For drop in value, see

[25] For FBI, see

- For EBA, see

[26] For journalist, see

- For economist, see
- For head of central bank, see

[27] http://newsbtc.com/2014/03/17/bitcoin-documentary-film-rise-rise-bitcoin-debut-tribeca-film-festival/

References

This article incorporates text from this source (http://en.bitcoinwiki.org/Double-spending)*, which is licensed under CC-BY-SA 3.0* (http://creativecommons.org/licenses/by-sa/3.0/).

External links

- Bitcoin (http://www.dmoz.org/Science/Social_Sciences/Economics/Financial_Economics/Currency_and_Money/Alternative_Monetary_Systems/Bitcoin) at DMOZ
- Regulation of Bitcoin in Selected Jurisdictions (http://www.loc.gov/law/help/bitcoin-survey/2014-010233 Compiled Report_.pdf?loclr=bloglaw) The Law Library of Congress
- Bitcoin video series (https://www.khanacademy.org/economics-finance-domain/core-finance/money-and-banking/bitcoin/v/bitcoin-what-is-it) at Khan Academy
- Bitcoin: a cryptographic currency (http://cert.inteco.es/extfrontinteco/img/File/intecocert/EstudiosInformes/int_bitcoin_en.pdf) INTECO

Bitcoin network

Main article: Bitcoin

The **Bitcoin network** is a peer-to-peer payment network that operates on a cryptographic protocol. Users send bitcoins, the unit of currency, by broadcasting digitally signed messages to the network using Bitcoin wallet software. Transactions are recorded into a distributed public database known as the block chain, with consensus achieved by a proof-of-work system called "mining". The block chain is distributed internationally using peer-to-peer filesharing technology similar to BitTorrent. The protocol was designed in 2008 and released in 2009 as open source software by "Satoshi Nakamoto", the pseudonym of the original developer or group of developers.

The network timestamps transactions by including them in blocks that form an ongoing chain called the block chain. Such blocks cannot be changed without redoing the work that was required to create each block since the modified block. The longest chain serves not only as proof of the sequence of events but also records that this sequence of events was verified by a majority of the Bitcoin network's computing power. As long as a majority of computing power is controlled by nodes that are not cooperating to attack the network, they will generate the longest chain of records and outpace attackers.

The network itself requires minimal structure to share transactions. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will. Upon reconnection, a node will download and verify new blocks from other nodes to complete its local copy of the block chain.

Transactions

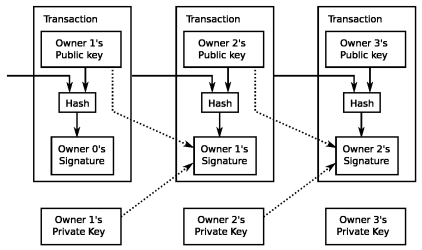
A transaction is a section of data confirmed by a signature. It is sent to the Bitcoin network and forms blocks. It typically contains references to preceding transactions and associates a certain number of bitcoins with one or several public keys (Bitcoin addresses). It is not encrypted because there is nothing to encrypt in the Bitcoin system. A block chain browser is where all transactions are combined in the form of a block chain. They can be found and verified. This is necessary to determine technical transaction parameters as well as verify the details of payments.[1]

A bitcoin is defined by a sequence of digitally signed transactions that began with its creation as a block reward. The owner of a bitcoin transfers it to the next owner by digitally signing it over to the next owner in a Bitcoin transaction, much like endorsing a traditional bank check. A payee can verify each previous transaction to verify the chain of

ownership.

A bitcoin is a currency object — an entity which is traded, though nothing prevents trades in fractions of, or multiple bitcoins. Bitcoins are intended to be fungible, though each has its own distinct history.

Although it would be possible to handle bitcoins individually, it would be unwieldy to make a separate transaction for every satoshi in a transfer. Transactions are therefore allowed to contain multiple inputs and outputs, and in that way bitcoins can be split and combined. Common transactions will have 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. Any difference between the total input and output amounts of a transaction is offered to miners as a transaction fee.



A diagram of a Bitcoin transfer.

Transaction confirmation

Transaction confirmation is needed to prevent double spending of the same money.

Usually when new bitcoins are earned, the owner isn't free to utilize them immediately. As soon as a transaction is started it is sent to the bitcoin network for processing and it has to be included in a block before becoming legitimate. The process of implementing a transaction in a newly found block is called a transaction confirmation. Inclusion in one block = one confirmation and when there are six or more of such confirmations the transaction is considered confirmed. This feature was introduced to protect the system from repeated spending of the same bitcoins (double-spending). Inclusion of transaction in the block happens along with the process of mining.

The classic Bitcoin client will show the transaction as "unconfirmed" until there are six confirmations (six found blocks). Sites or services that accept Bitcoin as payment for their products or services can set their own limits on how many blocks are needed to be found to confirm a transaction. The number six was chosen deliberately: it is based on a theory that there's low probability of wrongdoers being able to amass more than 10% of entire network's hash rate for purposes of transaction falsification and an insignificant risk (lower than 0.1%) is acceptable. For offenders who don't possess significant computing power, 6 confirmations are an insurmountable obstacle. In their turn people who possess more than 10% of power aren't going to find it hard to get 6 confirmations in a row. However to obtain such a power would require millions of dollars' worth of investments which lowers the risk of an attack. Bitcoins that are distributed by the network for finding a block can only be used after 100 confirmations e.g. 100 discovered blocks. The classic Bitcoin client won't display the coins earned for solving a block until there are 120 confirmations.[2]

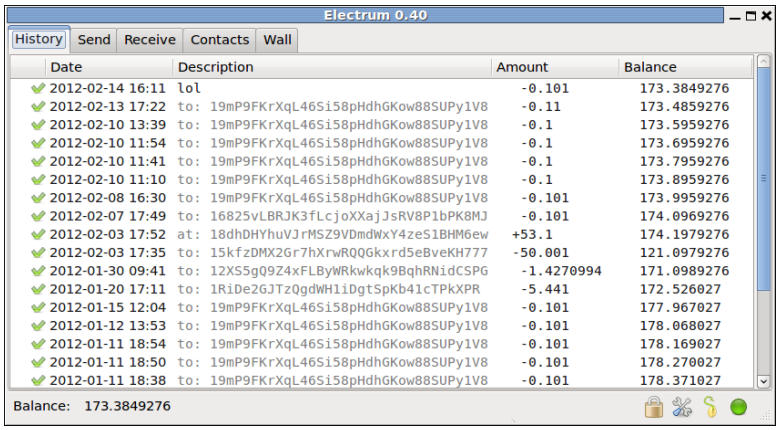
Hashes and signatures

Two consecutive SHA-256 hashes are used for transaction verification. RIPEMD-160 is used after a SHA-256 hash for Bitcoin digital signatures or "addresses". A Bitcoin address is the hash of an ECDSA public-key, computed as follows:

```
Key hash = Version concatenated with RIPEMD-160(SHA-256(public key))
Checksum = 1st 4 bytes of SHA-256(SHA-256(Key hash))
Bitcoin address = Base58Encode(Key hash concatenated with Checksum)
```

Bitcoin addresses

Bitcoin address is an identifier (account number), starting with 1 or 3 and containing 27-34 alphanumeric Latin characters (except 0, O, I, l). Address can be also represented as the QR-code, is anonymous and does not contain information about the owner. It can be obtained for free, using, for example, Bitcoin software.[3] Address example:



A Bitcoin transaction log showing addresses.

```
1AgRLMDWoJEjvd6xnu6ePqWXoSQQMWqo65
```

The ability to transact bitcoins without the assistance of a central registry is facilitated in part by the availability of a virtually unlimited supply of unique addresses which can be generated and disposed of at will. The balance of funds at a particular Bitcoin address can be ascertained by looking up the transactions to and from that address in the block chain. All valid transfers of bitcoins from an address are digitally signed using the private keys associated with it.

Bitcoin wallets

Bitcoin users manage their Bitcoin addresses by using a digital wallet. Wallets let users send bitcoins, request payment, calculate the total balance of addresses in use, and generate new addresses as needed. Many wallets include precautions to keep the private keys secret, for example by encrypting the wallet data with a password or by requiring two-factor authenticated logins. Bitcoin wallets provide the following functionality:[4]

- storage of Bitcoin addresses and corresponding closed/open keys on users computer in wallet.dat file,
- conduction transactions of obtaining and transferring Bitcoins (BTC), also without connection to internet
- providing information about the balance in BTC at all available addresses, prior transactions, spare keys.

Bitcoin wallets have been implemented as stand-alone software applications, web applications, and even printed documents or memorized passphrases.

Software wallets

Software that directly connects to the peer-to-peer Bitcoin network includes bitcoind and Bitcoin-Qt, the bitcoind GUI counterpart available for Linux, Windows, and Mac OS X. Other less resource intensive Bitcoin wallets have been developed, including mobile apps for iOS and Android devices that display and scan QR codes to simplify transactions between buyers and sellers.[5] Theoretically, the services typically provided by an application on a general purpose computer could be built into a stand-alone hardware device, and several projects aim to bring such a device to market.[6][7]

Website wallets

Many Bitcoin websites provide addresses associated with an online account to hold Bitcoin funds on the user's behalf, similar in ways to bank accounts. Other sites function primarily as real-time markets, facilitating the sale and purchase of bitcoins with other currencies such as US Dollars or Euros.[8] Users of this kind of wallet are not obliged to download all blocks of the Bitcoin block chain, and can manage one wallet with any device, regardless of location. Some wallets offer additional services. Wallet privacy is provided by the website operator. This "online" option is often preferred for the first acquaintance with Bitcoin system and short-term storage of small BTC amounts.[9]

Paper wallets

Example of a Bitcoin paper wallet with private key hidden beneath tamper-evident seals.

Any valid Bitcoin address keys may be printed on paper and used to store bitcoins offline. Compared with "hot wallets"—those that are connected to the Internet—these non-digital offline paper wallets are considered a "cold storage" mechanism better suited for safekeeping bitcoins. It is safe to use only if you have printed the paper yourself. Every such "cold storage" paper obtained from a second party as a present, gift, or payment should be immediately transferred to the safer wallet because the private key could have been copied and preserved by a grantor.

Various vendors offer banknotes, coins and cards denominated in bitcoins.[10] Bitcoin balance is bound to the private key printed on the banknote or embedded within the coin. Some of these instruments employ a tamper-evident seal that hides the private key. It is generally an insecure "cold storage" because one can't be sure that the producer of a banknote or a coin had destroyed the private key after the end of a printing process and doesn't preserve it. Tamper-evident seal in this case doesn't provide the needed level of security because the private key could be copied before the seal was applied on a coin. Some vendors will allow the user to verify the balance of a physical coin on their website, but that requires trusting that the vendor did not store the private key, which would allow them to transfer the balance at a future date.

To ensure **safety** of Bitcoin wallet the following measures are recommended:[11]

- wallet backup with printing or storing on flash drive in text editor without connection to Internet.
- encryption the wallet with the installation of a strong password.
- prudence when choosing a quality service.

Timestamps

The Bitcoin specification starts with the concept of a distributed timestamp server. A timestamp server works by taking a SHA256 hash function of some data and widely publishing the hash, for instance, in a newspaper or Usenet post. The timestamp proves that the data must have existed at the time, in order to produce the hash. For Bitcoin, each timestamp includes the previous timestamp hash as input for its own hash. This dependency of one hash on another is what forms a chain, with each additional timestamp providing evidence that each of the previous timestamp hashes existed.

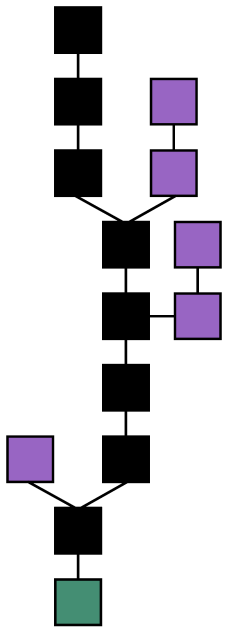
Bitcoin mining

To form a distributed timestamp server as a peer-to-peer network, Bitcoin uses a proof-of-work system similar to Adam Back's Hashcash and the internet rather than newspaper or Usenet posts. The work in this system is what is often referred to as Bitcoin mining.

The mining process involves scanning for a value that when hashed twice with SHA-256, begins with a number of zero bits. While the average work required increases exponentially with the number of leading zero bits required, a hash can always be verified by executing a single round of double SHA-256.

For the Bitcoin timestamp network, a valid "proof-of-work" is found by incrementing a nonce until a value is found that gives the block's hash the required number of leading zero bits. Once the hashing has produced a valid result, the block cannot be changed without redoing the work. As later records or "blocks" are chained after it, the work to change the block would include redoing the work for each subsequent block.

Majority consensus in Bitcoin is represented by the longest chain, which required the greatest amount of effort to produce it. If a majority of computing power is controlled by honest nodes, the honest chain will grow fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of that block and all blocks after it and then surpass the work of the honest nodes. The probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

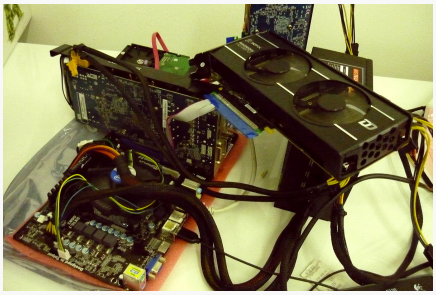


The best chain (black) consists of the longest series of transaction records from the genesis block (green) to the current block or record. Orphaned records (purple) exist outside of the best chain.

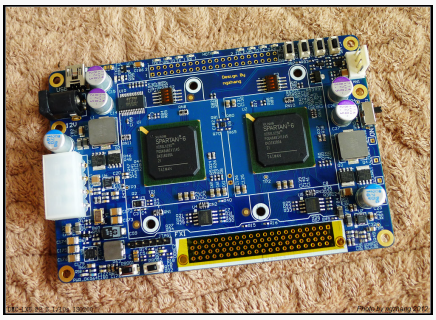
To compensate for increasing hardware speed and varying interest in running nodes over time, the difficulty of finding a valid hash is adjusted roughly every two weeks. If blocks were generated too quickly, the difficulty increases and more hashes are required to find a block and to generate new bitcoins.

Bitcoin mining [12] is a competitive endeavor. An arms race has been observed through the various hashing technologies that have been used to mine bitcoins: basic CPUs, high-end GPUs (graphics processing units) common in many gaming computers, FPGAs (field-programmable gate arrays) and ASICs (application-specific integrated circuits) all have been used with the latter reducing profitability of each former technology. The newest addition, ASICs, are built into devices that are specialized for Bitcoin mining. As bitcoins become more difficult to mine, computer hardware manufacturing companies have seen an increase in sales of high-end products.[13]

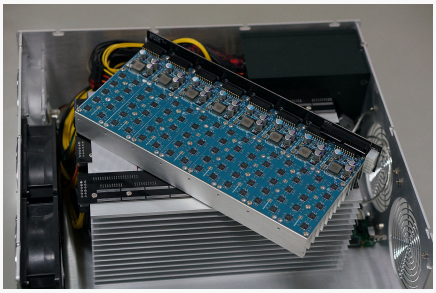
Computing power is often bundled together or "pooled" into a central server to reduce variance in miner income. Individual mining rigs often have to wait relatively long periods of time to confirm a block of transactions and receive payment. When miners cooperate in a pool, all participating miners receive a number of the bitcoins every time a participating server solves a block. This payment is proportional to the amount of work an individual miner contributed to help find that block.



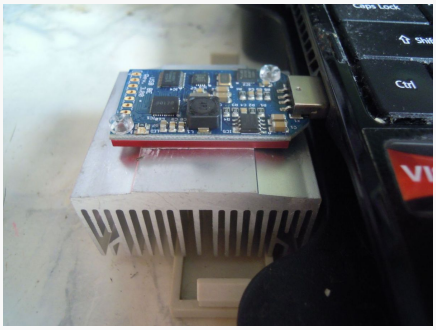
GPU-based mining rig



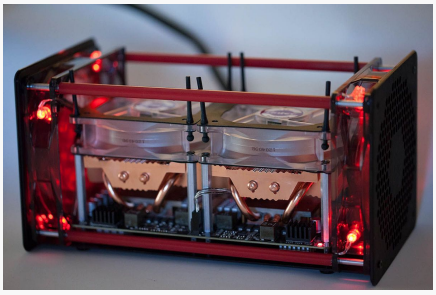
Lancelot FPGA-based mining board



Avalon ASIC-based mining machine



ASICMINER ASIC-based USB mining device



Butterfly Labs ASIC-based mining machine

Process

A rough overview of the process to mine bitcoins is:

1. New transactions are broadcast to all nodes.
2. Each miner node collects new transactions into a block.
3. Each miner 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. New bitcoins are successfully collected or "mined" by the receiving node which found the proof-of-work.
6. Nodes accept the block only if all transactions in it are valid and not already spent.
7. 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.
8. Repeat.

Nodes are incentivized to work on extending the longest chain or risk their work being wasted. 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, however, transactions 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.

Mined bitcoins

By convention, the first transaction in a block is a special transaction that produces new bitcoins 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 continual and steady addition of new coins is analogous to gold miners expending resources to add gold to circulation. In this case, it is computing power 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.

Local system resources

Once the latest transaction of a coin is buried under enough blocks, fully spent transactions which preceded it can be discarded in order to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle tree, 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 need not be stored.

A block header with no transactions would be about 80 bytes. Supposing that blocks are generated every 10 minutes, 80 bytes × 6 × 24 × 365 = 4.2 MB per year. With computer systems typically selling with 6 GB of RAM as of 2013, and Moore's law predicting current growth of 1.2 GB per year, storage should not be a problem even if the block headers need to be kept in memory.

Payment verification

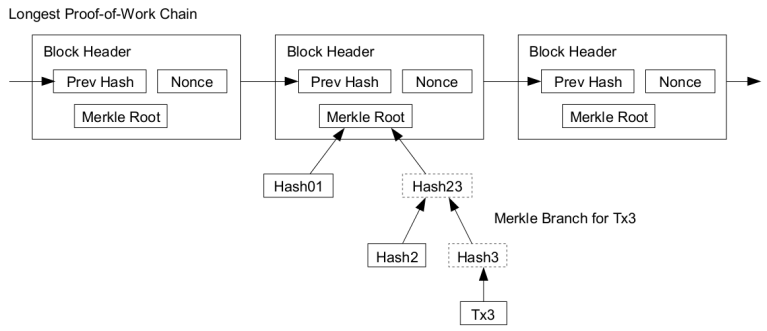


Diagram showing how Bitcoin transactions are verified.

It is possible to verify Bitcoin 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 are available by querying network nodes until it's apparent that the longest chain has been obtained. Then, get the Merkle branch linking the transaction to the block it is timestamped in. One can not check the transaction for oneself, but by linking it to a place in the chain, one 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. To protect against this, alerts from network nodes detecting an invalid block prompt the user's software to download the full block and verify alerted transactions to confirm their inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

References

[1] Bitcointransaction (http://en.bitcoinwiki.org/Bitcoin_transaction), Bitcoinwiki
[2] Transaction confirmation (http://en.bitcoinwiki.org/Transaction_confirmation), Bitcoinwiki
[3] Bitcoin address (http://en.bitcoinwiki.org/Bitcoin_address), Bitcoinwiki
[4] http://en.bitcoinwiki.org/Bitcoin_wallet
[5] Grilled cheese meets Bitcoin: Why this food truck is embracing digital currency (http://www.geekwire.com/2013/seattle-foodtruck-bitcoin-accept-payments-grilled-cheese-sandwiches/), GeekWire
[6] Bitcoin Hardware Wallet by Butterfly Labs (http://www.butterflylabs.com/bitcoin-hardware-wallet/)
[7] Trezor Bitcoin Hardware Wallet On Pace for October Deliveries (http://thegenesisblock.com/trezor-bitcoin-hardware-wallet-on-pace-for-october-deliveries/), The Genesis Block
[8] Web wallets (https://www.weusecoins.com/en/getting-started), weusecoins.com
[9] (http://en.bitcoinwiki.org/Bitcoin_wallet), Bitcoinwiki.org
[10] Physical Bitcoins by Casascius http://www.casascius.com/
[11] Security of Bitcoin wallet (http://en.bitcoinwiki.org/Bitcoin_wallet#Security), bitcoinwiki.org
[12] http://buybitcoinspaypal.com/bitcoins-mining/
[13] 2014-01-04, Bitcoin boom benefiting TSMC: report (http://www.taipeitimes.com/News/biz/archives/2014/01/04/2003580449), Taipei Times

External links

- Bitcoin Wiki (https://en.bitcoin.it/)

Certificate authority

In cryptography, a **certificate authority** or **certification authority** (**CA**) is an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others (relying parties) to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified. In this model of trust relationships, a CA is a trusted third party that is trusted by both the subject (owner) of the certificate and the party relying upon the certificate. CAs are characteristic of many public key infrastructure (PKI) schemes.

Overview

Trusted certificates are typically used to make secure connections to a server over the Internet. A certificate is required in order to avoid the case that a malicious party which happens to be on the path to the target server pretends to be the target. Such a scenario is commonly referred to as a man-in-the-middle attack. The client uses the CA certificate to verify the CA signature on the server certificate, as part of the checks before establishing a secure connection. Usually, client software—for example, browsers—include a set of trusted CA certificates. That makes sense inasmuch as users need to trust their client software: A malicious or compromised client can skip any security check and still fool its users into believing otherwise.

The customers of a CA are server administrators who need a certificate that their servers will present to clients. Commercial CAs charge to issue certificates, and their customers expect the CA's certificate to be included by most web browsers, so that secure connections to the certified server work smoothly out of the box. The number of web browsers and other devices and applications that trust a particular certificate authority is referred to as ubiquity. Mozilla, which is a non-profit organization, distributes several commercial CA certificates with its products. While Mozilla developed their own policy, the CA/Browser Forum developed similar guidelines for CA trust. A single CA certificate may be shared among multiple CAs or their resellers. A *root* CA certificate may be the base to issue multiple *intermediate* CA certificates with varying validation requirements.

Aside from commercial CAs, some providers issue digital certificates to the public at no cost; a noteworthy example is CAcert. Large institutions or government entities may have their own PKIs, each including their own CAs.

Formally, any site using self-signed certificates acts as its own CA too. At any rate, decent clients allow users to add or remove CA certificates at will. While server certificates usually last for a rather short period, CA certificates last much longer, so, for frequently visited servers, it is less error-prone to import and trust the CA that issues their certificates rather than confirm a security exception every time the server's certificate is renewed.

A less frequent usage of trusted certificates is for encrypting or signing messages. CAs issue end-user certificates too, which can be used with S/MIME. However, encryption requires the recipient's public key and, since authors and recipients of encrypted messages presumably know one another, the usefulness of a trusted third party remains confined to the signature verification of messages sent to public mailing lists.

Domain validation

The commercial CAs that issue the bulk of certificates that clients trust for email servers and public HTTPS servers typically use a technique called "domain validation" to authenticate the recipient of the certificate. Domain validation involves sending an email containing an authentication token or link, to an email address that is known to be administratively responsible for the domain. This could be the technical contact email address listed in the domain's WHOIS entry, or an administrative email like postmaster@ or root@ the domain. The theory behind domain validation is that only the legitimate owner of a domain would be able to read emails sent to these administrative addresses.

Domain validation suffers from certain structural security limitations. In particular, it is always vulnerable to attacks that allow an adversary to observe the domain validation emails that CAs send. These can include attacks against the DNS, TCP, or BGP protocols (which lack the cryptographic protections of TLS/SSL), or the compromise of routers. Such attacks are possible either on the network near a CA, or near the victim domain itself.

Some Certificate Authorities offer Extended Validation (EV) certificates as a more rigorous alternative to domain validated certificates. One limitation of EV as a solution to the weaknesses of domain validation is that attackers could still obtain a domain validated certificate for the victim domain, and deploy it during an attack; if that occurred, the only difference observable to the victim user would be a blue HTTPS address bar rather than a green one. Few users would be likely to recognise this difference as indicative of an attack being in progress.

Domain validation *implementations* have also sometimes been a source of security vulnerabilities. In one instance, security researchers showed that attackers could obtain certificates for webmail sites because a CA was willing to use an email address like `SSLCertificates@domain.com` for domain.com, but not all webmail systems had reserved the "SSLCertificates" username to prevent attackers from registering it.

Issuing a certificate

A CA issues digital certificates that contain a public key and the identity of the owner. The matching private key is not made available publicly, but kept secret by the end user who generated the key pair. The certificate is also a confirmation or validation by the CA that the public key contained in the certificate belongs to the person, organization, server or other entity noted in the certificate. A CA's obligation in such schemes is to verify an applicant's credentials, so that users and relying parties can trust the information in the CA's certificates. CAs use a variety of standards and tests to do so. In essence, the certificate authority is responsible for saying "yes, this person is who they say they are, and we, the CA, certify that".

If the user trusts the CA and can verify the CA's signature, then (s)he can also assume that a certain public key does indeed belong to whoever is identified in the certificate.

Example

Public-key cryptography can be used to encrypt data communicated between two parties. This can typically happen when a user logs on to any site that implements the HTTP Secure protocol. In this example let us suppose that the user logs on to his bank's homepage www.bank.example to do online banking. When the user opens www.bank.example homepage, he receives a public key along with all the data that his web-browser displays. The public key could be used to encrypt data from the client to the server but the safe procedure is to use it in a protocol that determines a shared symmetric encryption key; messages in such protocol are ciphered with the public key and only the bank server has the private key to read them. The rest of the communication proceeds using the new (disposable) symmetric key, so when the user enters some information to the bank's page and submits the page (sends the information back to the bank) then the data the user has entered to the page will be encrypted by his web browser. Therefore, even if someone can access the (encrypted) data that was communicated from the user to www.bank.example, such eavesdropper cannot read or decipher it.

This mechanism is only safe if the user can be sure that it is the bank that he sees in his web browser. If the user types in www.bank.example, but his communication is hi-jacked and a fake web-site (that pretends to be the bank web-site) sends the page information back to the user's browser, the fake web-page can send a fake public key to the user (for which the fake site owns a matching private key). The user will fill the form with his personal data and will submit the page. The fake web-page will get access to the user's data.

A certificate authority (CA) is an organization that stores public keys and their owners and every party in a communication trusts this organization (and knows its public key). When the user's web browser receives the public key from www.bank.example it also receives a digital signature of the key (with some more information, in a so-called X.509 certificate). The browser already possesses the public key of the CA and consequently can verify the signature, trust the certificate and the public key in it: since www.bank.example uses a public key that the certification authority certifies, a fake www.bank.example can only use the same public key. Since the fake www.bank.example does not know the corresponding private key, it cannot create the signature needed to verify its authenticity.

Security

The problem of assuring correctness of match between data and entity when the data are presented to the CA (perhaps over an electronic network), and when the credentials of the person/company/program asking for a certificate are likewise presented, is difficult. This is why commercial CAs often use a combination of authentication techniques including leveraging government bureaus, the payment infrastructure, third parties' databases and services, and custom heuristics. In some enterprise systems, local forms of authentication such as Kerberos can be used to obtain a certificate which can in turn be used by external relying parties. Notaries are required in some cases to personally know the party whose signature is being notarized; this is a higher standard than is reached by many CAs. According to the American Bar Association outline on Online Transaction Management the primary points of US Federal and State statutes enacted regarding digital signatures has been to "prevent conflicting and overly burdensome local regulation and to establish that electronic writings satisfy the traditional requirements associated with paper documents." Further the US E-Sign statute and the suggested UETA code help ensure that:

1. a signature, contract or other record relating to such transaction may not be denied legal effect, validity, or enforceability solely because it is in electronic form; and
2. a contract relating to such transaction may not be denied legal effect, validity or enforceability solely because an electronic signature or electronic record was used in its formation.

Despite the security measures undertaken to correctly verify the identities of people and companies, there is a risk of a single CA issuing a bogus certificate to an imposter. It is also possible to register individuals and companies with the same or very similar names, which may lead to confusion. To minimize this hazard, the *certificate transparency* initiative proposes auditing all certificates in a public unforgeable log, which could help in the prevention of

phishing.

In large-scale deployments, Alice may not be familiar with Bob's certificate authority (perhaps they each have a different CA server), so Bob's certificate may also include his CA's public key signed by a different CA_2, which is presumably recognizable by Alice. This process typically leads to a hierarchy or mesh of CAs and CA certificates.

Authority revocation lists

An *authority revocation list* (ARL) is a form of CRL containing certificates issued to certificate authorities, contrary to CRLs which contain revoked end-entity certificates.

Providers

Worldwide, the certificate authority business is fragmented, with national or regional providers dominating their home market. This is because many uses of digital certificates, such as for legally binding digital signatures, are linked to local law, regulations, and accreditation schemes for certificate authorities.

However, the market for SSL certificates, a kind of certificate used for website security, is largely held by a small number of multinational companies. This market has significant barriers to entry due to the technical requirements, while not legally required new providers may choose to undergo annual security audits (such as WebTrust for Certification Authorities in North American and ETSI in Europe) to be included in the list of web browser trusted authorities. More than 50 root certificates are trusted in the most popular web browser versions. A W3Techs survey from December 2013 shows:

- Symantec (which bought VeriSign's SSL interests and owns Thawte and Geotrust) with 38.1% market share
- Comodo Group with 29.1%
- Go Daddy with 13.4%
- GlobalSign with 10%

Industry organizations

- Certificate Authority Security Council (CASC) - In February 2013, the CASC was founded as an industry advocacy organization dedicated to addressing industry issues and educating the public on internet security. The founding members are the seven largest Certificate Authorities.
- Common Computing Security Standards Forum (CCSF) - In 2009 the CCSF was founded to promote industry standards that protect end users. Comodo Group CEO Melih Abdulhayoğlu is considered the founder of the CCSF.[1]Wikipedia:Verifiability
- CA/Browser Forum - In 2005, a new consortium of Certificate Authorities and web browser vendors was formed to promote industry standards and baseline requirements for internet security. Comodo Group CEO Melih Abdulhayoğlu organized the first meeting and is considered the founder of the CA/Browser Forum.

CA compromise

If the CA can be subverted, then the security of the entire system is lost; even as widely as all the entities that trust the compromised CA.

For example, suppose an attacker, Eve, manages to get a CA to issue to her a certificate that claims to represent Alice. That is, the certificate would publicly state that it represents Alice, and might include other information about Alice. Some of the information about Alice, such as her employer name, might be true, increasing the certificate's credibility. Eve, however, would have the all-important private key associated with the certificate. Eve could then use the certificate to send digitally signed email to Bob, tricking Bob into believing that the email was from Alice. Bob might even respond with encrypted email, believing that it could only be read by Alice, when Eve is actually

able to decrypt it using the private key.

A notable case of CA subversion like this occurred in 2001, when the certificate authority VeriSign issued two certificates to a person claiming to represent Microsoft. The certificates have the name "Microsoft Corporation", so could be used to spoof someone into believing that updates to Microsoft software came from Microsoft when they actually did not. The fraud was detected in early 2001. Microsoft and VeriSign took steps to limit the impact of the problem.

In 2011 fraudulent certificates were obtained from Comodo and DigiNotar, allegedly by Iranian hackers. There is evidence that the fraudulent DigiNotar certificates were used in a man-in-the-middle attack in Iran.

In 2012, it became known that Trustwave issued a subordinate root certificate that was used for transparent traffic management (man-in-the-middle) which effectively permitted an enterprise to sniff SSL internal network traffic using the subordinate certificate.

Open source implementations

There exist several open source implementations of certificate authority software. Common to all is that they provide the necessary services to issue, revoke and manage digital certificates.

Some open source implementations are:

- DogTag
- EJBCA
- gnoMint
- OpenCA
- OpenSSL, an SSL/TLS library that comes with tools allowing its use as a simple certificate authority
- Free Certificate Authority Management App [2] is a tool designed as a tutorial for learning about PKI, RSA Certificates, and ECC Certificates.
- r509
- TinyCA, which is a perl gui on top of some CPAN modules.
- XCA

References

[1] http://www.securitypark.co.uk/
[2] http://realtimelogic.com/blog/2014/05/How-to-act-as-a-Certificate-Authority-the-Easy-Way

External links

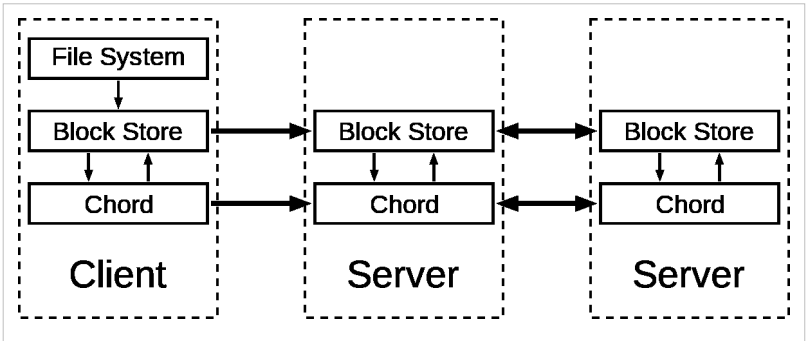
- Certificate authorities (http://www.dmoz.org/Computers/Security/Public_Key_Infrastructure/PKIX/Tools_and_Services/Third_Party_Certificate_Authorities/) at DMOZ
- Certificate Authority Reviews (http://www.sslshopper.com/certificate-authority-reviews.html)
- Certificate Authorities by Country (http://www.tractis.com/countries)
- How secure is HTTPS today? How often is it attacked? (https://www.eff.org/deeplinks/2011/10/how-secure-https-today), *Electronic Frontier Foundation* (25 October 2011)

Chord (peer-to-peer)

In computing, **Chord** is a protocol and algorithm for a peer-to-peer distributed hash table. A distributed hash table stores key-value pairs by assigning keys to different computers (known as "nodes"); a node will store the values for all the keys for which it is responsible. Chord specifies how keys are assigned to nodes, and how a node can discover the value for a given key by first locating the node responsible for that key.

Chord is one of the four original distributed hash table protocols, along with CAN, Tapestry, and Pastry. It was introduced in 2001 by Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan, and was developed at MIT.

Overview



IDs and keys are assigned an m -bit identifier using *consistent hashing*. The SHA-1 algorithm is the base hashing function for consistent hashing. Consistent hashing is integral to the robustness and performance of Chord because both keys and IDs (IP addresses) are uniformly distributed and in the same identifier space. Consistent hashing is also necessary to let nodes join and leave the network without disruption.

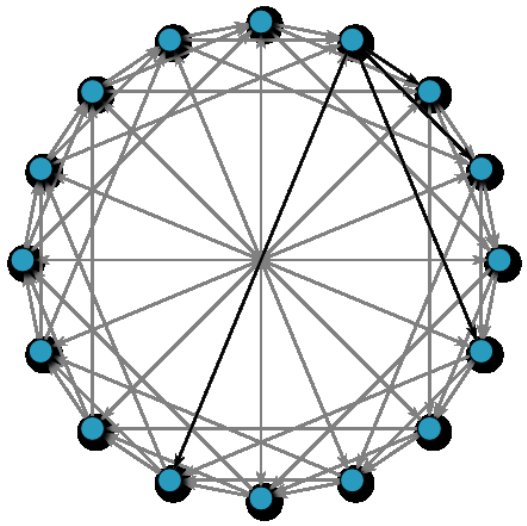
Using the Chord lookup protocol, node keys are arranged in a circle that has at most 2^m nodes. The circle can have IDs/keys ranging from 0 to $2^m - 1$.

Each node has a *successor* and a *predecessor*. The successor to a node (or key) is the next node in the identifier circle in a clockwise direction. The predecessor is counter-clockwise. If there is a node for each possible ID, the successor of node 2 is node 3, and the predecessor of node 1 is node 0; however, normally there are holes in the sequence. For example, the successor of node 153 may be node 167 (and nodes from 154 to 166 will not exist); in this case, the predecessor of node 167 will be node 153. That happens because not all nodes of the Chord are actual nodes. Some of them may be virtual ones.

Virtual nodes are stored into machines in the network that also contain at least one actual node. So successor and predecessor of each node points to the actual node of the Chord network.

Since the successor (or predecessor) node may disappear from the network (because of failure or departure), each node records a whole segment of the circle adjacent to it, i.e. the r nodes preceding it and the r nodes following it. This list results in a high probability that a node is able to correctly locate its successor or predecessor, even if the network in question suffers from a high failure rate.

Chord protocol



A 16-node Chord network. The "fingers" for one of the nodes are highlighted.

The Chord protocol is one solution for connecting the peers of a P2P network. There are many other algorithms in use by P2P, but this is a simple and common approach. Chord consistently maps a key onto a node. Both keys and nodes are assigned an m -bit identifier. For nodes, this identifier is a hash of the node's IP address. For keys, this identifier is a hash of a keyword, such as a file name. It is not uncommon to use the words "nodes" and "keys" to refer to these identifiers, rather than actual nodes or keys.

A logical ring with positions numbered $0$to $2^m - 1$ is formed among nodes. Key *k* is assigned to node successor(*k*), which is the node whose identifier is equal to or follows the identifier of *k*. If there are *N* nodes and *K* keys, then each node is responsible for roughly K/N keys.

When a new node joins or leaves the network, responsibility for $O(K/N)$keys changes hands.

If each node knows only the location of its successor, a linear search over the network could locate a particular key. This is a naive method for searching the network, since any given message could potentially have to be relayed through most of the network. Chord implements a faster search method.

Chord requires each node to keep a "finger table" containing up to m entries. The ith entry of node n will contain the address of successor $(n + 2^{i-1} \ mod \ 2^m)$.

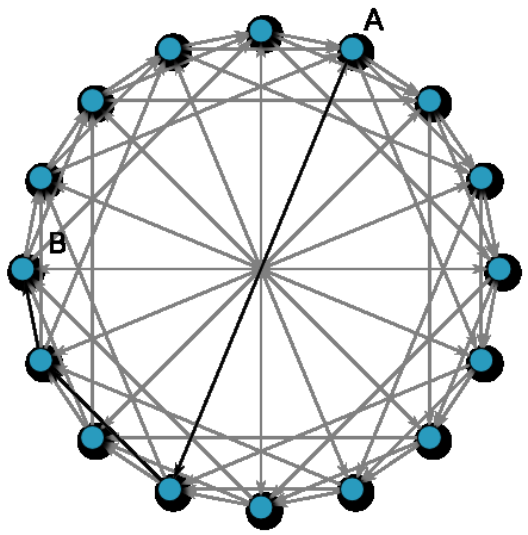
With such a finger table, the number of nodes that must be contacted to find a successor in an *N*-node network is $O(\log N)$. (See proof below.)

Potential uses

- Cooperative Mirroring: A load balancing mechanism by a local network hosting information available to computers outside of the local network. This scheme could allow developers to balance the load between many computers instead of a central server to ensure availability of their product.
- Time-shared storage: In a network, once a computer joins the network its available data is distributed throughout the network for retrieval when that computer disconnects from the network. As well as other computers' data is sent to the computer in question for offline retrieval when they are no longer connected to the network. Mainly for nodes without the ability to connect full-time to the network.
- Distributed Indices: Retrieval of files over the network within a searchable database. e.g. P2P file transfer clients.
- Large scale combinatorial searches: Keys being candidate solutions to a problem and each key mapping to the node, or computer, that is responsible for evaluating them as a solution or not. e.g. Code Breaking

Proof sketches

With high probability, Chord contacts $O(\log N)$ nodes to find a successor in an N-node network.



The routing path between nodes A and B. Each hop cuts the remaining distance in half (or better).

Suppose node n wishes to find the successor of key k. Let p be the predecessor of k. We wish to find an upper bound for the number of steps it takes for a message to be routed from n to p. Node n will examine its finger table and route the request to the closest predecessor of k that it has. Call this node f. If f is the ith entry n's finger table, then both f and p are at distances between 2^{i-1} and 2^i from n along the identifier circle. Hence, the distance between f and p along this circle is at most 2^{i-1}. Thus the distance from f to p is less than the distance from n to f: the new distance to p is at most half the initial distance.

This process of halving the remaining distance repeats itself, so after t steps, the distance remaining to p is at most $2^m/2^t$; in particular, after $\log N$ steps, the remaining distance is at most $2^m/N$. Because nodes are distributed uniformly at random along the identifier circle, the expected number of nodes falling within an interval of this length is 1, and with high probability, there are fewer than $\log N$ such nodes. Because the message always advances by at least one node, it takes at most $\log N$ steps for a message to traverse this remaining distance. The total expected routing time is thus $O(\log N)$.

If Chord keeps track of $r = O(\log N)$ predecessors/successors, then with high probability, if each node has probability of 1/4 of failing, find_successor (see below) and find_predecessor (see below) will return the correct nodes

Simply, the probability that all r nodes fail is $\left(\frac{1}{4}\right)^r = O\left(\frac{1}{N}\right)$, which is a low probability; so with high probability at least one of them is alive and the node will have the correct pointer.

Pseudocode

Definitions for pseudocode:

- finger[k]: first node that succeeds $(n + 2^{k-1}) \bmod 2^m, 1 \leq k \leq m$
- successor: the next node from the node in question on the identifier ring
- predecessor: the previous node from the node in question on the identifier ring

The pseudocode to find the *successor* node of an id is given below:

```
// ask node n to find the successor of id
n.find_successor(id)
  //Yes, that should be a closing square bracket to match the opening parenthesis.
  //It is a half closed interval.
  if (id ∈ (n, successor] )
    return successor;
  else
    // forward the query around the circle
```

```
    n0 = closest_preceding_node(id);
    return n0.find_successor(id);
```

```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
  for i = m downto 1
    if (finger[i]∈(n,id))
      return finger[i];
  return n;
```

The pseudocode to stabilize the chord ring/circle after node joins and departures is as follows:

```
// create a new Chord ring.
n.create()
  predecessor = nil;
  successor = n;
```

```
// join a Chord ring containing node n'.
n.join(n')
  predecessor = nil;
  successor = n'.find_successor(n);
```

```
// called periodically. n asks the successor
// about its predecessor, verifies if n's immediate
// successor is consistent, and tells the successor about n
n.stabilize()
  x = successor.predecessor;
  if (x∈(n, successor))
    successor = x;
  successor.notify(n);
```

```
// n' thinks it might be our predecessor.
n.notify(n')
  if (predecessor is nil or n'∈(predecessor, n))
    predecessor = n';
```

```
// called periodically. refreshes finger table entries.
// next stores the index of the finger to fix
n.fix_fingers()
  next = next + 1;
  if (next > m)
    next = 1;
  finger[next] = find_successor(n+2^(next−1));
```

```
// called periodically. checks whether predecessor has failed.
n.check_predecessor()
  if (predecessor has failed)
    predecessor = nil;
```

References

External links

- The Chord Project (http://www.pdos.lcs.mit.edu/chord)
- Open Chord - An Open Source Java Implementation (http://open-chord.sourceforge.net/)
- Chordless - Another Open Source Java Implementation (http://chordless.wiki.sourceforge.net/)
- jDHTUQ- An open source java implementation. API to generalize the implementation of peer-to-peer DHT systems. Contains GUI in mode data structure (http://sourceforge.net/projects/jdhtuq//)

Complex network

Network science
• **Theory**
• Graph • Complex network • Contagion • Small-world • Scale-free • Community structure • Percolation • Evolution • Controllability • Graph drawing • Social capital • Link analysis • Optimization • Reciprocity • Closure • Homophily • Transitivity • Preferential attachment • Balance theory • Network effect • Social influence
Network types
• Informational (computing) • Telecommunication • Social • Biological • Artificial neural • Interdependent

• Semantic • Random graph • Geometric • Dependency • Flow
Graphs
Features • Clique • Component • Cut • Cycle • Data structure • Edge • Loop • Neighborhood • Path • Vertex • Adjacency list / matrix • Incidence list / matrix *Types* • Bipartite • Complete • Directed • Hyper • Multi • Random • Weighted
• **Metrics** • **Algorithms**
• Centrality • Degree • Betweenness • Closeness • PageRank • Motif • Clustering • Degree distribution • Assortativity • Distance • Modularity
Models
• Random graph • Erdős–Rényi • Barabási–Albert • Watts–Strogatz • Exponential random (ERGM) • Epidemic • Hierarchical
Lists

• Topics • Software • Network scientists
Categories
• Graph theory • Network theory
• v • t • e [1]

In the context of network theory, a **complex network** is a graph (network) with non-trivial topological features—features that do not occur in simple networks such as lattices or random graphs but often occur in real graphs. The study of complex networks is a young and active area of scientific research inspired largely by the empirical study of real-world networks such as computer networks and social networks.

Definition

Most social, biological, and technological networks display substantial non-trivial topological features, with patterns of connection between their elements that are neither purely regular nor purely random. Such features include a heavy tail in the degree distribution, a high clustering coefficient, assortativity or disassortativity among vertices, community structure, and hierarchical structure. In the case of directed networks these features also include reciprocity, triad significance profile and other features. In contrast, many of the mathematical models of networks that have been studied in the past, such as lattices and random graphs, do not show these features. The most complex structures can be realized by networks with a medium number of interactions. This corresponds to the fact that the maximum information content (entropy) is obtained for medium probabilities.

Two well-known and much studied classes of complex networks are scale-free networks and small-world networks, whose discovery and definition are canonical case-studies in the field. Both are characterized by specific structural features—power-law degree distributions for the former and short path lengths and high clustering for the latter. However, as the study of complex networks has continued to grow in importance and popularity, many other aspects of network structure have attracted attention as well.

Recently, the study of complex networks has been expanded to networks of networks. If those networks are interdependent, they become significantly more vulnerable to random failures and targeted attacks and exhibit cascading failures and first-order percolation transitions.

The field continues to develop at a brisk pace, and has brought together researchers from many areas including mathematics, physics, biology, telecommunications, computer science, sociology, epidemiology, and others. Ideas from network science and engineering have been applied to the analysis of metabolic and genetic regulatory networks; the modeling and design of scalable communication networks such as the generation and visualization of complex wireless networks; the development of vaccination strategies for the control of disease; and a broad range of other practical issues. Research on networks has seen regular publication in some of the most visible scientific journals and vigorous funding in many countries, has been the topic of conferences in a variety of different fields, and has been the subject of numerous books both for the lay person and for the expert.

Scale-free networks

Main article: Scale-free networks

A network is named scale-free if its degree distribution, i.e., the probability that a node selected uniformly at random has a certain number of links (degree), follows a particular mathematical function called a power law. The power law implies that the degree distribution of these networks has no characteristic scale. In contrast, networks with a single well-defined scale are somewhat similar to a lattice in that every node has (roughly) the same degree. Examples of networks with a single scale include the Erdős–Rényi (ER) random graph and hypercubes. In a network with a scale-free degree distribution, some vertices have a degree that is orders of magnitude larger than the average - these vertices are often called "hubs", although this is a bit misleading as there is no inherent threshold above which a node can be viewed as a hub. If there were such a threshold, the network would not be scale-free.

Interest in scale-free networks began in the late 1990s with the reporting of discoveries of power-law degree distributions in real world networks such as the World Wide Web, the network of Autonomous systems (ASs), some networks of Internet routers, protein interaction networks, email networks, etc. Most of these reported "power laws" fail when challenged with rigorous statistical testing, but the more general idea of heavy-tailed degree distributions—which many of these networks do genuinely exhibit (before finite-size effects occur) -- are very different from what one would expect if edges existed independently and at random (i.e., if they followed a Poisson distribution). There are many different ways to build a network with a power-law degree distribution. The Yule process is a canonical generative process for power laws, and has been known since 1925. However, it is known by many other names due to its frequent reinvention, e.g., The Gibrat principle by Herbert A. Simon, the Matthew effect, cumulative advantage and, most recently, preferential attachment by Barabási and Albert for power-law degree distributions.

Some networks with a power-law degree distribution (and specific other types of structure) can be highly resistant to the random deletion of vertices—i.e., the vast majority of vertices remain connected together in a giant component. Such networks can also be quite sensitive to targeted attacks aimed at fracturing the network quickly. When the graph is uniformly random except for the degree distribution, these critical vertices are the ones with the highest degree, and have thus been implicated in the spread of disease (natural and artificial) in social and communication networks, and in the spread of fads (both of which are modeled by a percolation or branching process). While random graphs (ER) have an average distance of order log N between nodes, where N is the number of nodes, scale free graph can have a distance of log log N. Such graphs are called ultra small world networks.

Small-world networks

Main article: Small-world network

A network is called a small-world network by analogy with the small-world phenomenon (popularly known as six degrees of separation). The small world hypothesis, which was first described by the Hungarian writer Frigyes Karinthy in 1929, and tested experimentally by Stanley Milgram (1967), is the idea that two arbitrary people are connected by only six degrees of separation, i.e. the diameter of the corresponding graph of social connections is not much larger than six. In 1998, Duncan J. Watts and Steven Strogatz published the first small-world network model, which through a single parameter smoothly interpolates between a random graph and a lattice. Their model demonstrated that with the addition of only a small number of long-range links, a regular graph, in which the diameter is proportional to the size of the network, can be transformed into a "small world" in which the average number of edges between any two vertices is very small (mathematically, it should grow as the logarithm of the size of the network), while the clustering coefficient stays large. It is known that a wide variety of abstract graphs exhibit the small-world property, e.g., random graphs and scale-free networks. Further, real world networks such as the World Wide Web and the metabolic network also exhibit this property.

In the scientific literature on networks, there is some ambiguity associated with the term "small world." In addition to referring to the size of the diameter of the network, it can also refer to the co-occurrence of a small diameter and a high clustering coefficient. The clustering coefficient is a metric that represents the density of triangles in the network. For instance, sparse random graphs have a vanishingly small clustering coefficient while real world networks often have a coefficient significantly larger. Scientists point to this difference as suggesting that edges are correlated in real world networks.

Books

- Niloy Ganguly (Editor), Andreas Deutsch (Editor) and Animesh Mukherjee (Editor), *Dynamics On and Of Complex Networks Applications to Biology, Computer Science, and the Social Sciences*, 2009, ISBN 978-0-8176-4750-6
- Albert-László Barabási, *Linked: How Everything is Connected to Everything Else*, 2004, ISBN 0-452-28439-2
- Alain Barrat, Marc Barthelemy, Alessandro Vespignani, *Dynamical processes on complex networks*, Cambridge University Press, 2008, ISBN 978-0-521-87950-7
- Stefan Bornholdt (Editor) and Heinz Georg Schuster (Editor), *Handbook of Graphs and Networks: From the Genome to the Internet*, 2003, ISBN 3-527-40336-1
- Guido Caldarelli, *Scale-Free Networks* Oxford University Press, 2007, ISBN 978-0-19-921151-7
- E. Estrada, "The Structure of Complex Networks: Theory and Applications", Oxford University Press, 2011, ISBN 978-0-199-59175-6
- Reuven Cohen and Shlomo Havlin, *Complex Networks: Structure, Robustness and Function*, Cambridge University Press, 2010, ISBN 978-0-521-84156-6
- S.N. Dorogovtsev and J.F.F. Mendes, *Evolution of Networks: From biological networks to the Internet and WWW*, Oxford University Press, 2003, ISBN 0-19-851590-1
- Mark Newman, *Networks: An Introduction*, Oxford University Press, 2010, ISBN 978-0-19-920665-0
- Mark Newman, Albert-László Barabási, and Duncan J. Watts, *The Structure and Dynamics of Networks*, Princeton University Press, Princeton, 2006, ISBN 978-0-691-11357-9
- R. Pastor-Satorras and A. Vespignani, *Evolution and Structure of the Internet: A statistical physics approach*, Cambridge University Press, 2004, ISBN 0-521-82698-5
- Duncan J. Watts, *Six Degrees: The Science of a Connected Age*, W. W. Norton & Company, 2003, ISBN 0-393-04142-5
- Duncan J. Watts, *Small Worlds: The Dynamics of Networks between Order and Randomness*, Princeton University Press, 2003, ISBN 0-691-11704-7

References

[1] http://en.wikipedia.org/w/index.php?title=Template:Network_science&action=edit

- D. J. Watts and S. H. Strogatz (1998). "Collective dynamics of 'small-world' networks". *Nature* **393**: 440–442. doi: 10.1038/30918 (http://dx.doi.org/10.1038/30918). PMID 9623998 (http://www.ncbi.nlm.nih.gov/pubmed/9623998).
- S. H. Strogatz (2001). "Exploring Complex Networks". *Nature* **410** (6825): 268–276. doi: 10.1038/35065725 (http://dx.doi.org/10.1038/35065725). PMID 11258382 (http://www.ncbi.nlm.nih.gov/pubmed/11258382).
- R. Albert and A.-L. Barabási (2002). "Statistical mechanics of complex networks". *Reviews of Modern Physics* **74**: 47–97. arXiv: cond-mat/0106096 (http://arxiv.org/abs/cond-mat/0106096). Bibcode: 2002RvMP...74...47A (http://adsabs.harvard.edu/abs/2002RvMP...74...47A). doi: 10.1103/RevModPhys.74.47 (http://dx.doi.org/10.1103/RevModPhys.74.47).

- S. N. Dorogovtsev and J.F.F. Mendes (2002). "Evolution of Networks". *Adv. Phys.* **51**: 1079. arXiv: cond-mat/0106144 (http://arxiv.org/abs/cond-mat/0106144). doi: 10.1080/00018730110112519 (http://dx.doi.org/10.1080/00018730110112519).
- M. E. J. Newman, The structure and function of complex networks (http://arxiv.org/abs/cond-mat/0303516), SIAM Review 45, 167-256 (2003)
- S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, *Critical phenomena in complex networks* (http://arxiv.org/abs/0705.0010), Rev. Mod. Phys. 80, 1275, (2008)
- R. Cohen, K. Erez, D. ben-Avraham, S. Havlin, " Resilience of the Internet to random breakdown (http://havlin.biu.ac.il/Publications.php?keyword=Resilience+of+the+Internet+to+random+breakdown&year=*&match=all)" *Phys. Rev. Lett*. 85, 4626 (2000). http://arxiv.org/abs/1004.3989
- R. Cohen, K. Erez, D. ben-Avraham, S. Havlin, " Breakdown of the Internet under intentional attack (http://havlin.biu.ac.il/Publications.php?keyword=Breakdown+of+the+Internet+under+intentional+attack&year=*&match=all)" *Phys. Rev. Lett*. 86, 3682 (2001)
- R. Cohen, S. Havlin, " Scale-free networks are ultrasmall (http://havlin.biu.ac.il/Publications.php?keyword=Scale-free+networks+are+ultrasmall&year=*&match=all)" *Phys. Rev. Lett*. 90, 058701 (2003)
- A. E. Motter (2004). "Cascade control and defense in complex networks". *Phys. Rev. Lett.* **93**. arXiv: cond-mat/0401074 (http://arxiv.org/abs/cond-mat/0401074). Bibcode: 2004PhRvL..93i8701M (http://adsabs.harvard.edu/abs/2004PhRvL..93i8701M). doi: 10.1103/PhysRevLett.93.098701 (http://dx.doi.org/10.1103/PhysRevLett.93.098701).

Computer network

Network science
• **Theory**
• Graph • Complex network • Contagion • Small-world • Scale-free • Community structure • Percolation • Evolution • Controllability • Graph drawing • Social capital • Link analysis • Optimization • Reciprocity • Closure • Homophily • Transitivity

• Preferential attachment • Balance theory • Network effect • Social influence
Network types
• Informational (computing) • Telecommunication • Social • Biological • Artificial neural • Interdependent • Semantic • Random graph • Geometric • Dependency • Flow
Graphs
Features • Clique • Component • Cut • Cycle • Data structure • Edge • Loop • Neighborhood • Path • Vertex • Adjacency list / matrix • Incidence list / matrix *Types* • Bipartite • Complete • Directed • Hyper • Multi • Random • Weighted
• **Metrics** • **Algorithms**
• Centrality • Degree • Betweenness • Closeness • PageRank • Motif • Clustering • Degree distribution • Assortativity • Distance • Modularity

Models
• Random graph • Erdős–Rényi • Barabási–Albert • Watts–Strogatz • Exponential random (ERGM) • Epidemic • Hierarchical
Lists
• Topics • Software • Network scientists
Categories
• Graph theory • Network theory
• v • t • e [1]

A **computer network** or **data network** is a telecommunications network that allows computers to exchange data. In computer networks, networked computing devices pass data to each other along data connections. The connections (network links) between nodes are established using either cable media or wireless media. The best-known computer network is the Internet.

Network computer devices that originate, route and terminate the data are called network nodes. Nodes can include hosts such as personal computers, phones, servers as well as networking hardware. Two such devices are said to be networked together when one device is able to exchange information with the other device, whether or not they have a direct connection to each other.

Computer networks support applications such as access to the World Wide Web, shared use of application and storage servers, printers, and fax machines, and use of email and instant messaging applications. Computer networks differ in the physical media used to transmit their signals, the communications protocols to organize network traffic, the network's size, topology and organizational intent.

History

See also: History of the Internet

Today, computer networks are the core of modern communication. All modern aspects of the public switched telephone network (PSTN) are computer-controlled. Telephony increasingly runs over the Internet Protocol, although not necessarily the public Internet. The scope of communication has increased significantly in the past decade. This boom in communications would not have been possible without the progressively advancing computer network. Computer networks, and the technologies that make communication between networked computers possible, continue to drive computer hardware, software, and peripherals industries. The expansion of related industries is mirrored by growth in the numbers and types of people using networks, from the researcher to the home user.

The following is a chronology of significant computer network developments:

- In the late 1950s, early networks of communicating computers included the military radar system Semi-Automatic Ground Environment (SAGE).

- In 1960, the commercial airline reservation system semi-automatic business research environment (SABRE) went online with two connected mainframes.
- In 1962, J.C.R. Licklider developed a working group he called the "Intergalactic Computer Network", a precursor to the ARPANET, at the Advanced Research Projects Agency (ARPA).
- In 1964, researchers at Dartmouth developed the Dartmouth Time Sharing System for distributed users of large computer systems. The same year, at Massachusetts Institute of Technology, a research group supported by General Electric and Bell Labs used a computer to route and manage telephone connections.
- Throughout the 1960s, Leonard Kleinrock, Paul Baran, and Donald Davies independently developed network systems that used packets to transfer information between computers over a network.
- In 1965, Thomas Marill and Lawrence G. Roberts created the first wide area network (WAN). This was an immediate precursor to the ARPANET, of which Roberts became program manager.
- Also in 1965, the first widely used telephone switch that implemented true computer control was introduced by Western Electric.
- In 1969, the University of California at Los Angeles, the Stanford Research Institute, the University of California at Santa Barbara, and the University of Utah were connected as the beginning of the ARPANET network using 50 kbit/s circuits.
- In 1972, commercial services using X.25 were deployed, and later used as an underlying infrastructure for expanding TCP/IP networks.
- In 1973, Robert Metcalfe wrote a formal memo at Xerox PARC describing Ethernet, a networking system that was based on the Aloha network, developed in the 1960s by Norman Abramson and colleagues at the University of Hawaii. In July 1976, Robert Metcalfe and David Boggs published their paper "Ethernet: Distributed Packet Switching for Local Computer Networks"[1] and collaborated on several patents received in 1977 and 1978. In 1979, Robert Metcalfe pursued making Ethernet an open standard.
- In 1976, John Murphy of Datapoint Corporation created ARCNET, a token-passing network first used to share storage devices.
- In 1995, the transmission speed capacity for Ethernet was increased from 10 Mbit/s to 100 Mbit/s. By 1998, Ethernet supported transmission speeds of a Gigabit. The ability of Ethernet to scale easily (such as quickly adapting to support new fiber optic cable speeds) is a contributing factor to its continued use today.

Properties

Computer networking may be considered a branch of electrical engineering, telecommunications, computer science, information technology or computer engineering, since it relies upon the theoretical and practical application of the related disciplines.

A computer network facilitates interpersonal communications allowing people to communicate efficiently and easily via email, instant messaging, chat rooms, telephone, video telephone calls, and video conferencing. Providing access to information on shared storage devices is an important feature of many networks. A network allows sharing of files, data, and other types of information giving authorized users may access information stored on other computers on the network. A network allows sharing of network and computing resources. Users may access and use resources provided by devices on the network, such as printing a document on a shared network printer. Distributed computing uses computing resources across a network to accomplish tasks. A computer network may be used by computer Crackers to deploy computer viruses or computer worms on devices connected to the network, or to prevent these devices from accessing the network (denial of service). A complex computer network may be difficult to set up. It may be costly to set up an effective computer network in a large organization.

Network topology

Main article: Network topology

The physical layout of a network is usually less important than the topology that connects network nodes. Most diagrams that describe a physical network are therefore topological, rather than geographic. The symbols on these diagrams usually denote network links and network nodes.

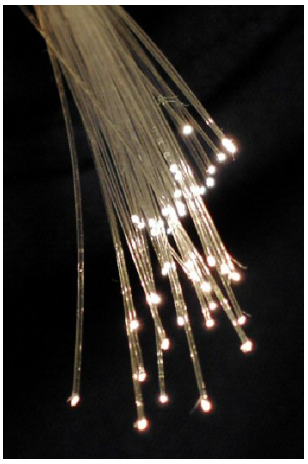
Network links

The communication media used to link devices to form a computer network include electrical cable (HomePNA, power line communication, G.hn), optical fiber (fiber-optic communication), and radio waves (wireless networking). In the OSI model, these are defined at layers 1 and 2 — the physical layer and the data link layer.

A widely adopted *family* of communication media used in local area network (LAN) technology is collectively known as Ethernet. The media and protocol standards that enable communication between networked devices over Ethernet are defined by IEEE 802.3. Ethernet transmit data over both copper and fiber cables. Wireless LAN standards (e.g. those defined by IEEE 802.11) use radio waves, or others use infrared signals as a transmission medium. Power line communication uses a building's power cabling to transmit data.

Wired technologies

The orders of the following wired technologies are, roughly, from slowest to fastest transmission speed.



Fiber optic cables are used to transmit light from one computer/network node to another

- *Twisted pair wire* is the most widely used medium for all telecommunication. Twisted-pair cabling consist of copper wires that are twisted into pairs. Ordinary telephone wires consist of two insulated copper wires twisted into pairs. Computer network cabling (wired Ethernet as defined by IEEE 802.3) consists of 4 pairs of copper cabling that can be utilized for both voice and data transmission. The use of two wires twisted together helps to reduce crosstalk and electromagnetic induction. The transmission speed ranges from 2 million bits per second to 10 billion bits per second. Twisted pair cabling comes in two forms: unshielded twisted pair (UTP) and shielded twisted-pair (STP). Each form comes in several category ratings, designed for use in various scenarios.
- *Coaxial cable* is widely used for cable television systems, office buildings, and other work-sites for local area networks. The cables consist of copper or aluminum wire surrounded by an insulating layer (typically a flexible material with a high dielectric constant), which itself is surrounded by a conductive layer. The insulation helps minimize interference and distortion. Transmission speed ranges from 200 million bits per second to more than 500 million bits per second.
- ITU-T G.hn technology uses existing home wiring (coaxial cable, phone lines and power lines) to create a high-speed (up to 1 Gigabit/s) local area network.
- An *optical fiber* is a glass fiber. It carries pulses of light that represent data. Some advantages of optical fibers over metal wires are very low transmission loss and immunity from electrical interference. Optical fibers can simultaneously carry multiple wavelengths of light, which greatly increases the rate that data can be sent, and helps enable data rates of up to trillions of bits per second. Optic fibers can be used for long runs of cable carrying very high data rates, and are used for undersea cables to interconnect continents.
- Price is a main factor distinguishing wired- and wireless-technology options in a business. Wireless options command a price premium that can make purchasing wired computers, printers and other devices a financial benefit. Before making the decision to purchase hard-wired technology products, a review of the restrictions and

limitations of the selections is necessary. Business and employee needs may override any cost considerations.[2]

Wireless technologies

Main article: Wireless network

Computers are very often connected to networks using wireless links

- *Terrestrial microwave* – Terrestrial microwave communication uses Earth-based transmitters and receivers resembling satellite dishes. Terrestrial microwaves are in the low-gigahertz range, which limits all communications to line-of-sight. Relay stations are spaced approximately 48 km (30 mi) apart.
- *Communications satellites* – Satellites communicate via microwave radio waves, which are not deflected by the Earth's atmosphere. The satellites are stationed in space, typically in geosynchronous orbit 35,400 km (22,000 mi) above the equator. These Earth-orbiting systems are capable of receiving and relaying voice, data, and TV signals.
- *Cellular and PCS systems* use several radio communications technologies. The systems divide the region covered into multiple geographic areas. Each area has a low-power transmitter or radio relay antenna device to relay calls from one area to the next area.
- *Radio and spread spectrum technologies* – Wireless local area networks use a high-frequency radio technology similar to digital cellular and a low-frequency radio technology. Wireless LANs use spread spectrum technology to enable communication between multiple devices in a limited area. IEEE 802.11 defines a common flavor of open-standards wireless radio-wave technology known as Wifi.
- *Free-space optical communication* uses visible or invisible light for communications. In most cases, line-of-sight propagation is used, which limits the physical positioning of communicating devices.

Exotic technologies

There have been various attempts at transporting data over exotic media:

- IP over Avian Carriers was a humorous April fool's Request for Comments, issued as **RFC 1149**. It was implemented in real life in 2001.
- Extending the Internet to interplanetary dimensions via radio waves.

Both cases have a large round-trip delay time, which gives slow two-way communication, but doesn't prevent sending large amounts of information.

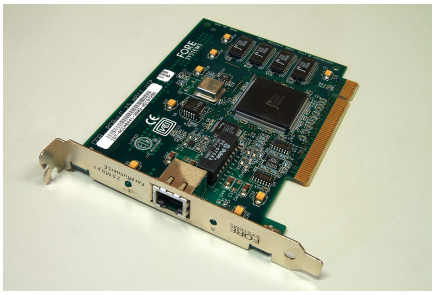
Network nodes

Main article: Node (networking)

Apart from the physical communications media described above, networks comprise additional basic system building blocks, such as network interface controller (NICs), repeaters, hubs, bridges, switches, routers, modems, and firewalls.

Network interfaces

A network interface controller (NIC) is computer hardware that provides a computer with the ability to access the transmission media, and has the ability to process low-level network information. For example the NIC may have a connector for accepting a cable, or an aerial for wireless transmission and reception, and the associated circuitry.



An ATM network interface in the form of an accessory card. A lot of network interfaces are built-in.

The NIC responds to traffic addressed to a network address for either the NIC or the computer as a whole.

In Ethernet networks, each network interface controller has a unique Media Access Control (MAC) address—usually stored in the controller's permanent memory. To avoid address conflicts between network devices, the Institute of Electrical and Electronics Engineers (IEEE) maintains and administers MAC address uniqueness. The size of an Ethernet MAC address is six octets. The three most significant octets are reserved to identify NIC manufacturers. These manufacturers, using only their assigned prefixes, uniquely assign the three least-significant octets of every Ethernet interface they produce.

Repeaters and hubs

A repeater is an electronic device that receives a network signal, cleans it of unnecessary noise, and regenerates it. The signal is retransmitted at a higher power level, or to the other side of an obstruction, so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable that runs longer than 100 meters. With fiber optics, repeaters can be tens or even hundreds of kilometers apart.

A repeater with multiple ports is known as a hub. Repeaters work on the physical layer of the OSI model. Repeaters require a small amount of time to regenerate the signal. This can cause a propagation delay that affects network performance. As a result, many network architectures limit the number of repeaters that can be used in a row, e.g., the Ethernet 5-4-3 rule.

Hubs have been mostly obsoleted by modern switches; but repeaters are used for long distance links, notably undersea cabling.

Bridges

A network bridge connects and filters traffic between two network segments at the data link layer (layer 2) of the OSI model to form a single network. This breaks the network's collision domain but maintains a unified broadcast domain. Network segmentation breaks down a large, congested network into an aggregation of smaller, more efficient networks.

Bridges come in three basic types:

- Local bridges: Directly connect LANs
- Remote bridges: Can be used to create a wide area network (WAN) link between LANs. Remote bridges, where the connecting link is slower than the end networks, largely have been replaced with routers.
- Wireless bridges: Can be used to join LANs or connect remote devices to LANs.

Switches

A network switch is a device that forwards and filters OSI layer 2 datagrams between ports based on the MAC addresses in the packets. A switch is distinct from a hub in that it only forwards the frames to the physical ports involved in the communication rather than all ports connected. It can be thought of as a multi-port bridge.[3] It learns to associate physical ports to MAC addresses by examining the source addresses of received frames. If an unknown destination is targeted, the switch broadcasts to all ports but the source. Switches normally have numerous ports, facilitating a star topology for devices, and cascading additional switches.

Multi-layer switches are capable of routing based on layer 3 addressing or additional logical levels. The term *switch* is often used loosely to include devices such as routers and bridges, as well as devices that may distribute traffic based on load or based on application content (e.g., a Web URL identifier).

Routers

A router is an internetworking device that forwards packets between networks by processing the routing information included in the packet or datagram (Internet protocol information from layer 3). The routing information is often processed in conjunction with the routing table (or forwarding table). A router uses its routing table to determine where to forward packets. (A destination in a routing table can include a "null" interface, also known as the "black hole" interface because data can go into it, however, no further processing is done for said data.)

A typical home or small office router showing the ADSL telephone line and Ethernet network cable connections

Modems

Modems (MOdulator-DEModulator) are used to connect network nodes via wire not originally designed for digital network traffic, or for wireless. To do this one or more frequencies are modulated by the digital signal to produce an analog signal that can be tailored to give the required properties for transmission. Modems are commonly used for telephone lines, using a Digital Subscriber Line technology.

Firewalls

A firewall is a network device for controlling network security and access rules. Firewalls are typically configured to reject access requests from unrecognized sources while allowing actions from recognized ones. The vital role firewalls play in network security grows in parallel with the constant increase in cyber attacks.

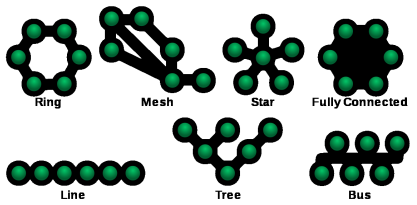
Network structure

Network topology is the layout or organizational hierarchy of interconnected nodes of a computer network. Different network topologies can affect throughput, but reliability is often more critical. With many technologies, such as bus networks, a single failure can cause the network to fail entirely. In general the more interconnections there are, the more robust the network is; but the more expensive it is to install.

Common layouts

Common layouts are:

- A bus network: all nodes are connected to a common medium along this medium. This was the layout used in the original Ethernet, called 10BASE5 and 10BASE2.
- A star network: all nodes are connected to a special central node. This is the typical layout found in a Wireless LAN, where each wireless client connects to the central Wireless access point.

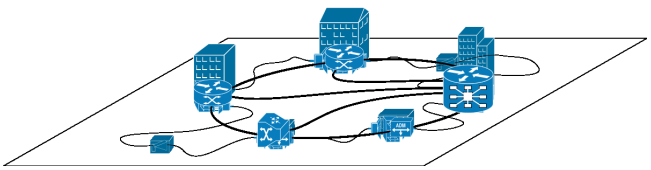


Common network topologies

- A ring network: each node is connected to its left and right neighbour node, such that all nodes are connected and that each node can reach each other node by traversing nodes left- or rightwards. The Fiber Distributed Data Interface (FDDI) made use of such a topology.
- A mesh network: each node is connected to an arbitrary number of neighbours in such a way that there is at least one traversal from any node to any other.
- A fully connected network: each node is connected to every other node in the network.
- A tree network: nodes are arranged hierarchically.

Note that the physical layout of the nodes in a network may not necessarily reflect the network topology. As an example, with FDDI, the network topology is a ring (actually two counter-rotating rings), but the physical topology is often a star, because all neighboring connections can be routed via a central physical location.

Overlay network


A sample overlay network

An overlay network is a virtual computer network that is built on top of another network. Nodes in the overlay network are connected by virtual or logical links. Each link corresponds to a path, perhaps through many physical links, in the underlying network. The topology of the overlay network may (and often does) differ from that of the underlying one. For example, many peer-to-peer networks are overlay networks. They are organized as nodes of a virtual system of links that run on top of the Internet.

Overlay networks have been around since the invention of networking when computer systems were connected over telephone lines using modems, before any data network existed.

The most striking example of an overlay network is the Internet itself. The Internet itself was initially built as an overlay on the telephone network. Even today, at the network layer, each node can reach any other by a direct

connection to the desired IP address, thereby creating a fully connected network. The underlying network, however, is composed of a mesh-like interconnect of sub-networks of varying topologies (and technologies). Address resolution and routing are the means that allow mapping of a fully connected IP overlay network to its underlying network.

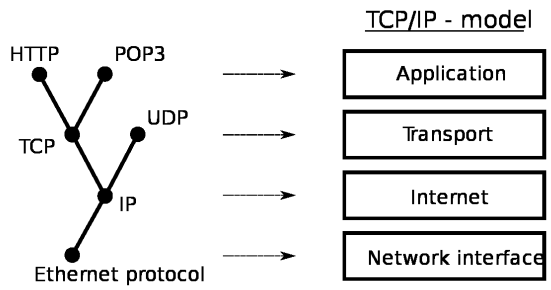
Another example of an overlay network is a distributed hash table, which maps keys to nodes in the network. In this case, the underlying network is an IP network, and the overlay network is a table (actually a map) indexed by keys.

Overlay networks have also been proposed as a way to improve Internet routing, such as through quality of service guarantees to achieve higher-quality streaming media. Previous proposals such as IntServ, DiffServ, and IP Multicast have not seen wide acceptance largely because they require modification of all routers in the network.Wikipedia:Citation needed On the other hand, an overlay network can be incrementally deployed on end-hosts running the overlay protocol software, without cooperation from Internet service providers. The overlay network has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for example, the sequence of overlay nodes that a message traverses before it reaches its destination.

For example, Akamai Technologies manages an overlay network that provides reliable, efficient content delivery (a kind of multicast). Academic research includes end system multicast, resilient routing and quality of service studies, among others.

Communications protocols

A communications protocol is a set of rules for exchanging information over network links. In a protocol stack (also see the OSI model), each protocol leverages the services of the protocol below it. An important example of a protocol stack is HTTP running over TCP over IP over IEEE 802.11. (TCP and IP are members of the Internet Protocol Suite. IEEE 802.11 is a member of the Ethernet protocol suite.) This stack is used between the wireless router and the home user's personal computer when the user is surfing the web.



The TCP/IP model or Internet layering scheme and its relation to common protocols often layered on top of it.

Whilst the use of protocol layering is today ubiquitous across the field of computer networking, it has been historically criticized by many researchers for two principle reasons. Firstly, abstracting the protocol stack in this way may cause a higher layer to duplicate functionality of a lower layer, a prime example being error recovery on both a per-link basis and an end-to-end basis. Secondly, it is common that a protocol implementation at one layer may require data, state or addressing information that is only present at another layer, thus defeating the point of separating the layers in the first place. For example, TCP uses the ECN field in the IPv4 header as an indication of congestion; IP is a network layer protocol whereas TCP is a transport layer protocol.

Communication protocols have various characteristics. They may be connection-oriented or connectionless, they may use circuit mode or packet switching, and they may use hierarchical addressing or flat addressing.

There are many communication protocols, a few of which are described below.

Ethernet

Ethernet is a family of protocols used in LANs, described by a set of standards together called IEEE 802 published by the Institute of Electrical and Electronics Engineers. It has a flat addressing scheme. It operates mostly at levels 1 and 2 of the OSI model. For home users today, the most well-known member of this protocol family is IEEE 802.11, otherwise known as Wireless LAN (WLAN). The complete IEEE 802 protocol suite provides a diverse set of networking capabilities. For example, MAC bridging (IEEE 802.1D) deals with the routing of Ethernet packets using a Spanning Tree Protocol, IEEE 802.1Q describes VLANs, and IEEE 802.1X defines a port-based Network Access Control protocol, which forms the basis for the authentication mechanisms used in VLANs (but it is also found in WLANs) – it is what the home user sees when the user has to enter a "wireless access key".

Internet Protocol Suite

The Internet Protocol Suite, also called TCP/IP, is the foundation of all modern networking. It offers connection-less as well as connection-oriented services over an inherently unreliable network traversed by data-gram transmission at the Internet protocol (IP) level. At its core, the protocol suite defines the addressing, identification, and routing specifications for Internet Protocol Version 4 (IPv4) and for IPv6, the next generation of the protocol with a much enlarged addressing capability.

SONET/SDH

Synchronous optical networking (SONET) and Synchronous Digital Hierarchy (SDH) are standardized multiplexing protocols that transfer multiple digital bit streams over optical fiber using lasers. They were originally designed to transport circuit mode communications from a variety of different sources, primarily to support real-time, uncompressed, circuit-switched voice encoded in PCM (Pulse-Code Modulation) format. However, due to its protocol neutrality and transport-oriented features, SONET/SDH also was the obvious choice for transporting Asynchronous Transfer Mode (ATM) frames.

Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is a switching technique for telecommunication networks. It uses asynchronous time-division multiplexing and encodes data into small, fixed-sized cells. This differs from other protocols such as the Internet Protocol Suite or Ethernet that use variable sized packets or frames. ATM has similarity with both circuit and packet switched networking. This makes it a good choice for a network that must handle both traditional high-throughput data traffic, and real-time, low-latency content such as voice and video. ATM uses a connection-oriented model in which a virtual circuit must be established between two endpoints before the actual data exchange begins.

While the role of ATM is diminishing in favor of next-generation networks, it still plays a role in the last mile, which is the connection between an Internet service provider and the home user. For an interesting write-up of the technologies involved, including the deep stacking of communications protocols used, see.

Geographic scale

A network can be characterized by its physical capacity or its organizational purpose. Use of the network, including user authorization and access rights, differ accordingly.

Personal area network

A personal area network (PAN) is a computer network used for communication among computer and different information technological devices close to one person. Some examples of devices that are used in a PAN are personal computers, printers, fax machines, telephones, PDAs, scanners, and even video game consoles. A PAN may include wired and wireless devices. The reach of a PAN typically extends to 10 meters. A wired PAN is usually constructed with USB and FireWire connections while technologies such as Bluetooth and infrared communication typically form a wireless PAN.

Local area network

A local area network (LAN) is a network that connects computers and devices in a limited geographical area such as a home, school, office building, or closely positioned group of buildings. Each computer or device on the network is a node. Wired LANs are most likely based on Ethernet technology. Newer standards such as ITU-T G.hn also provide a way to create a wired LAN using existing wiring, such as coaxial cables, telephone lines, and power lines.

A LAN is depicted in the accompanying diagram. All interconnected devices use the network layer (layer 3) to handle multiple subnets (represented by different colors). Those inside the library have 10/100 Mbit/s Ethernet connections to the user device and a Gigabit Ethernet connection to the central router. They could be called *Layer 3 switches*, because they only have Ethernet interfaces and support the Internet Protocol. It might be more correct to call them access routers, where the router at the top is a distribution router that connects to the Internet and to the academic networks' customer access routers.

The defining characteristics of a LAN, in contrast to a wide area network (WAN), include higher data transfer rates, limited geographic range, and lack of reliance on leased lines to provide connectivity. Current Ethernet or other IEEE 802.3 LAN technologies operate at data transfer rates up to 10 Gbit/s. The IEEE investigates the standardization of 40 and 100 Gbit/s rates. A LAN can be connected to a WAN using a router.

Home area network

A home area network (HAN) is a residential LAN used for communication between digital devices typically deployed in the home, usually a small number of personal computers and accessories, such as printers and mobile computing devices. An important function is the sharing of Internet access, often a broadband service through a cable TV or digital subscriber line (DSL) provider.

Storage area network

A storage area network (SAN) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to make storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system. A SAN typically has its own network of storage devices that are generally not accessible through the local area network by other devices. The cost and complexity of SANs dropped in the early 2000s to levels allowing wider adoption across both enterprise and small to medium sized business environments.

Campus area network

A campus area network (CAN) is made up of an interconnection of LANs within a limited geographical area. The networking equipment (switches, routers) and transmission media (optical fiber, copper plant, Cat5 cabling, etc.) are almost entirely owned by the campus tenant / owner (an enterprise, university, government, etc.).

For example, a university campus network is likely to link a variety of campus buildings to connect academic colleges or departments, the library, and student residence halls.

Backbone network

A backbone network is part of a computer network infrastructure that provides a path for the exchange of information between different LANs or sub-networks. A backbone can tie together diverse networks within the same building, across different buildings, or over a wide area.

For example, a large company might implement a backbone network to connect departments that are located around the world. The equipment that ties together the departmental networks constitutes the network backbone. When designing a network backbone, network performance and network congestion are critical factors to take into account. Normally, the backbone network's capacity is greater than that of the individual networks connected to it.

Another example of a backbone network is the Internet backbone, which is the set of wide area networks (WANs) and core routers that tie together all networks connected to the Internet.

Metropolitan area network

A Metropolitan area network (MAN) is a large computer network that usually spans a city or a large campus.

Wide area network

A wide area network (WAN) is a computer network that covers a large geographic area such as a city, country, or spans even intercontinental distances. A WAN uses a communications channel that combines many types of media such as telephone lines, cables, and air waves. A WAN often makes use of transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.

Enterprise private network

An enterprise private network is a network that a single organization builds to interconnect its office locations (e.g., production sites, head offices, remote offices, shops) so they can share computer resources.

Virtual private network

A virtual private network (VPN) is an overlay network in which some of the links between nodes are carried by open connections or virtual circuits in some larger network (e.g., the Internet) instead of by physical wires. The data link layer protocols of the virtual network are said to be tunneled through the larger network when this is the case. One common application is secure communications through the public Internet, but a VPN need not have explicit security features, such as authentication or content encryption. VPNs, for example, can be used to separate the traffic of different user communities over an underlying network with strong security features.

VPN may have best-effort performance, or may have a defined service level agreement (SLA) between the VPN customer and the VPN service provider. Generally, a VPN has a topology more complex than point-to-point.

Global area network

A global area network (GAN) is a network used for supporting mobile across an arbitrary number of wireless LANs, satellite coverage areas, etc. The key challenge in mobile communications is handing off user communications from one local coverage area to the next. In IEEE Project 802, this involves a succession of terrestrial wireless LANs.

Organizational scope

Networks are typically managed by the organizations that own them. Private enterprise networks may use a combination of intranets and extranets. They may also provide network access to the Internet, which has no single owner and permits virtually unlimited global connectivity.

Intranets

An intranet is a set of networks that are under the control of a single administrative entity. The intranet uses the IP protocol and IP-based tools such as web browsers and file transfer applications. The administrative entity limits use of the intranet to its authorized users. Most commonly, an intranet is the internal LAN of an organization. A large intranet typically has at least one web server to provide users with organizational information. An intranet is also

anything behind the router on a local area network.

Extranet

An extranet is a network that is also under the administrative control of a single organization, but supports a limited connection to a specific external network. For example, an organization may provide access to some aspects of its intranet to share data with its business partners or customers. These other entities are not necessarily trusted from a security standpoint. Network connection to an extranet is often, but not always, implemented via WAN technology.

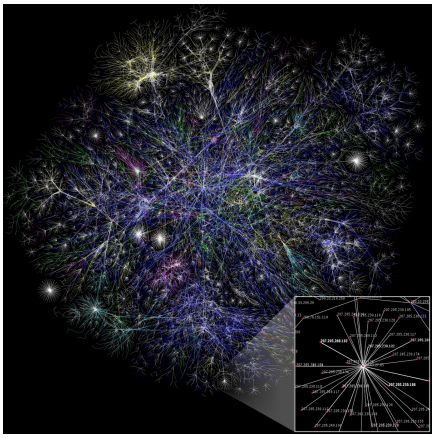
Internetwork

An internetwork is the connection of multiple computer networks via a common routing technology using routers.

Internet

The Internet is the largest example of an internetwork. It is a global system of interconnected governmental, academic, corporate, public, and private computer networks. It is based on the networking technologies of the Internet Protocol Suite. It is the successor of the Advanced Research Projects Agency Network (ARPANET) developed by DARPA of the United States Department of Defense. The Internet is also the communications backbone underlying the World Wide Web (WWW).

Participants in the Internet use a diverse array of methods of several hundred documented, and often standardized, protocols compatible with the Internet Protocol Suite and an addressing system (IP addresses) administered by the Internet Assigned Numbers Authority and address registries. Service providers and large enterprises exchange information about the reachability of their address spaces through the Border Gateway Protocol (BGP), forming a redundant worldwide mesh of transmission paths.



Partial map of the Internet based on the January 15, 2005 data found on opte.org [4]. Each line is drawn between two nodes, representing two IP addresses. The length of the lines are indicative of the delay between those two nodes. This graph represents less than 30% of the Class C networks reachable.

Darknet

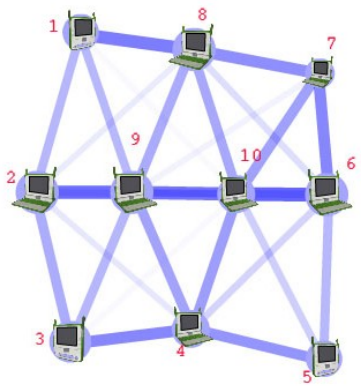
A Darknet is an overlay network, typically running on the internet, that is only accessible through specialized software. A darknet is an anonymizing network where connections are made only between trusted peers — sometimes called "friends" (F2F) — using non-standard protocols and ports.

Darknets are distinct from other distributed peer-to-peer networks as sharing is anonymous (that is, IP addresses are not publicly shared), and therefore users can communicate with little fear of governmental or corporate interference.

Routing

Routing is the process of selecting network paths to carry network traffic. Routing is performed for many kinds of networks, including circuit switching networks and packet switched networks.

In packet switched networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables, which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths.



Routing calculates good paths through a network for information to take. For example from node 1 to node 6 the best routes are likely to be 1-8-7-6 or 1-8-10-6, as this has the thickest routes.

There are usually multiple routes that can be taken, and to choose between them, different elements can be considered to decide which routes get installed into the routing table, such as (sorted by priority):

1. *Prefix-Length*: where longer subnet masks are preferred (independent if it is within a routing protocol or over different routing protocol)
2. *Metric*: where a lower metric/cost is preferred (only valid within one and the same routing protocol)
3. *Administrative distance*: where a lower distance is preferred (only valid between different routing protocols)

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.

Network service

Network services are applications hosted by servers on a computer network, to provide some functionality for members or users of the network, or to help the network itself to operate.

The World Wide Web, E-mail,[5] printing and network file sharing are examples of well-known network services. Network services such as DNS (Domain Name System) give names for IP and MAC addresses (people remember names like “nm.lan” better than numbers like “210.121.67.18”),[6] and DHCP to ensure that the equipment on the network has a valid IP address.[7]

Services are usually based on a service protocol that defines the format and sequencing of messages between clients and servers of that network service.

Network performance

Quality of service

Depending on the installation requirements, network performance is usually measured by the quality of service of a telecommunications product. The parameters that affect this typically can include throughput, jitter, bit error rate and latency.

The following list gives examples of network performance measures for a circuit-switched network and one type of packet-switched network, viz. ATM:

- Circuit-switched networks: In circuit switched networks, network performance is synonymous with the grade of service. The number of rejected calls is a measure of how well the network is performing under heavy traffic loads. Other types of performance measures can include the level of noise and echo.
- ATM: In an Asynchronous Transfer Mode (ATM) network, performance can be measured by line rate, quality of service (QoS), data throughput, connect time, stability, technology, modulation technique and modem enhancements.[8]

There are many ways to measure the performance of a network, as each network is different in nature and design. Performance can also be modelled instead of measured. For example, state transition diagrams are often used to model queuing performance in a circuit-switched network. The network planner uses these diagrams to analyze how the network performs in each state, ensuring that the network is optimally designed.

Network congestion

Network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queueing delay, packet loss or the blocking of new connections. A consequence of these latter two is that incremental increases in offered load lead either only to small increase in network throughput, or to an actual reduction in network throughput.

Network protocols that use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion—even after the initial load is reduced to a level that would not normally induce network congestion. Thus, networks using these protocols can exhibit two stable states under the same level of load. The stable state with low throughput is known as *congestive collapse*.

Modern networks use congestion control and congestion avoidance techniques to try to avoid congestion collapse. These include: exponential backoff in protocols such as 802.11's CSMA/CA and the original Ethernet, window reduction in TCP, and fair queueing in devices such as routers. Another method to avoid the negative effects of network congestion is implementing priority schemes, so that some packets are transmitted with higher priority than others. Priority schemes do not solve network congestion by themselves, but they help to alleviate the effects of congestion for some services. An example of this is 802.1p. A third method to avoid network congestion is the explicit allocation of network resources to specific flows. One example of this is the use of Contention-Free Transmission Opportunities (CFTXOPs) in the ITU-T G.hn standard, which provides high-speed (up to 1 Gbit/s) Local area networking over existing home wires (power lines, phone lines and coaxial cables).

For the Internet RFC 2914 addresses the subject of congestion control in detail.

Network resilience

Network resilience is "the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation."

Security

Network security

Network security consists of provisions and policies adopted by the network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of the computer network and its network-accessible resources. Network security is the authorization of access to data in a network, which is controlled by the network administrator. Users are assigned an ID and password that allows them access to information and programs within their authority. Network security is used on a variety of computer networks, both public and private, to secure daily transactions and communications among businesses, government agencies and individuals.

Network surveillance

Network surveillance is the monitoring of data being transferred over computer networks such as the Internet. The monitoring is often done surreptitiously and may be done by or at the behest of governments, by corporations, criminal organizations, or individuals. It may or may not be legal and may or may not require authorization from a court or other independent agency.

Computer and network surveillance programs are widespread today, and almost all Internet traffic is or could potentially be monitored for clues to illegal activity.

Surveillance is very useful to governments and law enforcement to maintain social control, recognize and monitor threats, and prevent/investigate criminal activity. With the advent of programs such as the Total Information Awareness program, technologies such as high speed surveillance computers and biometrics software, and laws such as the Communications Assistance For Law Enforcement Act, governments now possess an unprecedented ability to monitor the activities of citizens.

However, many civil rights and privacy groups—such as Reporters Without Borders, the Electronic Frontier Foundation, and the American Civil Liberties Union—have expressed concern that increasing surveillance of citizens may lead to a mass surveillance society, with limited political and personal freedoms. Fears such as this have led to numerous lawsuits such as *Hepting v. AT&T*. The hacktivist group Anonymous has hacked into government websites in protest of what it considers "draconian surveillance".[9][10]

End to end encryption

End-to-end encryption (E2EE) is a digital communications paradigm of uninterrupted protection of data traveling between two communicating parties. It involves the originating party encrypting data so only the intended recipient can decrypt it, with no dependency on third parties. End-to-end encryption prevents intermediaries, such as Internet providers or application service providers, from discovering or tampering with communications. End-to-end encryption generally protects both confidentiality and integrity.

Examples of end-to-end encryption include PGP for email, OTR for instant messaging, ZRTP for telephony, and TETRA for radio.

Typical server-based communications systems do not include end-to-end encryption. These systems can only guarantee protection of communications between clients and servers, not between the communicating parties themselves. Examples of non-E2EE systems are Google Talk, Yahoo Messenger, Facebook, and Dropbox. Some such systems, for example LavaBit and SecretInk, have even described themselves as offering "end-to-end" encryption when they do not. Some systems that normally offer end-to-end encryption have turned out to contain a

back door that subverts negotiation of the encryption key between the communicating parties, for example Skype.

The end-to-end encryption paradigm does not directly address risks at the communications endpoints themselves, such as the technical exploitation of clients, poor quality random number generators, or key escrow. E2EE also does not address traffic analysis, which relates to things such as the identities of the end points and the times and quantities of messages that are sent.

Views of networks

Users and network administrators typically have different views of their networks. Users can share printers and some servers from a workgroup, which usually means they are in the same geographic location and are on the same LAN, whereas a Network Administrator is responsible to keep that network up and running. A community of interest has less of a connection of being in a local area, and should be thought of as a set of arbitrarily located users who share a set of servers, and possibly also communicate via peer-to-peer technologies.

Network administrators can see networks from both physical and logical perspectives. The physical perspective involves geographic locations, physical cabling, and the network elements (e.g., routers, bridges and application layer gateways) that interconnect the physical media. Logical networks, called, in the TCP/IP architecture, subnets, map onto one or more physical media. For example, a common practice in a campus of buildings is to make a set of LAN cables in each building appear to be a common subnet, using virtual LAN (VLAN) technology.

Both users and administrators are aware, to varying extents, of the trust and scope characteristics of a network. Again using TCP/IP architectural terminology, an intranet is a community of interest under private administration usually by an enterprise, and is only accessible by authorized users (e.g. employees).[11] Intranets do not have to be connected to the Internet, but generally have a limited connection. An extranet is an extension of an intranet that allows secure communications to users outside of the intranet (e.g. business partners, customers).

Unofficially, the Internet is the set of users, enterprises, and content providers that are interconnected by Internet Service Providers (ISP). From an engineering viewpoint, the Internet is the set of subnets, and aggregates of subnets, which share the registered IP address space and exchange information about the reachability of those IP addresses using the Border Gateway Protocol. Typically, the human-readable names of servers are translated to IP addresses, transparently to users, via the directory function of the Domain Name System (DNS).

Over the Internet, there can be business-to-business (B2B), business-to-consumer (B2C) and consumer-to-consumer (C2C) communications. When money or sensitive information is exchanged, the communications are apt to be protected by some form of communications security mechanism. Intranets and extranets can be securely superimposed onto the Internet, without any access by general Internet users and administrators, using secure Virtual Private Network (VPN) technology.

References

[1] Ethernet: Distributed Packet Switching for Local Computer Networks (http://www.acm.org/classics/apr96/), Robert M. Metcalfe and David R. Boggs, Communications of the ACM (pp 395–404, Vol. 19, No. 5), July 1976.

[2] (http://smallbusiness.chron.com/disadvantages-wired-technology-17833.html), The Disadvantages of Wired Technology, Laura Acevedo, Demand Media.

[3] http://compnetworking.about.com/cs/internetworking/g/bldef_bridge.htm

[4] http://www.opte.org/maps/

[5] rfc5321

[6] RFC 1035, *Domain names - Implementation and Specification*, P. Mockapetris (November 1987)

[7] Peterson LL, Davie BS. (2011). Computer Networks: A Systems Approach (http://books.google.com/books?id=BvaFreun1W8C&pg=PA372&lpg=PA372).

[8] Telecommunications Magazine Online (http://www.telecommagazine.com), Americas January 2003, Issue Highlights, Online Exclusive: Broadband Access Maximum Performance, Retrieved on February 13, 2005.

[9] "Anonymous hacks UK government sites over 'draconian surveillance' " (http://www.zdnet.com/blog/security/anonymous-hacks-uk-government-sites-over-draconian-surveillance/11412), Emil Protalinski, ZDNet, 7 April 2012, retrieved 12 March 2013

[10] Hacktivists in the frontline battle for the internet (http://www.guardian.co.uk/technology/2012/apr/20/hacktivists-battle-internet) retrieved 17 June 2012
[11] RFC 2547

This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm).

Further reading

- Shelly, Gary, et al. "Discovering Computers" 2003 Edition
- Wendell Odom, Rus Healy, Denise Donohue. (2010) CCIE Routing and Switching. Indianapolis, IN: Cisco Press
- Kurose James F and Keith W. Ross : Computer Networking: A Top-Down Approach Featuring the Internet, Pearson Education 2005.
- William Stallings, *Computer Networking with Internet Protocols and Technology*, Pearson Education 2004.
- Important publications in computer networks
- Network Communication Architecture and Protocols: OSI Network Architecture 7 Layers Model

External links

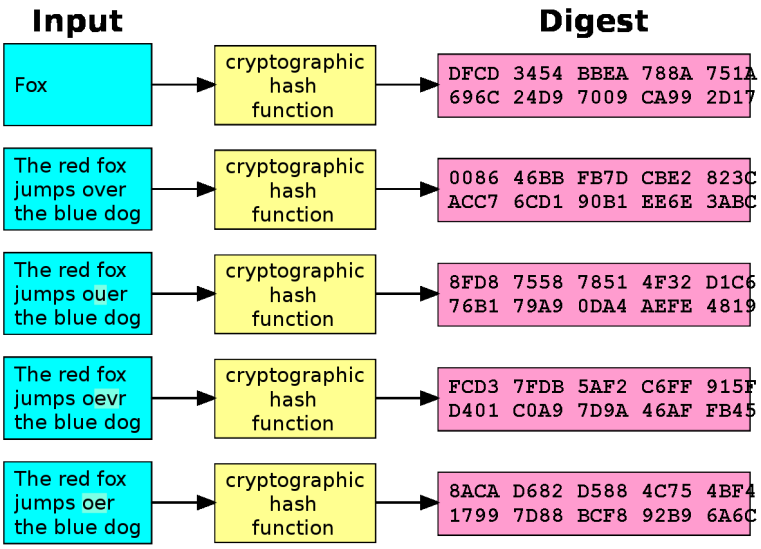
- Networking (http://www.dmoz.org/Computers/Software/Networking/) at DMOZ
- IEEE Ethernet manufacturer information (http://standards.ieee.org/regauth/oui/oui.txt)

Cryptographic hash function

A **cryptographic hash function** is a hash function which is considered practically impossible to invert, that is, to recreate the input data from its hash value alone. The input data is often called the *message*, and the hash value is often called the *message digest* or simply the *digest*.

The ideal cryptographic hash function has four main properties:

- it is easy to compute the hash value for any given message
- it is infeasible to generate a message that has a given hash
- it is infeasible to modify a message without changing the hash
- it is infeasible to find two different messages with the same hash.



A cryptographic hash function (specifically, SHA-1) at work. Note that even small changes in the source input (here in the word "over") drastically change the resulting output, by the so-called avalanche effect.

Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information security contexts, cryptographic hash values are sometimes

called (*digital*) *fingerprints*, *checksums*, or just *hash values*, even though all these terms stand for more general functions with rather different properties and purposes.

Properties

Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value.

A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. At a minimum, it must have the following properties:

- *Pre-image resistance*

  Given a hash h it should be difficult to find any message m such that $h = hash(m)$. This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks.
- *Second pre-image resistance*

  Given an input m_1 it should be difficult to find another input m_2 such that $m_1 \neq m_2$ and $hash(m_1) = hash(m_2)$. Functions that lack this property are vulnerable to second-preimage attacks.
- *Collision resistance*

  It should be difficult to find two different messages m_1 and m_2 such that $hash(m_1) = hash(m_2)$. Such a pair is called a cryptographic hash collision. This property is sometimes referred to as *strong collision resistance.* It requires a hash value at least twice as long as that required for preimage-resistance; otherwise collisions may be found by a birthday attack.

These properties imply that a malicious adversary cannot replace or modify the input data without changing its digest. Thus, if two strings have the same digest, one can be very confident that they are identical.

A function meeting these criteria may still have undesirable properties. Currently popular cryptographic hash functions are vulnerable to *length-extension* attacks: given $hash(m)$ and $len(m)$ but not m, by choosing a suitable m' an attacker can calculate $hash(m \parallel m')$ where $\parallel$ denotes concatenation. This property can be used to break naive authentication schemes based on hash functions. The HMAC construction works around these problems.

Ideally, one may wish for even stronger conditions. It should be impossible for an adversary to find two messages with substantially similar digests; or to infer any useful information about the data, given only its digest. Therefore, a cryptographic hash function should behave as much as possible like a random function while still being deterministic and efficiently computable.

Checksum algorithms, such as CRC32 and other cyclic redundancy checks, are designed to meet much weaker requirements, and are generally unsuitable as cryptographic hash functions. For example, a CRC was used for message integrity in the WEP encryption standard, but an attack was readily discovered which exploited the linearity of the checksum.

Degree of difficulty

In cryptographic practice, “difficult” generally means “almost certainly beyond the reach of any adversary who must be prevented from breaking the system for as long as the security of the system is deemed important”. The meaning of the term is therefore somewhat dependent on the application, since the effort that a malicious agent may put into the task is usually proportional to his expected gain. However, since the needed effort usually grows very quickly with the digest length, even a thousand-fold advantage in processing power can be neutralized by adding a few dozen bits to the latter.

In some theoretical analyses “difficult” has a specific mathematical meaning, such as "not solvable in asymptotic polynomial time". Such interpretations of *difficulty* are important in the study of provably secure cryptographic hash functions but do not usually have a strong connection to practical security. For example, an exponential time

algorithm can sometimes still be fast enough to make a feasible attack. Conversely, a polynomial time algorithm (e.g., one that requires n^{20} steps for n-digit keys) may be too slow for any practical use.

Illustration

An illustration of the potential use of a cryptographic hash is as follows: Alice poses a tough math problem to Bob and claims she has solved it. Bob would like to try it himself, but would yet like to be sure that Alice is not bluffing. Therefore, Alice writes down her solution, computes its hash and tells Bob the hash value (whilst keeping the solution secret). Then, when Bob comes up with the solution himself a few days later, Alice can prove that she had the solution earlier by revealing it and having Bob hash it and check that it matches the hash value given to him before. (This is an example of a simple commitment scheme; in actual practice, Alice and Bob will often be computer programs, and the secret would be something less easily spoofed than a claimed puzzle solution).

Applications

Verifying the integrity of files or messages

Main article: File verification

An important application of secure hashes is verification of message integrity. Determining whether any changes have been made to a message (or a file), for example, can be accomplished by comparing message digests calculated before, and after, transmission (or any other event).

For this reason, most digital signature algorithms only confirm the authenticity of a hashed digest of the message to be "signed". Verifying the authenticity of a hashed digest of the message is considered proof that the message itself is authentic.

MD5, SHA1, or SHA2 hashes are sometimes posted along with files on websites or forums to allow verification of integrity. This practice establishes a chain of trust so long as the hashes are posted on a site authenticated by HTTPS.

Password verification

A related application is password verification (first invented by Roger Needham). Storing all user passwords as cleartext can result in a massive security breach if the password file is compromised. One way to reduce this danger is to only store the hash digest of each password. To authenticate a user, the password presented by the user is hashed and compared with the stored hash. (Note that this approach prevents the original passwords from being retrieved if forgotten or lost, and they have to be replaced with new ones.) The password is often concatenated with a random, non-secret salt value before the hash function is applied. The salt is stored with the password hash. Because users have different salts, it is not feasible to store tables of precomputed hash values for common passwords. Key stretching functions, such as PBKDF2, Bcrypt or Scrypt, typically use repeated invocations of a cryptographic hash to increase the time required to perform brute force attacks on stored password digests.

Main article: Password cracking

In 2013 a long-term Password Hashing Competition was announced to choose a new, standard algorithm for password hashing.

File or data identifier

A message digest can also serve as a means of reliably identifying a file; several source code management systems, including Git, Mercurial and Monotone, use the sha1sum of various types of content (file content, directory trees, ancestry information, etc.) to uniquely identify them. Hashes are used to identify files on peer-to-peer filesharing networks. For example, in an ed2k link, an MD4-variant hash is combined with the file size, providing sufficient information for locating file sources, downloading the file and verifying its contents. Magnet links are another

example. Such file hashes are often the top hash of a hash list or a hash tree which allows for additional benefits.

One of the main applications of a hash function is to allow the fast look-up of a data in a hash table. Being hash functions of a particular kind, cryptographic hash functions lend themselves well to this application too.

However, compared with standard hash functions, cryptographic hash functions tend to be much more expensive computationally. For this reason, they tend to be used in contexts where it is necessary for users to protect themselves against the possibility of forgery (the creation of data with the same digest as the expected data) by potentially malicious participants.

Pseudorandom generation and key derivation

Hash functions can also be used in the generation of pseudorandom bits, or to derive new keys or passwords from a single, secure key or password.

Hash functions based on block ciphers

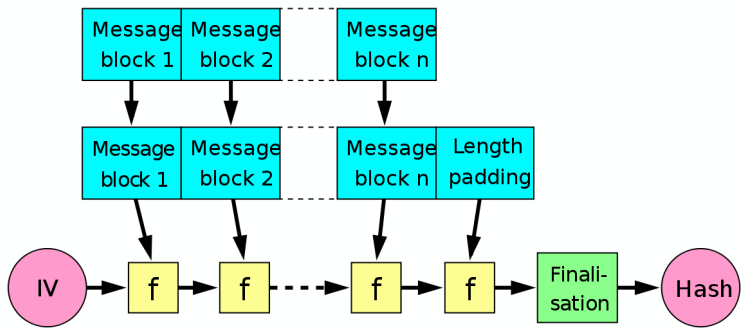
There are several methods to use a block cipher to build a cryptographic hash function, specifically a one-way compression function.

The methods resemble the block cipher modes of operation usually used for encryption. All well-known hash functions, including MD4, MD5, SHA-1 and SHA-2 are built from block-cipher-like components designed for the purpose, with feedback to ensure that the resulting function is not invertible. SHA-3 finalists included functions with block-cipher-like components (e.g., Skein, BLAKE) though the function finally selected, Keccak, was built on a cryptographic sponge instead.

A standard block cipher such as AES can be used in place of these custom block ciphers; that might be useful when an embedded system needs to implement both encryption and hashing with minimal code size or hardware area. However, that approach can have costs in efficiency and security. The ciphers in hash functions are built for hashing: they use large keys and blocks, can efficiently change keys every block, and have been designed and vetted for resistance to related-key attacks. General-purpose ciphers tend to have different design goals. In particular, AES has key and block sizes that make it nontrivial to use to generate long hash values; AES encryption becomes less efficient when the key changes each block; and related-key attacks make it potentially less secure for use in a hash function than for encryption.

Merkle–Damgård construction

Main article: Merkle–Damgård construction



The Merkle–Damgård hash construction.

A hash function must be able to process an arbitrary-length message into a fixed-length output. This can be achieved by breaking the input up into a series of equal-sized blocks, and operating on them in sequence using a one-way compression function. The compression function can either be specially designed for hashing or be built from a block cipher. A hash function built with the Merkle–Damgård construction is as resistant to collisions as is its compression function; any collision for the full hash function can be traced back to a collision in the compression function.

The last block processed should also be unambiguously length padded; this is crucial to the security of this construction. This construction is called the Merkle–Damgård construction. Most widely used hash functions, including SHA-1 and MD5, take this form.

The construction has certain inherent flaws, including length-extension and generate-and-paste attacks, and cannot be parallelized. As a result, many entrants in the current NIST hash function competition are built on different, sometimes novel, constructions.

Use in building other cryptographic primitives

Hash functions can be used to build other cryptographic primitives. For these other primitives to be cryptographically secure, care must be taken to build them correctly.

Message authentication codes (MACs) (also called keyed hash functions) are often built from hash functions. HMAC is such a MAC.

Just as block ciphers can be used to build hash functions, hash functions can be used to build block ciphers. Luby-Rackoff constructions using hash functions can be provably secure if the underlying hash function is secure. Also, many hash functions (including SHA-1 and SHA-2) are built by using a special-purpose block cipher in a Davies-Meyer or other construction. That cipher can also be used in a conventional mode of operation, without the same security guarantees. See SHACAL, BEAR and LION.

Pseudorandom number generators (PRNGs) can be built using hash functions. This is done by combining a (secret) random seed with a counter and hashing it.

Some hash functions, such as Skein, Keccak, and RadioGatún output an arbitrarily long stream and can be used as a stream cipher, and stream ciphers can also be built from fixed-length digest hash functions. Often this is done by first building a cryptographically secure pseudorandom number generator and then using its stream of random bytes as keystream. SEAL is a stream cipher that uses SHA-1 to generate internal tables, which are then used in a keystream generator more or less unrelated to the hash algorithm. SEAL is not guaranteed to be as strong (or weak) as SHA-1. Similarly, the key expansion of the HC-128 and HC-256 stream ciphers makes heavy use of the SHA256 hash function.

Concatenation of cryptographic hash functions

Concatenating outputs from multiple hash functions provides collision resistance as good as the strongest of the algorithms included in the concatenated result. For example, older versions of TLS/SSL use concatenated MD5 and SHA-1 sums—this ensures that a method to find collisions in one of the functions doesn't allow forging traffic protected with both functions.

For Merkle-Damgård hash functions, the concatenated function is as collision-resistant as its strongest component,[1] but not more collision-resistant.[2] Joux[3] noted that 2-collisions lead to n-collisions: if it is feasible to find two messages with the same MD5 hash, it is effectively no more difficult to find as many messages as the attacker desires with identical MD5 hashes. Among the n messages with the same MD5 hash, there is likely to be a collision in SHA-1. The additional work needed to find the SHA-1 collision (beyond the exponential birthday search) is polynomial. This argument is summarized by Finney [4]. A more current paper and full proof of the security of such a combined construction gives a clearer and more complete explanation of the above.

Cryptographic hash algorithms

There is a long list of cryptographic hash functions, although many have been found to be vulnerable and should not be used. Even if a hash function has never been broken, a successful attack against a weakened variant thereof may undermine the experts' confidence and lead to its abandonment. For instance, in August 2004 weaknesses were found in a number of hash functions that were popular at the time, including SHA-0, RIPEMD, and MD5. This has called into question the long-term security of later algorithms which are derived from these hash functions — in particular, SHA-1 (a strengthened version of SHA-0), RIPEMD-128, and RIPEMD-160 (both strengthened versions of RIPEMD). Neither SHA-0 nor RIPEMD are widely used since they were replaced by their strengthened versions.

As of 2009, the two most commonly used cryptographic hash functions are MD5 and SHA-1. However, MD5 has been broken; an attack against it was used to break SSL in 2008.[5]

The SHA-0 and SHA-1 hash functions were developed by the NSA.

On 12 August 2004, a collision for the full SHA-0 algorithm was announced by Joux, Carribault, Lemuet, and Jalby. This was done by using a generalization of the Chabaud and Joux attack. Finding the collision had complexity 2^{51} and took about 80,000 CPU hours on a supercomputer with 256 Itanium 2 processors. (Equivalent to 13 days of full-time use of the computer.)

In February 2005, an attack on SHA-1 was reported that would find collision in about 2^{69} hashing operations, rather than the 2^{80} expected for a 160-bit hash function. In August 2005, another attack on SHA-1 was reported that would find collisions in 2^{63} operations. Though theoretical weaknesses of SHA-1 exist,[6][7] no collision (or near-collision) has yet to be found. Nonetheless, it is often suggested that it may be practical to break within years, and that new applications can avoid these problems by using later members of the SHA family, such as SHA-2, or using techniques such as randomized hashing[8][9] that do not require collision resistance.

However, to ensure the long-term robustness of applications that use hash functions, there was a competition to design a replacement for SHA-2. On October 2, 2012, Keccak was selected as the winner of the NIST hash function competition. A version of this algorithm is expected to become a FIPS standard in 2014 under the name SHA-3.[10]

Some of the following algorithms are used often in cryptography; consult the article for each specific algorithm for more information on the status of each algorithm. Note that this list does not include candidates in the current NIST hash function competition. For additional hash functions see the box at the bottom of the page.

Algorithm	Output size (bits)	Internal state size[11]	Block size	Length size	Word size	Rounds	Best known attacks (complexity:rounds)[12]		
							Collision	Second preimage	Preimage
GOST	256	256	256	256	32	256	Yes (2^{105} [13])	Yes[14] (2^{192} [13])	Yes (2^{192} [13])
HAVAL	256/224/192/160/128	256	1,024	64	32	160/128/96	Yes	No	No
MD2	128	384	128	-	32	864	Yes ($2^{63.3}$ [15])	Yes (2^{73} [16])	Yes (2^{73} [16])
MD4	128	128	512	64	32	48	Yes (3 [17])	Yes (2^{64} [18])	Yes ($2^{78.4}$ [18])
MD5	128	128	512	64	32	64	Yes ($2^{20.96}$ [19])	Yes ($2^{123.4}$ [20])	Yes ($2^{123.4}$ [20])
PANAMA	256	8,736	256	-	32	-	Yes	No	No

RadioGatún	Up to 608/1,216 (19 words)	58 words	3 words	-	1–64	-	With flaws (2^{352} or 2^{704} [21])	No	No
RIPEMD	128	128	512	64	32	48	Yes (2^{18} [15])	No	No
RIPEMD-128/256	128/256	128/256	512	64	32	64	No	No	No
RIPEMD-160	160	160	512	64	32	80	Yes (2^{51}:48 [22])	No	No
RIPEMD-320	320	320	512	64	32	80	No	No	No
RTR0	160	16 words	512	64	64	16	No	No	No
SHA-0	160	160	512	64	32	80	Yes ($2^{33.6}$ [23])	No	No
SHA-1	160	160	512	64	32	80	Theoretical (2^{51} [24])	No	No
SHA-224, SHA-256	256/224	256	512	64	32	64	Theoretical[] ($2^{28.5}$:24 [25])	Theoretical ($2^{248.4}$:42 [18])	Theoretical ($2^{248.4}$:42 [18])
SHA-384, SHA-512, SHA-512/224, SHA-512/256	384/512/224/256	512	1,024	128	64	80	Theoretical ($2^{32.5}$:24 [25])	Theoretical ($2^{494.6}$:42 [18])	Theoretical ($2^{494.6}$:42 [18])
SHA-3	224/256/384/512[26]	1600	1600-2*bits	-	64	24	No	No	No
SHA3-224	224	1600	1152	-	64	24	No	No	No
SHA3-256	256	1600	1088	-	64	24	No	No	No
SHA3-384	384	1600	832	-	64	24	No	No	No
SHA3-512	512	1600	576	-	64	24	No	No	No
Tiger(2)-192/160/128	192/160/128	192	512	64	64	24	Yes (2^{62}:19 [27])	Yes ($2^{184.3}$ [18])	Yes ($2^{184.3}$ [18])
WHIRLPOOL	512	512	512	256	8	10	Yes (2^{120}:4.5 [28])	No	No

Notes

[1] Note that any two messages that collide the concatenated function also collide each component function, by the nature of concatenation. For example, if concat(sha1(message1), md5(message1)) == concat(sha1(message2), md5(message2)) then sha1(message1) == sha1(message2) and md5(message1)==md5(message2). The concatenated function could have other problems that the strongest hash lacks -- for example, it might leak information about the message when the strongest component does not, or it might be detectably nonrandom when the strongest component is not -- but it can't be less collision-resistant.

[2] More generally, if an attack can produce a collision in one hash function's *internal state,* attacking the combined construction is only as difficult as a birthday attack against the other function(s). For the detailed argument, see the Joux and Finney references that follow.

[3] Antoine Joux. *Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions*. LNCS 3152/2004, pages 306-316 Full text (http://www.springerlink.com/index/DWWVMQJU0N0A3UGJ.pdf).

[4] http://article.gmane.org/gmane.comp.encryption.general/5154

[5] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, MD5 considered harmful today: Creating a rogue CA certificate (http://www.win.tue.nl/hashclash/rogue-ca/), accessed March 29, 2009

[6] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, Finding Collisions in the Full SHA-1 (http://people.csail.mit.edu/yiqun/SHA1AttackProceedingVersion.pdf)

[7] Bruce Schneier, Cryptanalysis of SHA-1 (http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html) (summarizes Wang et al. results and their implications)
[8] Shai Halevi, Hugo Krawczyk, Update on Randomized Hashing (http://csrc.nist.gov/groups/ST/hash/documents/HALEVI_UpdateonRandomizedHashing0824.pdf)
[9] Shai Halevi and Hugo Krawczyk, Randomized Hashing and Digital Signatures (http://www.ee.technion.ac.il/~hugo/rhash/)
[10] NIST.gov - Computer Security Division - Computer Security Resource Center (http://csrc.nist.gov/groups/ST/hash/sha-3/timeline_fips.html)
[11] The *internal state* here means the "internal hash sum" after each compression of a data block. Most hash algorithms also internally use some additional variables such as length of the data compressed so far since that is needed for the length padding in the end. See the Merkle-Damgård construction for details.
[12] When omitted, rounds are full number.
[13] http://www.springerlink.com/content/2514122231284103/
[14] There isn't a unique second preimage attack against this hash. However, the second preimage challenge reduces to the ordinary preimage attack by simply constructing a hash of the given message.
[15] http://www.springerlink.com/content/n5vrtdha97a2udkx/
[16] http://eprint.iacr.org/2008/089.pdf
[17] http://www.springerlink.com/content/v6526284mu858v37/
[18] http://eprint.iacr.org/2010/016.pdf
[19] http://eprint.iacr.org/2009/223.pdf
[20] http://springerlink.com/content/d7pm142n58853467/
[21] http://eprint.iacr.org/2008/515
[22] http://www.springerlink.com/content/3540l03h1w31n6w7
[23] http://www.springerlink.com/content/3810jp9730369045/
[24] http://eprint.iacr.org/2008/469.pdf
[25] http://eprint.iacr.org/2008/270.pdf
[26] Although the underlying algorithm Keccak has arbitrary hash lengths, the NIST specified 224, 256, 384 and 512 bits output as valid modes for SHA-3.
[27] http://www.springerlink.com/content/u762587644802p38/
[28] https://www.cosic.esat.kuleuven.be/fse2009/slides/2402_1150_Schlaeffer.pdf

References

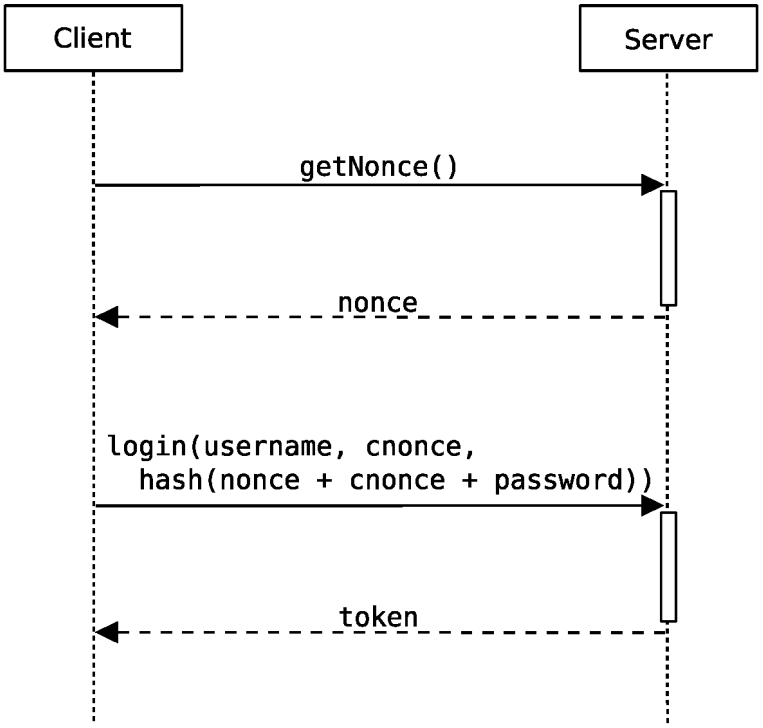
External links

- Paar, Christof; Pelzl, Jan (2009). "11: Hash Functions" (http://wiki.crypto.rub.de/Buch/movies.php). *Understanding Cryptography, A Textbook for Students and Practitioners*. Springer. (companion web site contains online cryptography course that covers hash functions)
- "The ECRYPT Hash Function Website" (http://ehash.iaik.tugraz.at/wiki/The_eHash_Main_Page).
- Buldas, A. (2011). *Series of mini-lectures about cryptographic hash functions* (http://www.guardtime.com/educational-series-on-hashes/).
- Rogaway, P.; Shrimpton, T. (2004). *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*. CiteSeerX: 10.1.1.3.6200 (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.6200).

Cryptographic nonce

For other uses of the word, see Nonce.

In security engineering, a **nonce** is an arbitrary number used only once in a cryptographic communication. It is similar in spirit to a nonce word, hence the name. It is often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in *replay attacks*. For instance, nonces are used in HTTP digest access authentication to calculate an MD5 digest of the password. The nonces are different each time the 401 authentication challenge response code is presented, thus making replay attacks virtually impossible.



Typical client-server communication during a nonce-based authentication process including both a server nonce and a client nonce.

An example is in ordering products over the Internet. An attacker could take the encrypted information and, without needing to decrypt, could continue to send the order into the company, thereby ordering products over and over again under the same name and purchase information. The nonce is used to give 'originality' to the message, where a nonce is used once and only once. If the company receives any other orders from the same person with the same nonce, it will discard those as invalid orders. Many nonces also include a timestamp to ensure exact timeliness, though this requires clock synchronization between organizations. The addition of a client nonce ("**cnonce**") helps to improve the security in some ways as implemented in Digest access authentication.

A nonce may be used to ensure security for a stream cipher. Where the same key is used for more than one message then a different nonce is used to ensure that the keystream is different for different messages encrypted with that key. Often the message number is used.

Some also refer to initialization vectors as nonces for the above reasons. To ensure that a nonce is used only once, it should be time-variant (including a suitably fine-grained timestamp in its value), or generated with enough random bits to ensure a probabilistically insignificant chance of repeating a previously generated value. Some authors define pseudorandomness (or unpredictability) as a requirement for a nonce.[1]

Secret nonce values are used by the Lamport signature scheme as a signer-side secret which can be selectively revealed for comparison to public hashes for signature creation and verification.

Nonces are used in proof of work systems to vary the input to a cryptographic hash function so as to obtain a hash for a certain input that fulfills certain arbitrary conditions. In doing so, it becomes far more difficult to create a "desirable" hash than to verify it, shifting the burden of work onto one side of a transaction or system. For example, proof of work, using hash functions, was considered as a means to combat email spam by forcing email senders to find a hash value for the email (which included a timestamp to prevent pre-computation of useful hashes for later use) that had an arbitrary number of leading zeroes, by hashing the same input with a large number of nonce values

until a "desirable" hash was obtained. Similarly, the bitcoin block-chain hashing algorithm can be tuned to an arbitrary difficulty by changing the required minimum/maximum value of the hash so that the number of bitcoins awarded for new blocks does not increase linearly with increased network computation power as new users join. This is likewise achieved by forcing bitcoin miners to add nonce values to the value being hashed to change the hash algorithm output. Because cryptographic hash algorithms cannot easily be predicted based on their inputs, this makes the act of blockchain hashing and the possibility of being awarded bitcoins something of a lottery, where the first "miner" to find a nonce that delivers a desirable hash is awarded valuable bitcoins.

References

[1] Nonce-Based Symmetric Encryption (http://www.cs.ucdavis.edu/~rogaway/papers/nonce.pdf)

External links

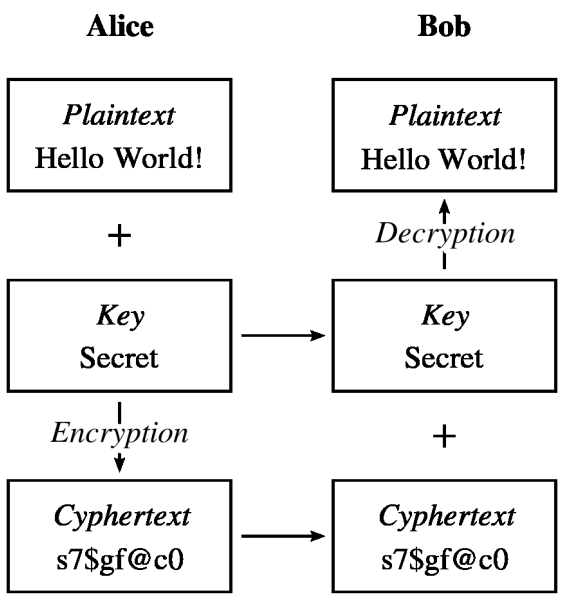
- Sam Ruby Blogging on Nonce with an implementation (http://www.intertwingly.net/blog/1585.html)
- RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication
- RFC 3540 - Robust Explicit Congestion Notification (ECN) Signaling with Nonces
- RFC 4418 - UMAC: Message Authentication Code using Universal Hashing

Cryptography

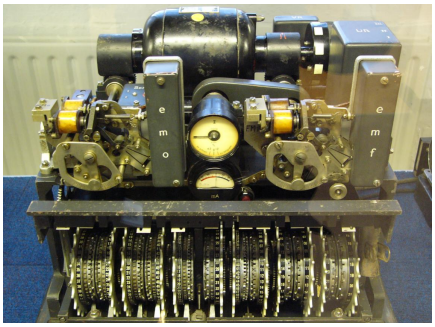
"Secret code" redirects here. For the Aya Kamiki album, see Secret Code.

"Cryptology" redirects here. For the David S. Ware album, see Cryptology (album).

Cryptography (or *cryptology*; from Greek κρυπτός, "hidden, secret"; and γράφειν, *graphein*, "writing", or -λογία, *-logia*, "study", respectively)[1] is the practice and study of techniques for secure communication in the presence of third parties (called adversaries). More generally, it is about constructing and analyzing protocols that overcome the influence of adversaries and which are related to various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.



Symmetric-key cryptography, where the same key is used both for encryption and decryption



German Lorenz cipher machine, used in World War II to encrypt very-high-level general staff messages

Cryptography prior to the modern age was effectively synonymous with *encryption*, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War I and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances, e.g., improvements in integer factorization algorithms, and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that provably cannot be broken even with unlimited computing power—an example is the one-time pad—but these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

Cryptology-related technology has raised a number of legal issues. In the United Kingdom, additions to the Regulation of Investigatory Powers Act 2000 require a suspected criminal to hand over his or her decryption key if asked by law enforcement. Otherwise the user will face a criminal charge. The Electronic Frontier Foundation (EFF) was involved in a case in the United States which questioned whether requiring suspected criminals to provide their decryption keys to law enforcement is unconstitutional. The EFF argued that this is a violation of the right of not being forced to incriminate oneself, as given in the fifth amendment.

Terminology

Until modern times cryptography referred almost exclusively to *encryption*, which is the process of converting ordinary information (called plaintext) into unintelligible text (called ciphertext). Decryption is the reverse, in other words, moving from the unintelligible ciphertext back to plaintext. A *cipher* (or *cypher*) is a pair of algorithms that create the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and in each instance by a "key". This is a secret (ideally known only to the communicants), usually a short string of characters, which is needed to decrypt the ciphertext. A "cryptosystem" is the ordered list of elements of finite possible plaintexts, finite possible cyphertexts, finite possible keys, and the encryption and decryption algorithms which correspond to each key. Keys are important, as ciphers without variable keys can be trivially broken with only the knowledge of the cipher used and are therefore useless (or even counter-productive) for most purposes. Historically, ciphers were often used directly for encryption or decryption without additional procedures such as authentication or integrity checks.

In colloquial use, the term "code" is often used to mean any method of encryption or concealment of meaning. However, in cryptography, *code* has a more specific meaning. It means the replacement of a unit of plaintext (i.e., a meaningful word or phrase) with a code word (for example, `wallaby` replaces `attack at dawn`). Codes are no longer used in serious cryptography—except incidentally for such things as unit designations (e.g., Bronco Flight or Operation Overlord)—since properly chosen ciphers are both more practical and more secure than even the best codes and also are better adapted to computers.

Cryptanalysis is the term used for the study of methods for obtaining the meaning of encrypted information without access to the key normally required to do so; i.e., it is the study of how to crack encryption algorithms or their

implementations.

Some use the terms *cryptography* and *cryptology* interchangeably in English, while others (including US military practice generally) use *cryptography* to refer specifically to the use and practice of cryptographic techniques and *cryptology* to refer to the combined study of cryptography and cryptanalysis.[2] English is more flexible than several other languages in which *cryptology* (done by cryptologists) is always used in the second sense above. In the English Wikipedia the general term used for the entire field is *cryptography* (done by cryptographers). RFC 2828[3] advises that steganography is sometimes included in cryptology.

The study of characteristics of languages which have some application in cryptography[4] (or cryptology), i.e. frequency data, letter combinations, universal patterns, etc., is called cryptolinguistics.

History of cryptography and cryptanalysis

Main article: History of cryptography

Before the modern era, cryptography was concerned solely with message confidentiality (i.e., encryption)—conversion of messages from a comprehensible form into an incomprehensible one and back again at the other end, rendering it unreadable by interceptors or eavesdroppers without secret knowledge (namely the key needed for decryption of that message). Encryption was used to (attempt to) ensure secrecy in communications, such as those of spies, military leaders, and diplomats. In recent decades, the field has expanded beyond confidentiality concerns to include techniques for message integrity checking, sender/receiver identity authentication, digital signatures, interactive proofs and secure computation, among others.

Classic cryptography

Reconstructed ancient Greek *scytale*, an early cipher device

The earliest forms of secret writing required little more than writing implements since most people could not read. More literacy, or literate opponents, required actual cryptography. The main classical cipher types are transposition ciphers, which rearrange the order of letters in a message (e.g., 'hello world' becomes 'ehlol owrdl' in a trivially simple rearrangement scheme), and substitution ciphers, which systematically replace letters or groups of letters with other letters or groups of letters (e.g., 'fly at once' becomes 'gmz bu podf' by replacing each letter with the one following it in the Latin alphabet). Simple versions of either have never offered much confidentiality from enterprising opponents. An early substitution cipher was the Caesar cipher, in which each letter in the plaintext was replaced by a letter some fixed number of positions further down the alphabet. Suetonius reports that Julius Caesar used it with a shift of three to communicate with his generals. Atbash is an example of an early Hebrew cipher. The earliest known use of cryptography is some carved ciphertext on stone in Egypt (ca 1900 BCE), but this may have been done for the amusement of literate observers rather than as a way of concealing information.

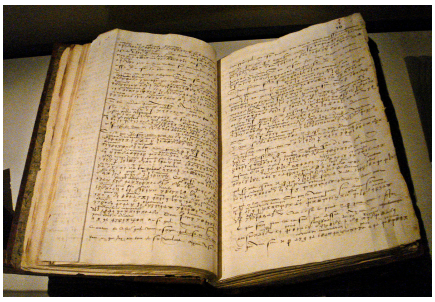
The Greeks of Classical times are said to have known of ciphers (e.g., the scytale transposition cipher claimed to have been used by the Spartan military).[5] Steganography (i.e., hiding even the existence of a message so as to keep it confidential) was also first developed in ancient times. An early example, from Herodotus, concealed a message—a tattoo on a slave's shaved head—under the regrown hair.[] More modern examples of steganography include the use of invisible ink, microdots, and digital watermarks to conceal information.

Ciphertexts produced by a classical cipher (and some modern ciphers) always reveal statistical information about the plaintext, which can often be used to break them. After the discovery of frequency analysis perhaps by the Arab mathematician and polymath Al-Kindi (also known as *Alkindus*) in the 9th century,[6][7] nearly all such ciphers

became more or less readily breakable by any informed attacker. Such classical ciphers still enjoy popularity today, though mostly as puzzles (see cryptogram). Al-Kindi wrote a book on cryptography entitled *Risalah fi Istikhraj al-Mu'amma* (*Manuscript for the Deciphering Cryptographic Messages*), which described the first cryptanalysis techniques.[8][9]

Essentially all ciphers remained vulnerable to cryptanalysis using the frequency analysis technique until the development of the polyalphabetic cipher, most clearly by Leon Battista Alberti around the year 1467, though there is some indication that it was already known to Al-Kindi. Alberti's innovation was to use different ciphers (i.e., substitution alphabets) for various parts of a message (perhaps for each successive plaintext letter at the limit). He also invented what was probably the first automatic cipher device, a wheel which implemented a partial realization of his invention. In the polyalphabetic Vigenère cipher, encryption uses a *key word*, which controls letter substitution depending on which letter of the key word is used. In the mid-19th century Charles Babbage showed that the Vigenère cipher was vulnerable to Kasiski examination, but this was first published about ten years later by Friedrich Kasiski.

16th-century book-shaped French cipher machine, with arms of Henri II of France



Enciphered letter from Gabriel de Luetz d'Aramon, French Ambassador to the Ottoman Empire, after 1546, with partial decipherment

Although frequency analysis is a powerful and general technique against many ciphers, encryption has still often been effective in practice, as many a would-be cryptanalyst was unaware of the technique. Breaking a message without using frequency analysis essentially required knowledge of the cipher used and perhaps of the key involved, thus making espionage, bribery, burglary, defection, etc., more attractive approaches to the cryptanalytically uninformed. It was finally explicitly recognized in the 19th century that secrecy of a cipher's algorithm is not a sensible nor practical safeguard of message security; in fact, it was further realized that any adequate cryptographic scheme (including ciphers) should remain secure even if the adversary fully understands the cipher algorithm itself. Security of the key used should alone be sufficient for a good cipher to maintain confidentiality under an attack. This fundamental principle was first explicitly stated in 1883 by Auguste Kerckhoffs and is generally called Kerckhoffs's Principle; alternatively and more bluntly, it was restated by Claude Shannon, the inventor of information theory and the fundamentals of theoretical cryptography, as *Shannon's Maxim*—'the enemy knows the system'.

Different physical devices and aids have been used to assist with ciphers. One of the earliest may have been the scytale of ancient Greece, a rod supposedly used by the Spartans as an aid for a transposition cipher (see image above). In medieval times, other aids were invented such as the cipher grille, which was also used for a kind of steganography. With the invention of polyalphabetic ciphers came more sophisticated aids such as Alberti's own cipher disk, Johannes Trithemius' tabula recta scheme, and Thomas Jefferson's multi cylinder (not publicly known, and reinvented independently by Bazeries around 1900). Many mechanical encryption/decryption devices were invented early in the 20th century, and several patented, among them rotor machines—famously including the Enigma machine used by the German government and military from the late 1920s and during World War II. The ciphers implemented by better quality examples of these machine designs brought about a substantial increase in cryptanalytic difficulty after WWI.[10]

Computer era

Cryptanalysis of the new mechanical devices proved to be both difficult and laborious. In the United Kingdom, cryptanalytic efforts at Bletchley Park during WWII spurred the development of more efficient means for carrying out repetitious tasks. This culminated in the development of the Colossus, the world's first fully electronic, digital, programmable computer, which assisted in the decryption of ciphers generated by the German Army's Lorenz SZ40/42 machine.

Just as the development of digital computers and electronics helped in cryptanalysis, it made possible much more complex ciphers. Furthermore, computers allowed for the encryption of any kind of data representable in any binary format, unlike classical ciphers which only encrypted written language texts; this was new and significant. Computer use has thus supplanted linguistic cryptography, both for cipher design and cryptanalysis. Many computer ciphers can be characterized by their operation on binary bit sequences (sometimes in groups or blocks), unlike classical and mechanical schemes, which generally manipulate traditional characters (i.e., letters and digits) directly. However, computers have also assisted cryptanalysis, which has compensated to some extent for increased cipher complexity. Nonetheless, good modern ciphers have stayed ahead of cryptanalysis; it is typically the case that use of a quality cipher is very efficient (i.e., fast and requiring few resources, such as memory or CPU capability), while breaking it requires an effort many orders of magnitude larger, and vastly larger than that required for any classical cipher, making cryptanalysis so inefficient and impractical as to be effectively impossible.

Extensive open academic research into cryptography is relatively recent; it began only in the mid-1970s. In recent times, IBM personnel designed the algorithm that became the Federal (i.e., US) Data Encryption Standard; Whitfield Diffie and Martin Hellman published their key agreement algorithm;[11] and the RSA algorithm was published in Martin Gardner's Scientific American column. Since then, cryptography has become a widely used tool in communications, computer networks, and computer security generally. Some modern cryptographic techniques can only keep their keys secret if certain mathematical problems are intractable, such as the integer factorization or the discrete logarithm problems, so there are deep connections with abstract mathematics. There are no absolute proofs that a cryptographic technique is secure (but see one-time pad); at best, there are proofs that some techniques are secure *if* some computational problem is difficult to solve, or certain assumptions about implementation or practical use are met.

As well as being aware of cryptographic history, cryptographic algorithm and system designers must also sensibly consider probable future developments while working on their designs. For instance, continuous improvements in computer processing power have increased the scope of brute-force attacks, thus when specifying key lengths, the required key lengths are similarly advancing. The potential effects of quantum computing are already being considered by some cryptographic system designers; the announced imminence of small implementations of these machines may be making the need for this preemptive caution rather more than merely speculative.[]

Essentially, prior to the early 20th century, cryptography was chiefly concerned with linguistic and lexicographic patterns. Since then the emphasis has shifted, and cryptography now makes extensive use of mathematics, including aspects of information theory, computational complexity, statistics, combinatorics, abstract algebra, number theory, and finite mathematics generally. Cryptography is, also, a branch of engineering, but an unusual one as it deals with active, intelligent, and malevolent opposition (see cryptographic engineering and security engineering); other kinds of engineering (e.g., civil or chemical engineering) need deal only with neutral natural forces. There is also active research examining the relationship between cryptographic problems and quantum physics (see quantum cryptography and quantum computer).

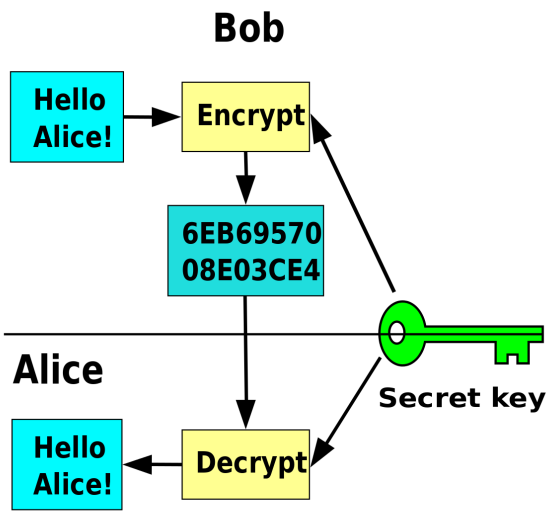
Modern cryptography

The modern field of cryptography can be divided into several areas of study. The chief ones are discussed here; see Topics in Cryptography for more.

Symmetric-key cryptography

Main article: Symmetric-key algorithm

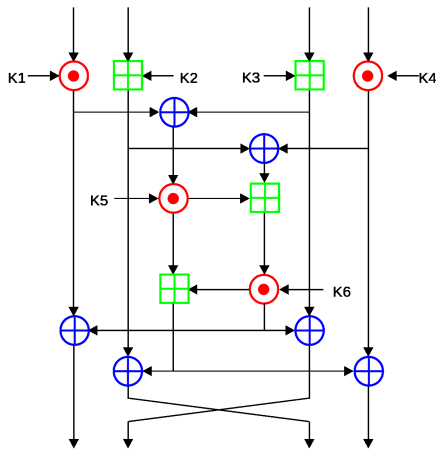
Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976.



Symmetric-key cryptography, where a single key is used for encryption and decryption

Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher.

The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs which have been designated cryptography standards by the US government (though DES's designation was finally withdrawn after the AES was adopted).[12] Despite its deprecation as an official standard, DES (especially its still-approved and much more secure triple-DES variant) remains quite popular; it is used across a wide range of applications, from ATM encryption[13] to e-mail privacy[14] and secure remote access.[15] Many other block ciphers have been designed and released, with considerable variation in quality. Many have been thoroughly broken, such as FEAL.[16]



One round (out of 8.5) of the patented IDEA cipher, used in some versions of PGP for high-speed encryption of, for instance, e-mail

Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state which changes as the cipher operates. That internal state is initially set up using the secret key material. RC4 is a widely used stream cipher; see Category:Stream ciphers. Block ciphers can be used as stream ciphers; see Block cipher modes of operation.

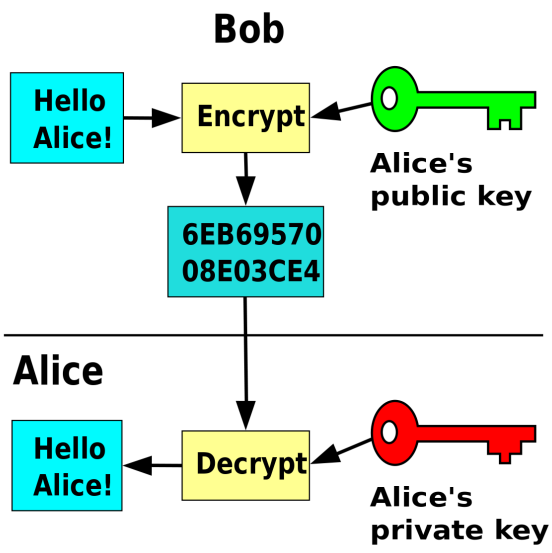
Cryptographic hash functions are a third type of cryptographic algorithm. They take a message of any length as input, and output a short, fixed length hash which can be used in (for example) a digital signature. For good hash functions, an attacker cannot find two messages that produce the same hash. MD4 is a long-used hash function which is now broken; MD5, a strengthened variant of MD4, is also widely used but broken in practice. The U.S. National Security Agency developed the Secure Hash Algorithm series of MD5-like hash functions: SHA-0 was a flawed algorithm that the agency withdrew; SHA-1 is widely deployed and more secure than MD5, but cryptanalysts have identified attacks against it; the SHA-2 family improves on SHA-1, but it isn't yet widely deployed, and the U.S. standards authority thought it "prudent" from a security perspective to develop a new standard to "significantly improve the robustness of NIST's overall hash algorithm toolkit." Thus, a hash function design competition was meant to select a new U.S. national standard, to be called SHA-3, by 2012. The competition ended on October 2, 2012 when the NIST announced that Keccak would be the new SHA-3 hash algorithm.

Message authentication codes (MACs) are much like cryptographic hash functions, except that a secret key can be used to authenticate the hash value upon receipt.

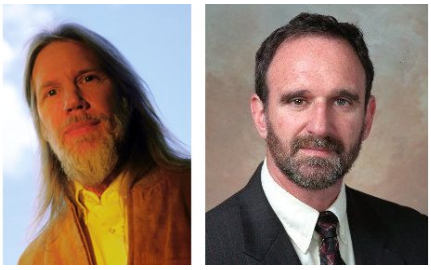
Public-key cryptography

Main article: Public-key cryptography

Symmetric-key cryptosystems use the same key for encryption and decryption of a message, though a message or group of messages may have a different key than others. A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps each ciphertext exchanged as well. The number of keys required increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all consistent and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for cryptography users in the real world.



Public-key cryptography, where different keys are used for encryption and decryption



Whitfield Diffie and Martin Hellman, authors of the first published paper on public-key cryptography

In a groundbreaking 1976 paper, Whitfield Diffie and Martin Hellman proposed the notion of *public-key* (also, more generally, called *asymmetric key*) cryptography in which two different but mathematically related keys are used—a *public* key and a *private* key.[17] A public key system is so constructed that calculation of one key (the 'private key') is computationally infeasible from the other (the 'public key'), even though they are necessarily related. Instead, both keys are generated secretly, as an interrelated pair.[18] The historian David Kahn described public-key cryptography as "the most revolutionary new concept in the field since polyalphabetic substitution emerged in the Renaissance".[19]

In public-key cryptosystems, the public key may be freely distributed, while its paired private key must remain secret. In a public-key encryption system, the *public key* is used for encryption, while the *private* or *secret key* is used for decryption. While Diffie and Hellman could not find such a system, they showed that public-key cryptography was indeed possible by presenting the Diffie–Hellman key exchange protocol, a solution that is now widely used in secure communications to allow two parties to secretly agree on a shared encryption key.

Diffie and Hellman's publication sparked widespread academic efforts in finding a practical public-key encryption system. This race was finally won in 1978 by Ronald Rivest, Adi Shamir, and Len Adleman, whose solution has since become known as the RSA algorithm.[20]

The Diffie–Hellman and RSA algorithms, in addition to being the first publicly known examples of high quality public-key algorithms, have been among the most widely used. Others include the Cramer–Shoup cryptosystem, ElGamal encryption, and various elliptic curve techniques. See Category:Asymmetric-key cryptosystems.

To much surprise, a document published in 1997 by the Government Communications Headquarters (GCHQ), a British intelligence organization, revealed that cryptographers at GCHQ had anticipated several academic developments. Reportedly, around 1970, James H. Ellis had conceived the principles of asymmetric key cryptography. In 1973, Clifford Cocks invented a solution that essentially resembles the RSA algorithm.[21] And in 1974, Malcolm J. Williamson is claimed to have developed the Diffie-Hellman key exchange.

Padlock icon from the Firefox Web browser, meant to indicate a page has been sent in SSL or TLS-encrypted protected form. However, seeing an icon results when code is intended to render it. Malicious code can provide the icon even when the connection is not actually being protected by SSL or TLS. Wikipedia:Citation needed

Public-key cryptography can also be used for implementing digital signature schemes. A digital signature is reminiscent of an ordinary signature; they both have the characteristic of being easy for a user to produce, but difficult for anyone else to forge. Digital signatures can also be permanently tied to the content of the message being signed; they cannot then be 'moved' from one document to another, for any attempt will be detectable. In digital signature schemes, there are two algorithms: one for *signing*, in which a secret key is used to process the message (or a hash of the message, or both), and one for *verification,* in which the matching public key is used with the message to check the validity of the signature. RSA and DSA are two of the most popular digital signature schemes. Digital signatures are central to the operation of public key infrastructures and many network security schemes (e.g., SSL/TLS, many VPNs, etc.).

Public-key algorithms are most often based on the computational complexity of "hard" problems, often from number theory. For example, the hardness of RSA is related to the integer factorization problem, while Diffie–Hellman and DSA are related to the discrete logarithm problem. More recently, *elliptic curve cryptography* has developed in which security is based on number theoretic problems involving elliptic curves. Because of the difficulty of the underlying problems, most public-key algorithms involve operations such as modular multiplication and exponentiation, which are much more computationally expensive than the techniques used in most block ciphers, especially with typical key sizes. As a result, public-key cryptosystems are commonly hybrid cryptosystems, in which a fast high-quality symmetric-key encryption algorithm is used for the message itself, while the relevant symmetric key is sent with the message, but encrypted using a public-key algorithm. Similarly, hybrid signature schemes are often used, in which a cryptographic hash function is computed, and only the resulting hash is digitally signed.

Cryptanalysis

Main article: Cryptanalysis

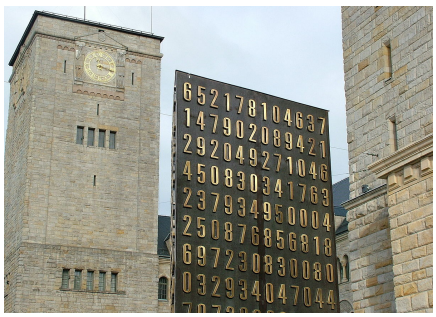
Variants of the Enigma machine, used by Germany's military and civil authorities from the late 1920s through World War II, implemented a complex electro-mechanical polyalphabetic cipher. Breaking and reading of the Enigma cipher at Poland's Cipher Bureau, for 7 years before the war, and subsequent decryption at Bletchley Park, was important to Allied victory.

The goal of cryptanalysis is to find some weakness or insecurity in a cryptographic scheme, thus permitting its subversion or evasion.

It is a common misconception that every encryption method can be broken. In connection with his WWII work at Bell Labs, Claude Shannon proved that the one-time pad cipher is unbreakable, provided the key material is truly random, never reused, kept secret from all possible attackers, and of equal or greater length than the message.[22] Most ciphers, apart from the one-time pad, can be broken with enough computational effort by brute force attack, but the amount of effort needed may be exponentially dependent on the key size, as compared to the effort needed to make use of the cipher. In such cases, effective security could be achieved if it is proven that the effort required (i.e., "work factor", in Shannon's terms) is beyond the ability of any adversary. This means it must be shown that no efficient method (as opposed to the time-consuming brute force method) can be found to break the cipher. Since no such proof has been found to date, the one-time-pad remains the only theoretically unbreakable cipher.

There are a wide variety of cryptanalytic attacks, and they can be classified in any of several ways. A common distinction turns on what an attacker knows and what capabilities are available. In a ciphertext-only attack, the cryptanalyst has access only to the ciphertext (good modern cryptosystems are usually effectively immune to ciphertext-only attacks). In a known-plaintext attack, the cryptanalyst has access to a ciphertext and its corresponding plaintext (or to many such pairs). In a chosen-plaintext attack, the cryptanalyst may choose a plaintext and learn its corresponding ciphertext (perhaps many times); an example is gardening, used by the British during WWII. Finally, in a chosen-ciphertext attack, the cryptanalyst may be able to *choose* ciphertexts and learn their corresponding plaintexts. Also important, often overwhelmingly so, are mistakes (generally in the design or use of one of the protocols involved; see Cryptanalysis of the Enigma for some historical examples of this).

Cryptanalysis of symmetric-key ciphers typically involves looking for attacks against the block ciphers or stream ciphers that are more efficient than any attack that could be against a perfect cipher. For example, a simple brute force attack against DES requires one known plaintext and 2^{55} decryptions, trying approximately half of the possible keys, to reach a point at which chances are better than even that the key sought will have been found. But this may not be enough assurance; a linear cryptanalysis attack against DES requires 2^{43} known plaintexts and approximately 2^{43} DES operations.[23] This is a considerable improvement on brute force attacks.



Poznań monument (*center*) to Polish cryptologists whose breaking of Germany's Enigma machine ciphers, beginning in 1932, altered the course of World War II

Public-key algorithms are based on the computational difficulty of various problems. The most famous of these is integer factorization (e.g., the RSA algorithm is based on a problem related to integer factoring), but the discrete logarithm problem is also important. Much public-key cryptanalysis concerns numerical algorithms for solving these computational problems, or some of them, efficiently (i.e., in a practical time). For instance, the best known algorithms for solving

the elliptic curve-based version of discrete logarithm are much more time-consuming than the best known algorithms for factoring, at least for problems of more or less equivalent size. Thus, other things being equal, to achieve an equivalent strength of attack resistance, factoring-based encryption techniques must use larger keys than elliptic curve techniques. For this reason, public-key cryptosystems based on elliptic curves have become popular since their invention in the mid-1990s.

While pure cryptanalysis uses weaknesses in the algorithms themselves, other attacks on cryptosystems are based on actual use of the algorithms in real devices, and are called *side-channel attacks*. If a cryptanalyst has access to, for example, the amount of time the device took to encrypt a number of plaintexts or report an error in a password or PIN character, he may be able to use a timing attack to break a cipher that is otherwise resistant to analysis. An attacker might also study the pattern and length of messages to derive valuable information; this is known as traffic analysis,[24] and can be quite useful to an alert adversary. Poor administration of a cryptosystem, such as permitting too short keys, will make any system vulnerable, regardless of other virtues. And, of course, social engineering, and other attacks against the personnel who work with cryptosystems or the messages they handle (e.g., bribery, extortion, blackmail, espionage, torture, ...) may be the most productive attacks of all.

Cryptographic primitives

Much of the theoretical work in cryptography concerns cryptographic *primitives*—algorithms with basic cryptographic properties—and their relationship to other cryptographic problems. More complicated cryptographic tools are then built from these basic primitives. These primitives provide fundamental properties, which are used to develop more complex tools called *cryptosystems* or *cryptographic protocols*, which guarantee one or more high-level security properties. Note however, that the distinction between cryptographic *primitives* and cryptosystems, is quite arbitrary; for example, the RSA algorithm is sometimes considered a cryptosystem, and sometimes a primitive. Typical examples of cryptographic primitives include pseudorandom functions, one-way functions, etc.

Cryptosystems

One or more cryptographic primitives are often used to develop a more complex algorithm, called a cryptographic system, or *cryptosystem*. Cryptosystems (e.g. El-Gamal encryption) are designed to provide particular functionality (e.g. public key encryption) while guaranteeing certain security properties (e.g. chosen-plaintext attack (CPA) security in the random oracle model). Cryptosystems use the properties of the underlying cryptographic primitives to support the system's security properties. Of course, as the distinction between primitives and cryptosystems is somewhat arbitrary, a sophisticated cryptosystem can be derived from a combination of several more primitive cryptosystems. In many cases, the cryptosystem's structure involves back and forth communication among two or more parties in space (e.g., between the sender of a secure message and its receiver) or across time (e.g., cryptographically protected backup data). Such cryptosystems are sometimes called *cryptographic protocols*.

Some widely known cryptosystems include RSA encryption, Schnorr signature, El-Gamal encryption, PGP, etc. More complex cryptosystems include electronic cash[25] systems, signcryption systems, etc. Some more 'theoretical' cryptosystems include interactive proof systems,[26] (like zero-knowledge proofs),[27] systems for secret sharing,[28][29] etc.

Until recentlyWP:DATED, most security properties of most cryptosystems were demonstrated using empirical techniques, or using ad hoc reasoning. RecentlyWP:DATED, there has been considerable effort to develop formal techniques for establishing the security of cryptosystems; this has been generally called *provable security*. The general idea of provable security is to give arguments about the computational difficulty needed to compromise some security aspect of the cryptosystem (i.e., to any adversary).

The study of how best to implement and integrate cryptography in software applications is itself a distinct field; see: Cryptographic engineering and Security engineering.

Legal issues

See also: Cryptography laws in different nations

Prohibitions

Cryptography has long been of interest to intelligence gathering and law enforcement agencies. Secret communications may be criminal or even treasonous. Because of its facilitation of privacy, and the diminution of privacy attendant on its prohibition, cryptography is also of considerable interest to civil rights supporters. Accordingly, there has been a history of controversial legal issues surrounding cryptography, especially since the advent of inexpensive computers has made widespread access to high quality cryptography possible.

In some countries, even the domestic use of cryptography is, or has been, restricted. Until 1999, France significantly restricted the use of cryptography domestically, though it has since relaxed many of these rules. In China and Iran, a license is still required to use cryptography. Many countries have tight restrictions on the use of cryptography. Among the more restrictive are laws in Belarus, Kazakhstan, Mongolia, Pakistan, Singapore, Tunisia, and Vietnam.

In the United States, cryptography is legal for domestic use, but there has been much conflict over legal issues related to cryptography. One particularly important issue has been the export of cryptography and cryptographic software and hardware. Probably because of the importance of cryptanalysis in World War II and an expectation that cryptography would continue to be important for national security, many Western governments have, at some point, strictly regulated export of cryptography. After World War II, it was illegal in the US to sell or distribute encryption technology overseas; in fact, encryption was designated as auxiliary military equipment and put on the United States Munitions List.[30] Until the development of the personal computer, asymmetric key algorithms (i.e., public key techniques), and the Internet, this was not especially problematic. However, as the Internet grew and computers became more widely available, high quality encryption techniques became well-known around the globe. As a result, export controls came to be seen to be an impediment to commerce and to research.

Export controls

Main article: Export of cryptography

In the 1990s, there were several challenges to US export regulation of cryptography. After the source code for Philip Zimmermann's Pretty Good Privacy (PGP) encryption program found its way onto the Internet in June 1991, a complaint by RSA Security (then called RSA Data Security, Inc.), resulted in a lengthy criminal investigation of Zimmermann by the U.S. Customs Service and the FBI, though no charges were ever filed.[31] Daniel J. Bernstein, then a graduate student at UC Berkeley, brought a lawsuit against the US government challenging some aspects of the restrictions based on free speech grounds. The 1995 case Bernstein v. United States ultimately resulted in a 1999 decision that printed source code for cryptographic algorithms and systems was protected as free speech by the United States Constitution.[32]

In 1996, thirty-nine countries signed the Wassenaar Arrangement, an arms control treaty that deals with the export of arms and "dual-use" technologies such as cryptography. The treaty stipulated that the use of cryptography with short key-lengths (56-bit for symmetric encryption, 512-bit for RSA) would no longer be export-controlled. Cryptography exports from the US became less strictly regulated as a consequence of a major relaxation in 2000; there are no longer very many restrictions on key sizes in US-exported mass-market software. Since this relaxation in US export restrictions, and because most personal computers connected to the Internet include US-sourced web browsers such as Firefox or Internet Explorer, almost every Internet user worldwide has potential access to quality cryptography via their browsers; e.g., via Transport Layer Security. The Mozilla Thunderbird and Microsoft Outlook E-mail client programs similarly can transmit and received emails via TLS, and can send and receive email encrypted with S/MIME. Many Internet users don't realize that their basic application software contains such extensive cryptosystems. These browsers and email programs are so ubiquitous that even governments whose intent is to

regulate civilian use of cryptography generally don't find it practical to do much to control distribution or use of cryptography of this quality, so even when such laws are in force, actual enforcement is often effectively impossible.Wikipedia:Citation needed

NSA involvement

See also: Clipper chip

Another contentious issue connected to cryptography in the United States is the influence of the National Security Agency on cipher development and policy. The NSA was involved with the design of DES during its development at IBM and its consideration by the National Bureau of Standards as a possible Federal Standard for cryptography.[33] DES was designed to be resistant to differential cryptanalysis, a powerful and general cryptanalytic technique known to the NSA and IBM, that became publicly known only when it was rediscovered in the late 1980s.[34] According to Steven Levy, IBM discovered differential cryptanalysis,[35] but kept the technique secret at the NSA's request. The technique became publicly known only when Biham and Shamir re-discovered and announced it some years later. The entire affair illustrates the difficulty of determining what resources and knowledge an attacker might actually have.

Another instance of the NSA's involvement was the 1993 Clipper chip affair, an encryption microchip intended to be part of the Capstone cryptography-control initiative. Clipper was widely criticized by cryptographers for two reasons. The cipher algorithm (called Skipjack) was then classified (declassified in 1998, long after the Clipper initiative lapsed). The classified cipher caused concerns that the NSA had deliberately made the cipher weak in order to assist its intelligence efforts. The whole initiative was also criticized based on its violation of Kerckhoffs's Principle, as the scheme included a special escrow key held by the government for use by law enforcement, for example in wiretaps.

Digital rights management

Main article: Digital rights management

Cryptography is central to digital rights management (DRM), a group of techniques for technologically controlling use of copyrighted material, being widely implemented and deployed at the behest of some copyright holders. In 1998, American President Bill Clinton signed the Digital Millennium Copyright Act (DMCA), which criminalized all production, dissemination, and use of certain cryptanalytic techniques and technology (now known or later discovered); specifically, those that could be used to circumvent DRM technological schemes. This had a noticeable impact on the cryptography research community since an argument can be made that *any* cryptanalytic research violated, or might violate, the DMCA. Similar statutes have since been enacted in several countries and regions, including the implementation in the EU Copyright Directive. Similar restrictions are called for by treaties signed by World Intellectual Property Organization member-states.

The United States Department of Justice and FBI have not enforced the DMCA as rigorously as had been feared by some, but the law, nonetheless, remains a controversial one. Niels Ferguson, a well-respected cryptography researcher, has publicly stated that he will not release some of his research into an Intel security design for fear of prosecution under the DMCA. Both Alan Cox (longtime number 2 in Linux kernel development) and Edward Felten (and some of his students at Princeton) have encountered problems related to the Act. Dmitry Sklyarov was arrested during a visit to the US from Russia, and jailed for five months pending trial for alleged violations of the DMCA arising from work he had done in Russia, where the work was legal. In 2007, the cryptographic keys responsible for Blu-ray and HD DVD content scrambling were discovered and released onto the Internet. In both cases, the MPAA sent out numerous DMCA takedown notices, and there was a massive internet backlash triggered by the perceived impact of such notices on fair use and free speech.

Forced disclosure of encryption keys

Main article: Key disclosure law

In the United Kingdom, the Regulation of Investigatory Powers Act gives UK police the powers to force suspects to decrypt files or hand over passwords that protect encryption keys. Failure to comply is an offence in its own right, punishable on conviction by a two-year jail sentence or up to five years in cases involving national security. Successful prosecutions have occurred under the Act; the first in 2009, resulting in a term of 13 months' imprisonment.

Similar forced disclosure laws in Australia, Finland, France, and India compel individual suspects under investigation to hand over encryption keys or passwords during a criminal investigation (see Key disclosure law).

In the United States, the federal criminal case of United States v. Fricosu addressed whether a person can be compelled to reveal his or her encryption passphrase or password, despite the U.S. Constitution's Fifth Amendment protection against self-incrimination. In 2012, the court ruled that under the All Writs Act, the defendant was required to produce an unencrypted hard drive.[36]

In many jurisdictions, the legal status of forced disclosure remains unclear.

References

[1] Liddell and Scott's Greek-English Lexicon. Oxford University Press. (1984)

[2] Oded Goldreich, *Foundations of Cryptography, Volume 1: Basic Tools*, Cambridge University Press, 2001, ISBN 0-521-79172-3

[3] https://tools.ietf.org/html/rfc2828

[4] http://staff.neu.edu.tr/~fahri/cryptography.html

[5] V. V. I□A□shchenko (2002). " *Cryptography: an introduction* (http://books.google.com/books?id=cH-NGrpcIMcC&pg=PA6&dq&hl=en#v=onepage&q=&f=false)". AMS Bookstore. p.6. ISBN 0-8218-2986-6

[6] Ibrahim A. Al-Kadi, "The origins of cryptology: The Arab contributions." *Cryptologia*, 16(2) (April 1992) pp. 97–126.

[7] Simon Singh, *The Code Book*, pgs. 14-20. New York: Anchor Books, 2000. ISBN 9780385495325

[8] Simon Singh, *The Code Book*, pp. 14-20

[9] Ibrahim A. Al-Kadi (April 1992), "The origins of cryptology: The Arab contributions", *Cryptologia* **16** (2): 97–126

[10] James Gannon, *Stealing Secrets, Telling Lies: How Spies and Codebreakers Helped Shape the Twentieth Century*, Washington, D.C., Brassey's, 2001, ISBN 1-57488-367-4.

[11] Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644–654. (pdf (http://citeseer.ist.psu.edu/rd/86197922,340126,1,0.25,Download/http://citeseer.ist.psu.edu/cache/papers/cs/16749/http:zSzzSzwww.cs.rutgers.eduzSz~tdnguyenzSzclasseszSzcs671zSzpresentationszSzArvind-NEWDIRS.pdf/diffie76new.pdf))

[12] FIPS PUB 197: The official Advanced Encryption Standard (http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

[13] NCUA letter to credit unions (http://www.ncua.gov/letters/2004/04-CU-09.pdf), July 2004

[14] RFC 2440 - Open PGP Message Format

[15] SSH at windowsecurity.com (http://www.windowsecurity.com/articles/SSH.html) by Pawel Golen, July 2004

[16] Bruce Schneier, *Applied Cryptography*, 2nd edition, Wiley, 1996, ISBN 0-471-11709-9.

[17] Whitfield Diffie and Martin Hellman, "Multi-user cryptographic techniques" [Diffie and Hellman, AFIPS Proceedings 45, pp109–112, June 8, 1976].

[18] Ralph Merkle was working on similar ideas at the time and encountered publication delays, and Hellman has suggested that the term used should be Diffie–Hellman–Merkle aysmmetric key cryptography.

[19] David Kahn, "Cryptology Goes Public", 58 *Foreign Affairs* 141, 151 (fall 1979), p. 153.

[20] R. Rivest, A. Shamir, L. Adleman. . Communications of the ACM, Vol. 21 (2), pp.120–126. 1978. Previously released as an MIT "Technical Memo" in April 1977, and published in Martin Gardner's *Scientific American* Mathematical recreations column

[21] Clifford Cocks. A Note on 'Non-Secret Encryption', CESG Research Report, 20 November 1973 (http://www.fi.muni.cz/usr/matyas/lecture/paper2.pdf).

[22] "Shannon": Claude Shannon and Warren Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, 1963, ISBN 0-252-72548-4

[23] Pascal Junod, "On the Complexity of Matsui's Attack" (http://citeseer.ist.psu.edu/cache/papers/cs/22094/http:zSzzSzeprint.iacr.orgzSz2001zSz056.pdf/junod01complexity.pdf), SAC 2001.

[24] Dawn Song, David Wagner, and Xuqing Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH" (http://citeseer.ist.psu.edu/cache/papers/cs/22094/http:zSzzSzeprint.iacr.orgzSz2001zSz056.pdf/junod01complexity.pdf), In Tenth USENIX Security Symposium, 2001.

[25] S. Brands, "Untraceable Off-line Cash in Wallets with Observers" (http://ftp.se.kde.org/pub/security/docs/ecash/crypto93.ps.gz), In *Advances in Cryptology—Proceedings of CRYPTO*, Springer-Verlag, 1994.
[26] László Babai. "Trading group theory for randomness" (http://portal.acm.org/citation.cfm?id=22192). *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing*, ACM, 1985.
[27] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", SIAM J. Computing, vol. 18, num. 1, pp. 186–208, 1989.
[28] G. Blakley. "Safeguarding cryptographic keys." In *Proceedings of AFIPS 1979*, volume 48, pp. 313–317, June 1979.
[29] A. Shamir. "How to share a secret." In *Communications of the ACM*, volume 22, pp. 612–613, ACM, 1979.
[30] Cryptography & Speech (https://web.archive.org/web/20051201184530/http://www.cyberlaw.com/cylw1095.html) from Cyberlaw
[31] "Case Closed on Zimmermann PGP Investigation" (http://www.ieee-security.org/Cipher/Newsbriefs/1996/960214.zimmerman.html), press note from the IEEE.
[32] Bernstein v USDOJ (http://www.epic.org/crypto/export_controls/bernstein_decision_9_cir.html), 9th Circuit court of appeals decision.
[33] "The Data Encryption Standard (DES)" (http://www.schneier.com/crypto-gram-0006.html#DES) from Bruce Schneier's CryptoGram newsletter, June 15, 2000
[34] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems" (http://www.springerlink.com/index/K54H077NP8714058.pdf), Journal of Cryptology, vol. 4 num. 1, pp. 3–72, Springer-Verlag, 1991.
[35] Levy, pg. 56
[36] http://www.wired.com/images_blogs/threatlevel/2012/01/decrypt.pdf

Further reading

Further information: Books on cryptography

- Becket, B (1988). *Introduction to Cryptology*. Blackwell Scientific Publications. ISBN 0-632-01836-4. OCLC 16832704 (http://www.worldcat.org/oclc/16832704). Excellent coverage of many classical ciphers and cryptography concepts and of the "modern" DES and RSA systems.
- *Cryptography and Mathematics* by Bernhard Esslinger, 200 pages, part of the free open-source package CrypTool, PDF download (https://web.archive.org/web/20110722183013/http://www.cryptool.org/download/CrypToolScript-en.pdf) at the Wayback Machine (archived July 22, 2011). CrypTool is the most widespread e-learning program about cryptography and cryptanalysis, open source.
- *In Code: A Mathematical Journey* by Sarah Flannery (with David Flannery). Popular account of Sarah's award-winning project on public-key cryptography, co-written with her father.
- James Gannon, *Stealing Secrets, Telling Lies: How Spies and Codebreakers Helped Shape the Twentieth Century*, Washington, D.C., Brassey's, 2001, ISBN 1-57488-367-4.
- Oded Goldreich, Foundations of Cryptography (http://www.wisdom.weizmann.ac.il/~oded/foc-book.html), in two volumes, Cambridge University Press, 2001 and 2004.
- *Introduction to Modern Cryptography* (http://www.cs.umd.edu/~jkatz/imc.html) by Jonathan Katz and Yehuda Lindell.
- *Alvin's Secret Code* by Clifford B. Hicks (children's novel that introduces some basic cryptography and cryptanalysis).
- Ibrahim A. Al-Kadi, "The Origins of Cryptology: the Arab Contributions," Cryptologia, vol. 16, no. 2 (April 1992), pp. 97–126.
- Handbook of Applied Cryptography (http://www.cacr.math.uwaterloo.ca/hac/) by A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone CRC Press, (PDF download available), somewhat more mathematical than Schneier's Applied Cryptography.
- Christof Paar (http://www.crypto.rub.de/en_paar.html), Jan Pelzl, Understanding Cryptography, A Textbook for Students and Practitioners. (http://www.cryptography-textbook.com/) Springer, 2009. (Slides, online cryptography lectures and other information are available on the companion web site.) Very accessible introduction to practical cryptography for non-mathematicians.
- *Introduction to Modern Cryptography* by Phillip Rogaway and Mihir Bellare, a mathematical introduction to theoretical cryptography including reduction-based security proofs. PDF download (http://www.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf).

- Johann-Christoph Woltag, 'Coded Communications (Encryption)' in Rüdiger Wolfrum (ed) Max Planck Encyclopedia of Public International Law (Oxford University Press 2009). * "Max Planck Encyclopedia of Public International Law" (http://www.mpepil.com)., giving an overview of international law issues regarding cryptography.
- Jonathan Arbib & John Dwyer, Discrete Mathematics for Cryptography, 1st Edition ISBN 978-1-907934-01-8.
- stallings, william (March 2013), *Cryptography and Network Security: Principles and Practice* (6 ed.), Prentice Hall, ISBN 978-0133354690

External links

- The dictionary definition of cryptography at Wiktionary
- Media related to Cryptography at Wikimedia Commons
- Cryptography (http://www.bbc.co.uk/programmes/p004y272) on *In Our Time* at the BBC. (listen now (http://www.bbc.co.uk/iplayer/console/p004y272/In_Our_Time_Cryptography))
- Crypto Glossary and Dictionary of Technical Cryptography (http://ciphersbyritter.com/GLOSSARY.HTM)
- NSA's CryptoKids (http://www.nsa.gov/kids/).
- Overview and Applications of Cryptology (http://www.cryptool.org/images/ct1/presentations/CrypToolPresentation-en.pdf) by the CrypTool Team; PDF; 3.8 MB—July 2008
- A Course in Cryptography (http://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf) by Raphael Pass & Abhi Shelat - offered at Cornell in the form of lecture notes.
- For more on the use of cryptographic elements in fiction, see: Dooley, John F., William and Marilyn Ingersoll Professor of Computer Science, Knox College (23 August 2012). "Cryptology in Fiction Sorted by Author" (http://www.spiegel.de/international/germany/0,1518,482136,00.html).

Decentralization

Decentralization (or **decentralisation**) is the process of redistributing or dispersing functions, powers, people or things away from a central location or authority.[1][2] While centralization, especially in the governmental sphere, is widely studied and practiced, there is no common definition or understanding of decentralization. The meaning of decentralization may vary in part because of the different ways it is applied.[3] Concepts of decentralization have been applied to group dynamics and management science in private businesses and organizations, political science, law and public administration, economics and technology.

History

Alexis de Tocqueville

The word "centralization" came into use in France in 1794 as the post-French Revolution French Directory leadership created a new government structure. The word "decentralization" came into usage in the 1820s.[4] "Centralization" entered written English in the first third of the 1800s;[5] mentions of decentralization also first appear during those years. In the mid-1800s Alexis de Tocqueville wrote that the French Revolution began with "a push towards decentralization...[but became,]in the end, an extension of centralization."[6] In 1863 retired French bureaucrat Maurice Block wrote an article called “Decentralization” for a French journal which reviewed the dynamics of government and bureaucratic centralization and recent French efforts at decentralization of government functions.[7]

Ideas of liberty and decentralization were carried to their logical conclusions during the 19th and 20th centuries by anti-state political activists calling themselves "anarchists", "libertarians and even decentralists. Alexis de Tocqueville was an advocate, writing: "Decentralization has, not only an administrative value, but also a civic dimension, since it increases the opportunities for citizens to take interest in public affairs; it makes them get accustomed to using freedom. And from the accumulation of these local, active, persnickety freedoms, is born the most efficient counterweight against the claims of the central government, even if it were supported by an impersonal, collective will."[8] Pierre-Joseph Proudhon (1809-1865), influential anarchist theorist wrote: "All my economic ideas as developed over twenty-five years can be summed up in the words: agricultural-industrial federation. All my political ideas boil down to a similar formula: political federation or decentralization."[9]

In early twentieth century America a response to the centralization of economic wealth and political power was a decentralist movement. It blamed large-scale industrial production for destroying middle class shop keepers and small manufacturers and promoted increased property ownership and a return to small scale living. The decentralist movement attracted Southern Agrarians like Robert Penn Warren, as well as journalist Herbert Agar.[10] New Left and libertarian individuals who identified with social, economic, and often political decentralism through the ensuing years included Ralph Borsodi, Wendell Berry, Paul Goodman, Carl Oglesby, Karl Hess, Donald Livingston, Kirkpatrick Sale (author of *Human Scale*), Murray Bookchin,[11] Dorothy Day,[12] Senator Mark O. Hatfield,[13] Mildred J. Loomis[14] and Bill Kauffman.[15]

Leopold Kohr, author of the 1957 book *The Breakdown of Nations*—known for it statement “Whenever something is wrong, something is too big”—was a major influence on E.F. Schumacher, author of the 1973 bestseller *Small is Beautiful:Economics As If People Mattered* .[16][17] In the next few years a number of best-selling books promoted decentralization. Daniel Bell's *The Coming of Post-Industrial Society* discussed the need for decentralization and a “comprehensive overhaul of government structure to find the appropriate size and scoope of units”, as well as the need to detach functions from current state boundaries, creating regions based on functions like water, transport, education and economics which might have “different ‘overlays’ on the map.”[18][19] Alvin Toffler published *Future Shock* (1970) and *The Third Wave* (1980). Discussing the books in a later interview, Toffler said that industrial-style, centralized, top-down bureaucratic planning would be replaced by a more open, democratic, decentralized style which he called “anticipatory democracy.”[20] Futurist John Naisbitt's 1982 book “Megatrends” was on The New York Times Best Seller list for more than two years and sold 14 million copies.[21] Naisbitt’s book outlines 10 “megatrends”, the fifth of which is from centralization to decentralization.[22] In 1996 David Osborne and Ted Gaebler had a best selling book *Reinventing Government* proposing decentralist public administration theories which became labeled the "New Public Management".[23]

Stephen Cummings wrote that decentralization became a "revolutionary megatrend" in the 1980s.[24] In 1983 Diana Conyers asked if decentralization was the "latest fashion" in development administration.[25] Cornell University's project on Restructuring Local Government states that decentralization refers to the "global trend" of devolving responsibilities to regional or local governments.[26] Robert J. Bennett's *Decentralization, Intergovernmental Relations and Markets: Towards a Post-Welfare Agenda* describes how after World War II governments pursued a centralized "welfarist" policy of entitlements which now has become a "post-welfare" policy of intergovernmental and market-based decentralization.

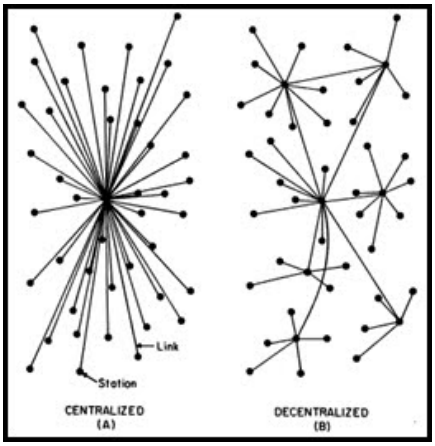
According to a 1999 United Nations Development Programme report:

> "A large number of developing and transitional countries have embarked on some form of decentralization programmes. This trend is coupled with a growing interest in the role of civil society and the private sector as partners to governments in seeking new ways of service delivery...Decentralization of governance and the strengthening of local governing capacity is in part also a function of broader societal trends. These include, for example, the growing distrust of government generally, the spectacular demise of some of the most centralized regimes in the world (especially the Soviet Union) and the emerging separatist demands that seem to routinely pop up in one or another part of the world. The movement toward local accountability and greater control over one's destiny is, however, not solely the result of the negative attitude towards central government. Rather, these developments, as we have already noted, are principally being driven by a strong desire for greater participation of citizens and private sector organizations in governance."[27]

Overview

Systems approach

Those studying the goals and processes of implementing decentralization often use a systems theory approach. The United Nations Development Programme report applies to the topic of decentralization "a whole systems perspective, including levels, spheres, sectors and functions and seeing the community level as the entry point at which holistic definitions of development goals are most likely to emerge from the people themselves and where it is most practical to support them. It involves seeing multi-level frameworks and continuous, synergistic processes of interaction and iteration of cycles as critical for achieving wholeness in a decentralized system and for sustaining its development."[28]



Graphical comparison of centralized and decentralized system.

However, decentralization itself has been seen as part of a systems approach. Norman Johnson of Los Alamos National Laboratory wrote in 1999 paper: "A decentralized system is where some decisions by the agents are made without centralized control or processing. An important property of agent systems is the degree of connectivity or connectedness between the agents, a measure global flow of information or influence. If each agent is connected (exchange states or influence) to all other agents, then the system is highly connected."

University of California, Irvine's Institute for Software Research's "PACE" project is creating an "architectural style for trust management in decentralized applications." It adopted Rohit Khare's definition of decentralization: "A decentralized system is one which requires multiple parties to make their own independent decisions" and applies it to Peer-to-peer software creation, writing:

> ...In such a decentralized system, there is no single centralized authority that makes decisions on behalf of all the parties. Instead each party, also called a peer, makes local autonomous decisions towards its individual goals which may possibly conflict with those of other peers. Peers directly interact with each other and share information or provide service to other peers. An open decentralized system is one in which the entry of peers is not regulated. Any peer can enter or leave the system at any time...[29]

Goals

Decentralization in any area is a response to the problems of centralized systems. Decentralization in government, the topic most studied, has been seen as a solution to problems like economic decline, government inability to fund services and their general decline in performance of overloaded services, the demands of minorities for a greater say in local governance, the general weakening legitimacy of the public sector and global and international pressure on countries with inefficient, undemocratic, overly centralized systems.[30] The following four goals or objectives are frequently stated in various analyses of decentralization.

Participation

In decentralization the principle of subsidiarity often is invoked. It holds that the lowest or least centralized authority which is capable of addressing an issue effectively should do so. According to one definition: "Decentralization, or decentralizing governance, refers to the restructuring or reorganization of authority so that there is a system of co-responsibility between institutions of governance at the central, regional and local levels according to the principle of subsidiarity, thus increasing the overall quality and effectiveness of the system of governance, while increasing the authority and capacities of sub-national levels."[31]

Decentralization is often linked to concepts of participation in decision-making, democracy, equality and liberty from higher authority.[32][33] Decentralization enhances the democratic voice. Theorists believe that local representative authorities with actual discretionary powers are the basis of decentralisation that can lead to local efficiency, equity and development." Columbia University's Earth Institute identified one of three major trends relating to decentralization as: "increased involvement of local jurisdictions and civil society in the management of their affairs, with new forms of participation, consultation, and partnerships."

Decentralization has been described as a "counterpoint to globalization" which removes decisions from the local and national stage to the global sphere of multi-national or non-national interests. Decentralization brings decision-making back to the sub-national levels. Decentralization strategies must the interrelations of the global, regional, national, sub-national, local levels.[34]

Diversity

Norman L. Johnson writes that diversity plays an important role in decentralized systems like ecosystems, social groups, large organizations, political systems. "Diversity is defined to be unique properties of entities, agents, or individuals that are not shared by the larger group, population, structure. Decentralized is defined as a property of a system where the agents have some ability to operate "locally." Both decentralization and diversity are necessary attributes to achieve the self-organizing properties of interest."

Advocates of political decentralization hold that greater participation by better informed diverse interests in society will lead to more relevant decisions than those made only by authorities on the national level.[35] Decentralization has been described as a response to demands for diversity.[36]

Efficiency

In business, decentralization leads to a management by results philosophy which focuses on definite objectives to be achieved by unit results.[37] Decentralization of government programs is said to increase efficiency - and effectiveness - due to reduction of congestion in communications, quicker reaction to unanticipated problems, improved ability to deliver of services, improved information about local conditions, and more support from beneficiaries of programs.[38]

Firms may prefer decentralization because it ensures efficiency by making sure that managers closest to the local information make decisions and in a more timely fashion; that their taking responsibility frees upper management for long term strategizing rather than day-to-day decision-making; that managers have hands on training to prepare them to move up the management hierarchy; that managers are motivated by having the freedom to exercise their own initiative and creativity; that managers and divisions are encouraged to prove that they are profitable, instead of allowing their failures to be masked by the overall profitability of the company.[39]

The same principles can be applied to government. Decentralization promises to enhance efficiency through both inter-governmental competition with market features and fiscal discipline which assigns tax and expenditure authority to the lowest level of government possible. It works best where members of subnational government have strong traditions of democracy, accountability and professionalism.

Conflict resolution

Economic and/or political decentralization can help prevent or reduce conflict because they reduce actual or perceived inequities between various regions or between a region and the central government.[40] Dawn Brancati finds that political decentralization reduces intrastate conflict unless politicians create political parties that mobilize minority and even extremist groups to demand more resources and power within national governments. However, the likelihood this will be done depends on factors like how democratic transitions happen and features like a regional party's proportion of legislative seats, a country's number of regional legislatures, elector procedures, and the order in which national and regional elections occur. Brancati holds that decentralization can promote peace if it encourages statewide parties to incorporate regional demands and limit the power of regional parties.[41]

Processes

The processes of decentralization redefines structures, procedures and practices of governance to be closer to the citizenry and to make them more aware of the costs and benefits; it is not merely a movement of power from the central to the local government. According to the United Nations Development Programme it is "more than a process, it is a way of life and a state of mind." The report provides a chart-formatted framework for defining the application of the concept ‘decentralization’ describing and elaborating on the "who, what, when, where, why and how" factors in any process of decentralization.[42]

Initiation

The processes by which entities move from a more to a less centralized state vary. They can be initiated from the centers of authority ("top-down") or from individuals, localities or regions ("bottom-up"),[43] or from a "mutually desired" combination of authorities and localities working together.[44] Bottom-up decentralization usually stresses political values like local responsiveness and increased participation and tends to increase political stability. Top-down decentralization may be motivated by the desire to “shift deficits downwards” and find more resources to pay for services or pay off government debt. Some hold that decentralization should not be imposed, but done in a respectful manner.[45]

Analysis of operations

Project and program planners must assess the lowest organizational level at which functions can be carried out efficiently and effectively Governments deciding to privatize functions must decide which are best privatized. Existing types of decentralization must be studied. The appropriate balance of centralization and decentralization should be studied. Training for both national and local managers and officials is necessary, as well as technical assistance in the planning, financing, and management of decentralized functions.

Appropriate size

Gauging the appropriate size or scale of decentralized units has been studied in relation to the size of sub-units of hospitals[46] and schools, road networks,[47] administrative units in business[48] and public administration, and especially town and city governmental areas and decision making bodies.[49][50]

In creating planned communities ("new towns"), it is important to determine the appropriate population and geographical size. While in earlier years small towns were considered appropriate, by the 1960s, 60,000 inhabitants was considered the size necessary to support a diversified job market and an adequate shopping center and array of services and entertainment. Appropriate size of governmental units for revenue raising also is a consideration.[51]

Even in bioregionalism, which seeks to reorder many functions and even the boundaries of governments according to physical and environmental features, including watershed boundaries and soil and terrain characteristics, appropriate size must be considered. The unit may be larger than many decentralist bioregionalists prefer.[52]

Inadvertent or silent

Decentralization ideally happens as a careful, rational, and orderly process, but it often takes place during times of economic and political crisis, the fall of a regime and the resultant power struggles. Even when it happens slowly, there is a need for experimentation, testing, adjusting, and replicating successful experiments in other contexts. There is no one blueprint for decentralization since it depends on the initial state of a country and the power and views of political interests and whether they support or oppose decentralization.[53]

Decentralization usually is conscious process based on explicit policies. However, it may occur as "silent decentralization" in the absence of reforms as changes in networks, policy emphasize and resource availability lead inevitably to a more decentralized system.[54] A variation on this is "inadvertent decentralization", when other policy innovations produce an unintended decentralization of power and resources. In both China and Russia, lower level authorities attained greater powers than intended by central authorities.[55]

Asymmetry

Decentralization may be uneven and "asymmetric" given any one country's population, political, ethnic and other forms of diversity. In many countries, political, economic and administrative responsibilities may be decentralized to the larger urban areas, while rural areas are administered by the central government. Decentralization of responsibilities to provinces may be limited only to those provinces or states which want or are capable of handling responsibility. Some privatization may be more appropriate to an urban than a rural area; some types of privatization may be more appropriate for some states and provinces but not others.[56]

Measurement

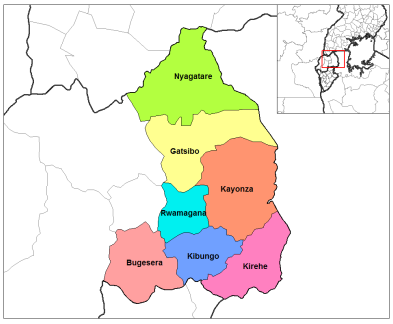
Measuring the amount of decentralization, especially politically, is difficult because different studies of it use different definitions and measurements. An OECD study quotes *Chanchal Kumar Sharma* as stating:[57] "a true assessment of the degree of decentralization in a country can be made only if a comprehensive approach is adopted and rather than trying to simplify the syndrome of characteristics into the single dimension of autonomy, interrelationships of various dimensions of decentralization are taken into account."[58]

Government decentralization

Historians have described the history of governments and empires in terms of centralization and decentralization. In his 1910 *The History of Nations* Henry Cabot Lodge wrote that Persian king Darius I (550-486 BCE) was a master of organization and “for the first time in history centralization becomes a political fact.” He also noted that this contrasted with the decentralization of ancient Ancient Greece.[59] Since the 1980s a number of scholars have written about cycles of centralization and decentralizations. Stephen K. Sanderson wrote that over the last 4000 years chiefdoms and actual states have gone through sequences of centralization and decentralization of economic, political and social power.[60] Yildiz Atasoy writes this process has been going on “since the Stone Age” through not just chiefdoms and states, but empires and today’s “hegemonic core states”.[61] Christopher K. Chase-Dunn and Thomas D. Hall review other works that detail these cycles, including works which analyze the concept of core elites which compete with state accumulation of wealth and how their "intra-ruling-class competition accounts for the rise and fall of states" and of their phases of centralization and decentralization.[62]

Rising government expenditures, poor economic performance and the rise of free market-influenced ideas have convinced governments to decentralize their operations, to induce competition within their services, to contract out to private firms operating in the market, and to privatize some functions and services entirely.[63]

Government decentralization has both political and administrative aspects. Its decentralization may be territorial, moving power from a central city to other localities, and it may be functional, moving decision-making from the top administrator of any branch of government to lower level officials, or divesting of the function entirely through privatization.[64] It has been called the "new public management" which has been described as decentralization, management by objectives, contracting out, competition within government and consumer orientation.[65]



East Province, Rwanda, created in 2006 as part of a government decentralization process.

Political

Political decentralization aims to give citizens or their elected representatives more power. It may be associated with pluralistic politics and representative government, but it also means giving citizens, or their representatives, more influence in the formulation and implementation of laws and policies. Depending on the country, this may require constitutional or statutory reforms, the development of new political parties, increased power for legislatures, the creation of local political units, and encouragement of advocacy groups.

Administrative

Four major forms of administrative decentralization have been described.[][66]

- Deconcentration, the weakest form of decentralization, shifts responsibility for decision-making, finance and implementation of certain public functions from officials of central governments to those in existing districts or, if necessary, new ones under direct control of the central government.
- Delegation passes down responsibility for decision-making, finance and implementation of certain public functions to semi-autonomous organizations not wholly controlled by the central government, but ultimately accountable to it. It involves the creation of public-private enterprises or corporations, or of "authorities", special projects or service districts. All of them will have a great deal of decision-making discretion and they may be exempt from civil service requirements and may be permitted to charge users for services.
- Devolution transfers all responsibility for decision-making, finance and implementation of certain public functions to the sub-national level, such as a regional, local, or state government.
- Divestment, also called privatization, may mean merely contracting out services to private companies. Or it may mean relinquishing totally all responsibility for decision-making, finance and implementation of certain public functions. Facilities will be sold off, workers transferred or fired and private companies or non-for-profit organizations allowed to provide the services.[67] Many of these functions originally were done by private individuals, companies, or associations and later taken over by the government, either directly, or by regulating out of business entities which competed with newly created government programs.[68]

Fiscal

Fiscal decentralization means decentralizing revenue raising and/or expenditure of moneys to a lower level of government while maintaining financial responsibility. While this process usually is called fiscal federalism it may be relevant to unitary, federal and confederal governments. Fiscal federalism also concerns the "vertical imbalances" where the central government gives too much or too little money to the lower levels. It actually can be a way of increasing central government control of lower levels of government, if it is not linked to other kinds of

responsibilities and authority.[69][70][71]

Fiscal decentralization can be achieved through user fees, user participation through monetary or labor contributions, expansion of local property or sales taxes, intergovernmental transfers of central government tax monies to local governments through transfer payments or grants, and authorization of municipal borrowing with national government loan guarantees. Transfers of money may be given conditionally with instructions or unconditionally without them.[72]

Economic or market

Economic decentralization can be done through privatization of public owned functions and businesses, as described briefly above. But it also is done through deregulation, the abolition of restrictions on businesses competing with government services, for example, postal services, schools, garbage collection. Even as private companies and corporations have worked to have such services contracted out to or privatized by them, others have worked to have these turned over to non-profit organizations or associations,

Since the 1970s there has been deregulation of some industries, like banking, trucking, airlines and telecommunications which resulted generally in more competition and lower prices. Wikipedia:Citation needed According to Cato Institute, an American libertarian think-tank, some industries deregulation of aspects of an industry were offset by more ambitious regulations elsewhere that hurt consumers, the electricity industry being a prime example.[73] For example in banking, Cato Institute believes some deregulation allowed banks to compete across state lines, increasing consumer choice, while an actual increase in regulators and regulations forced banks to do business the way central government regulators commanded, including making loans to individuals incapable of repaying them, leading eventually to the financial crisis of 2007–2008.[74] Wikipedia:Identifying reliable sources

Some argue that government standardisation in areas from commodity market, inspection and testing procurement bidding, Building codes, professional and vocational education, trade certification, safety, etc. are necessary.Wikipedia:Citation needed Emmanuelle Auriol and Michel Benaim write about the "comparative benefits" of decentralization versus government regulation in the setting of standards. They find that while there may be a need for public regulation if public safety is at stake, private creation of standards usually is better because "regulators or 'experts' might misrepresent consumers' tastes and needs." As long as companies are averse to incompatible standards, standards will be created that satisfy needs of a modern economy.[75]

Environmental

Central governments themselves may own large tracts of land and control the forest, water, mineral, wildlife and other resources they contain. They may manage them through government operations or leasing them to private businesses; or they may neglect them to be exploited by individuals or groups who defy non-enforced laws against exploitation. It also may control most private land through land-use, zoning, environmental and other regulations.[76] Selling off or leasing lands can be profitable for governments willing to relinquish control, but such programs can face public scrutiny because of fear of a loss of heritage or of environmental damage. Devolution of control to regional or local governments has been found to be an effective way of dealing with these concerns. Such decentralization has happened in India[77] and other third world nations.

Ideological decentralization

Libertarian socialist decentralization

Pierre Joseph Proudhon, anarchist theorist who advocated for a decentralist non-state system which he called "federalism"[78]

Libertarian socialism is a group of political philosophies that promote a non-hierarchical, non-bureaucratic society without private property in the means of production. Libertarian socialists believe in converting present-day private productive property into common or public goods.[79] Libertarian socialism is opposed to coercive forms of social organization. It promotes free association in place of government and opposes the social relations of capitalism, such as wage labor.[80] The term *libertarian socialism* is used by some socialists to differentiate their philosophy from state socialism,[81][82] and by some as a synonym for left anarchism.[83][84][85]

Accordingly, libertarian socialists believe that "the exercise of power in any institutionalized form—whether economic, political, religious, or sexual—brutalizes both the wielder of power and the one over whom it is exercised". Libertarian socialists generally place their hopes in decentralized means of direct democracy such as libertarian municipalism, citizens' assemblies, trade unions, and workers' councils. Libertarian socialists are strongly critical of coercive institutions, which often leads them to reject the legitimacy of the state in favor of anarchism.[86] Adherents propose achieving this through decentralization of political and economic power, usually involving the socialization of most large-scale private property and enterprise (while retaining respect for personal property). Libertarian socialism tends to deny the legitimacy of most forms of economically significant private property, viewing capitalist property relations as forms of domination that are antagonistic to individual freedom.[87][88]

Political philosophies commonly described as libertarian socialist include most varieties of anarchism (especially anarchist communism, anarchist collectivism, anarcho-syndicalism, and mutualism[89]) as well as autonomism, Communalism, participism, libertarian Marxist philosophies such as council communism and Luxemburgism,[90] and some versions of "utopian socialism"[91] and individualist anarchism.[92][93][94] For Murray Bookchin "In the modern world, anarchism first appeared as a movement of the peasantry and yeomanry against declining feudal institutions. In Germany its foremost spokesman during the Peasant Wars was Thomas Muenzer; in England, Gerrard Winstanley, a leading participant in the Digger movement. The concepts held by Muenzer and Winstanley were superbly attuned to the needs of their time — a historical period when the majority of the population lived in the countryside and when the most militant revolutionary forces came from an agrarian world. It would be painfully academic to argue whether Muenzer and Winstanley could have achieved their ideals. What is of real importance is that they spoke to their time; their anarchist concepts followed naturally from the rural society that furnished the bands of the peasant armies in Germany and the New Model in England."[95] The term "anarchist" first entered the English language in 1642, during the English Civil War, as a term of abuse, used by Royalists against their Roundhead opponents. By the time of the French Revolution some, such as the *Enragés*, began to use the term positively,[96] in opposition to Jacobin centralisation of power, seeing "revolutionary government" as oxymoronic. By the turn of the 19th century, the English word "anarchism" had lost its initial negative connotation.

For Proudhon, mutualism involved creating *"industrial democracy,"* a system where workplaces would be *"handed over to democratically organised workers' associations . . . We want these associations to be models for agriculture, industry and trade, the pioneering core of that vast federation of companies and societies woven into the common cloth of the democratic social Republic."*[97] He urged *"workers to form themselves into democratic societies, with*

equal conditions for all members, on pain of a relapse into feudalism." This would result in *"Capitalistic and proprietary exploitation, stopped everywhere, the wage system abolished, equal and just exchange guaranteed."*[98] Workers would no longer sell their labour to a capitalist but rather work for themselves in co-operatives.Anarcho-communism calls for a confederal form in relationships of mutual aid and free association between communes as an alternative to the centralism of the nation-state. Peter Kropotkin thus suggested that "Representative government has accomplished its historical mission; it has given a mortal blow to court-rule; and by its debates it has awakened public interest in public questions. But to see in it the government of the future socialist society is to commit a gross error. Each economic phase of life implies its own political phase; and it is impossible to touch the very basis of the present economic life-private property -without a corresponding change in the very basis of the political organization. Life already shows in which direction the change will be made. Not in increasing the powers of the State, but in resorting to free organization and free federation in all those branches which are now considered as attributes of the State."[99] When the First Spanish Republic was established in 1873 after the abdication of King Amadeo, the first president, Estanislao Figueras, named Francesc Pi i Margall Minister of the Interior. His acquaintance with Proudhon enabled Pi to warm relations between the Republicans and the socialists in Spain. Pi i Margall became the principal translator of Proudhon's works into Spanish[100] and later briefly became president of Spain in 1873 while being the leader of the Democratic Republican Federal Party. According to George Woodcock "These translations were to have a profound and lasting effect on the development of Spanish anarchism after 1870, but before that time Proudhonian ideas, as interpreted by Pi, already provided much of the inspiration for the federalist movement which sprang up in the early 1860's."[101] According to the *Encyclopedia Britannica* "During the Spanish revolution of 1873, Pi y Margall attempted to establish a decentralized, or “cantonalist,” political system on Proudhonian lines."[102]

To date, the best-known examples of an anarchist communist society (i.e., established around the ideas as they exist today and achieving worldwide attention and knowledge in the historical canon), are the anarchist territories during the Spanish Revolution[103] and the Free Territory during the Russian Revolution. Through the efforts and influence of the Spanish Anarchists during the Spanish Revolution within the Spanish Civil War, starting in 1936 anarchist communism existed in most of Aragon, parts of the Levante and Andalusia, as well as in the stronghold of Anarchist Catalonia before being crushed by the combined forces of the regime that won the war, Hitler, Mussolini, Spanish Communist Party repression (backed by the USSR) as well as economic and armaments blockades from the capitalist countries and the Second Spanish Republic itself.[104] During the Russian Revolution, anarchists such as Nestor Makhno worked to create and defend—through the Revolutionary Insurrectionary Army of Ukraine—anarchist communism in the Free Territory of the Ukraine from 1919 before being conquered by the Bolsheviks in 1921. Several libertarian socialists, notably Noam Chomsky among others, believe that anarchism shares much in common with certain variants of Marxism (see libertarian marxism) such as the council communism of Marxist Anton Pannekoek. In Chomsky's *Notes on Anarchism*,[105] he suggests the possibility "that some form of council communism is the natural form of revolutionary socialism in an industrial society. It reflects the belief that democracy is severely limited when the industrial system is controlled by any form of autocratic elite, whether of owners, managers, and technocrats, a 'vanguard' party, or a State bureaucracy."

Free market decentralization

Free market ideas popular in the 19th century, such as those of Adam Smith returned to prominence in the 1970s and 1980s. Nobel Prize winning economist Friedrich von Hayek emphasized that free markets themselves are decentralized systems where outcomes are produced without explicit agreement or coordination by individuals who use prices as their guide.[106] As Eleanor Doyle writes: "Economic decision-making in free markets is decentralized across all the individuals dispersed in each market and is synchronized or coordinated by the price system." The individual right to property is part of this decentralized system.[107] Analyzing the problems of central government control, Hayek wrote in *The Road to Serfdom*:

> There would be no difficulty about efficient control or planning were conditions so simple that a single person or board could effectively survey all the relevant facts. It is only as the factors which have to be taken into account become so numerous that it is impossible to gain a synoptic view of them that decentralization becomes imperative.[108]

According to Bruce M. Owen, this does not mean that all firms themselves have to be equally decentralized. He writes: "markets allocate resources through arms-length transactions among decentralized actors. Much of the time, markets work very efficiently, but there is a variety of conditions under which firms do better. Hence, goods and services are produced and sold by firms with various degrees of horizontal and vertical integration." Additionally, he writes that the "economic incentive to expand horizontally or vertically is usually, but not always, compatible with the social interest in maximizing long-run consumer welfare." When it does not, he writes regulation may be necessary.[109]

It often is claimed that free markets and private property generate centralized monopolies and other ills; the counter is that government is the source of monopoly.[110] Historian Gabriel Kolko in his book *The Triumph of Conservatism* argued that in the first decade of the 20th century businesses were highly decentralized and competitive, with new businesses constantly entering existing industries. There was no trend towards concentration and monopolization. While there were a wave of mergers of companies trying to corner markets, they found there was too much competition to do so. This also was true in banking and finance, which saw decentralization as leading to instability as state and local banks competed with the big New York City firms. The largest firms turned to the power of the state and working with leaders like United States Presidents Theodore Roosevelt, William H. Taft and Woodrow Wilson passed as "progressive reforms" centralizing laws like The Federal Reserve Act of 1913 that gave control of the monetary system to the wealthiest bankers; the formation of monopoly "public utilities" that made competition with those monopolies illegal; federal inspection of meat packers biased against small companies; extending Interstate Commerce Commission to regulating telephone companies and keeping rates high to benefit AT&T; and using the Sherman Anti-trust Act against companies which might combine to threaten larger or monopoly companies.[111][112] When government licensing, franchises, and other legal restrictions create monopoly and protect companies from open competition, deregulation is the solution.[113]

Author and activist Jane Jacobs's influential 1961 book *The Death and Life of American Cities* criticized large-scale redevelopment projects which were part of government-planned decentralization of population and businesses to suburbs. She believed it destroyed cities' economies and impoverished remaining residents.[114] Her 1980 book *The Question of Separatism: Quebec and the Struggle over Sovereignty* supported secession of Quebec from Canada.[115] Her 1984 book *Cities and the Wealth of Nations* proposed a solution to the various ills plaguing cities whose economies were being ruined by centralized national governments: decentralization through the "multiplication of sovereignties", i.e., acceptance of the right of cities to secede from the larger nation states that were squelching their ability to produce wealth.[116][117]

Technological decentralization

The Living Machine installation in the lobby of the Port of Portland headquarters, which was completed and ready for occupation May 2010. The decentralized wastewater reuse system contributed to the headquarter's certification as a LEED Platinum building by the U.S. Green Building Council.

Technology includes tools, materials, skills, techniques and processes by which goals are accomplished in the public and private spheres. Concepts of decentralization of technology are used throughout all types of technology, including especially information technology and appropriate technology.

Technologies often mentioned as best implemented in a decentralized manner, include: water purification, delivery and waste water disposal,[118][119] agricultural technology[120] and energy technology.[121] Advancing technology may allow decentralized, privatized and free market solutions for what have been public services, such utilities producing and/or delivering power, water, mail, telecommunications and services like consumer product safety, money and banking, medical licensing and detection and metering technologies for highways, parking, and auto emissions.[122]

Information technology

Information technology encompasses computers and computer networks, as well as information distribution technologies such as television and telephones. The whole computer industry of computer hardware, software, electronics, internet, telecommunications equipment, e-commerce and computer services are included.

Executives and managers face a constant tension between centralizing and decentralizing information technology for their organizations. They must find the right balance of centralizing which lowers costs and allows more control by upper management, and decentralizing which allows sub-units and users more control. This will depend on analysis of the specific situation. Decentralization is particularly applicable to business or management units which have a high level of independence, complicated products and customers, and technology less relevant to other units.[123]

Information technology applied to government communications with citizens, often called e-Government, is supposed to support decentralization and democratization. Various forms have been instituted in most nations worldwide.[124]

The internet is an example of an extremely decentralized network, having no owners at all (although some have argued that this is less the case in recent years). "No one is in charge of internet, and everyone is." As long as they follow a certain minimal number of rules, anyone can be a service provider or a user. Voluntary boards establish protocols, but cannot stop anyone from developing new ones.[125] Other examples of open source or decentralized movements are Wikis which allow users to add, modify, or delete content via the internet.[126] Wikipedia itself has been described as decentralized.[127] Smartphones have greatly increased the role of decentralized social network services in daily lives worldwide.[128]

Decentralization continues throughout the industry, for example as the decentralized architecture of wireless routers installed in homes and offices supplement and even replace phone companies relatively centralized long-range cell towers.[129]

Inspired by system and cybernetics theorists like Norbert Wiener, Marshall MacLuhan and Buckminster Fuller, in the 1960s Stewart Brand started the Whole Earth Catalog and later computer networking efforts to bring Silicon Valley computer technologists and entrepreneurs together with countercultural ideas. This resulted in ideas like personal computing, virtual communities and the vision of an "electronic frontier" which would be a more decentralized, egalitarian and free-market libertarian society. Related ideas coming out of Silicon Valley included the free software and creative commons movements which produced visions of a "networked information

economy".[130]

Because human interactions in cyberspace transcend physical geography, there is a necessity for new theories in legal and other rule-making systems to deal with decentralized decision-making processes in such systems. For example, what rules should apply to conduct on the global digital network and who should set them? The laws of which nations govern issues of internet transactions (like seller disclosure requirements or definitions of "fraud"), copyright and trademark?[131]

Centralization and redecentralization of the Internet

The New Yorker reports that although the Internet was originally decentralized, in recent years it has become less so: "a staggering percentage of communications flow through a small set of corporations—and thus, under the profound influence of those companies and other institutions [...] One solution, espoused by some programmers, is to make the Internet more like it used to be—less centralized and more distributed."

Examples of projects that attempt to contribute to the redecentralization of the Internet include ArkOS, Diaspora, FreedomBox and Namecoin, as well as advocacy group Redecentralize.org, which provides support for projects that aim to make the Web less centralized.

In an interview with BBC Radio 5 Live one of the co-founders of Redecentralize.org explained that:

> "As we've gone on there's been more and more internet traffic focused through particular nodes such as Google or Facebook. [...] Centralised services that hold all the user data and host it themselves have become increasingly popular because that business model has worked. As the Internet has become more mass market, people are not necessarily willing or knowledgable to host it themselves, so where that hosting is outsourced it's become the default, which allows a centralization of power and a centralization of data that I think is worrying."

Appropriate technology

"Appropriate technology", originally described as "intermediate technology" by economist E. F. Schumacher in *Small is Beautiful*, is generally recognized as encompassing technologies that are small-scale, decentralized, labor-intensive, energy-efficient, environmentally sound, and locally controlled. It is most commonly discussed as an alternative to transfers of capital-intensive technology from industrialized nations to developing countries. Even developed countries developed appropriate technologies, as did the United States in 1977 when it created the National Center for Appropriate Technology (NCAT), though funding later dropped off.[132] A related concept is "design for the other 90 percent" - low-cost solutions for the great majority of the world's low income people.

Critiques

Factors hindering decentralization include weak local administrative or technical capacity may result in inefficient or ineffective services; inadequate financial resources may be made available to perform new local responsibilities, especially in the start-up phase when they are most needed; inequitable distribution of resources may result; decentralization can make national policy coordination too complex; it may allow local elites to capture functions; local cooperation maybe undermined by any distrust between private and public sectors; decentralization may result in higher enforcement costs and conflict for resources if there is no higher level of authority.[133] Additionally, decentralization may not be as efficient for standardized, routine, network-based services, as opposed to those that need more complicated inputs. If there is a loss of economies of scale in procurement of labor or resources, the expense of decentralization can rise, even as central governments lose control over financial resources.

Other challenges, and even dangers, include the possibility that corrupt local elites can capture regional or local power centers, while constituents lose representation; patronage politics will become rampant and civil servants feel compromised; further necessary decentralization can be stymied; incomplete information and hidden

decision-making can occur up and down the hierarchies; centralized power centers can find reasons to frustrate decentralization and bring power back to themselves.Wikipedia:Citation needed

It has been noted that while decentralization may increase "productive efficiency" it may undermine "allocative efficiency" by making redistribution of wealth more difficult. Decentralization will cause greater disparities between rich and poor regions, especially during times of crisis when the national government may not be able to help regions needing it.[134]

References

[1] Definition of decentralization (http://www.thefreedictionary.com/decentralization), TheFreeDictionary.com, *accessed February 4, 2013.*

[2] Definition of decentralization (http://www.merriam-webster.com/dictionary/decentralization), Merriam-Webster Dictionary, *accessed February 4, 2013.*

[3] Decentralization: A Sampling of Definitions (http://web.undp.org/evaluation/documents/decentralization_working_report.PDF), Joint UNDP (United Nations Development Programme)-Government of Germany evaluation of the UNDP role in decentralization and local governance, at the United Nations Development Programme website, October 1999, p. 1, 3-6. Quote: "In fact, a quick review of the literature shows that there is no common definition or understanding of decentralization, although much work has gone into exploring its differing applications."

[4] Vivien A. Schmidt, *Democratizing France: The Political and Administrative History of Decentralization*, Cambridge University Press, 2007, p. 22 (http://books.google.com/books?id=ZsltI4XKXTUC&pg=PA22&dq=word+centralization&hl=en&sa=X&ei=58sSUc_MFfKB0QHC6oFQ&ved=0CFoQ6AEwBw#v=onepage&q=word centralization&f=false), ISBN 9780521036054

[5] Barbara Levick, *Claudius*, Psychology Press, 2012, p. 81 (http://books.google.com/books?id=rTDNNO4_IMAC&pg=PA81&dq=word+centralization+english&hl=en&sa=X&ei=WMwSUe2VKuuA0AHik4CoCA&ved=0CDAQ6AEwAA#v=onepage&q=word centralization english&f=false), ISBN 9780415166195

[6] Vivien A. Schmidt, *Democratizing France: The Political and Administrative History of Decentralization*, p. 10 (http://books.google.com/books?id=ZsltI4XKXTUC&pg=PA10&dq=de+tocqueville+Decentralization&hl=en&sa=X&ei=f6ESUbyGIaqB0QGztoGoCA&ved=0CEUQ6AEwAw#v=onepage&q=de tocqueville Decentralization&f=false).

[7] Robert Leroux, *French Liberalism in the 19th Century: An Anthology*, Chapter 6: Maurice Block on "Decentralization", Routledge, 2012, p. 255 (http://books.google.com/books?id=Hhf1iGshBKEC&pg=PA255&dq=19th+century+decentralization&hl=en&sa=X&ei=csUSUaWxJYHj0gHnuoCoBQ&ved=0CD0Q6AEwAg#v=onepage&q=19th century decentralization&f=false), ISBN 9781136313011

[8] A History of Decentralization (http://www.ciesin.org/decentralization/English/General/history_fao.html), Earth Institute of Columbia University website, *accessed February 4, 2013.*

[9] "Du principe Fédératif" ("Principle of Federation"), 1863.

[10] Craig R. Prentiss, *Debating God's Economy: Social Justice in America on the Eve of Vatican II,* Penn State Press, 2008, p. 43 (http://books.google.com/books?id=wWY-DhOnFXkC&pg=PA42&dq=early+advocate+of+decentralism&hl=en&sa=X&ei=0B0TUYi_OOqs0AG044GIDg&ved=0CDAQ6AEwAA#v=onepage&q=early advocate of decentralism&f=false), ISBN 9780271033419

[11] David De Leon, *Leaders from the 1960s: A Biographical Sourcebook of American Activism*, Greenwood Publishing Group, 1994, p. 297 (http://books.google.com/books?hl=en&lr=&id=oHYD-XUiSBEC&oi=fnd&pg=PA297&dq=libertarian+decentralism&ots=d5GNBa93mV&sig=v6uJPO8s5xhyUzL63gTKduGlcUY#v=onepage&q&f=false), ISBN 9780313274145

[12] Nancy L. Roberts, *Dorothy Day and the Catholic worker,* Volume 84, Issue 1 of National security essay series, State University of New York Press, 1984, p. 11 (http://books.google.com/books?id=Wx5A4UE05QYC&pg=PA11&dq=libertarian+decentralist&hl=en&ei=7hyyTJ73CML68AaK97GdCQ&sa=X&oi=book_result&ct=result&resnum=3&ved=0CDAQ6AEwAjge#v=onepage&q=libertarian decentralist&f=false), ISBN 9780873959391

[13] Jesse Walker, Mark O. Hatfield, RIP (http://reason.com/blog/2011/08/08/mark-o-hatfield-rip), Reason, August 8, 2011.

[14] Mildred J. Loomis, *Decentralism: Where It Came From - Where Is It Going?*, Black Rose Books, 2005, ISBN 9781551642499

[15] Bill Kauffman, *Bye Bye, Miss American Empire: Neighborhood Patriots, Backcountry rebels*, Chelsea Green Publishing, 2010, p xxxi (http://books.google.com/books?id=b98t6YKdmCYC&pg=PR31&dq=Bill+Kauffman+decentralist+libertarian&hl=en&sa=X&ei=7EQTUdjqJ9CM0QGp4IHADA&ved=0CDwQ6AEwAg#v=onepage&q=Bill Kauffman decentralist libertarian&f=false), ISBN 9781933392806

[16] Dr. Leopold Kohr, 84; Backed Smaller States (http://www.nytimes.com/1994/02/28/obituaries/dr-leopold-kohr-84-backed-smaller-states.html), New York Times obituary, February 28, 1994.

[17] John Fullerton, The Relevance of E. F. Schumacher in the 21st Century (http://neweconomicsinstitute.org/publications/essays/fullerton/john/the-relevance-of-ef-schumacher-in-the-21st-century), New Economics Institute, *accessed February 7, 2013.*

[18] W. Patrick McCray, *The Visioneers: How a Group of Elite Scientists Pursued Space Colonies, Nanotechnlogies, and a Limitless Future*, Princeton University Press, 2012, p. 70 (http://books.google.com/books?id=i-Xh0i4-mnYC&pg=PA70&dq=The+Visioneers+Daniel+bell&hl=en&sa=X&ei=2aEbUZXmK8f20gHQ54HgBA&ved=0CDAQ6AEwAA#v=onepage&q=The Visioneers Daniel bell&f=false), ISBN 9780691139838

[19] Daniel Bell, *The Coming Of Post-industrial Society*, Basic Books version, 2008, p. 320-321 (http://books.google.com/books?id=q6_56x5tB7gC&pg=PA319&dq="appropriate+size"+decentralization&hl=en&sa=X&ei=TCEbUZKAJo6k8ASEzoDoDQ&ved=0CH4Q6AEwDg#v=onepage&q="appropriate size" decentralization&f=false), ISBN 9780786724734

[20] Alvin Toffler, *Previews & Premises: An Interview with the Author of Future Shock and The Third Wave,* Black Rose books, 1987, p. 50 (http://books.google.com/books?id=gjbGnzqrulMC&pg=PA49&dq="Future+Shock"+decentralization&hl=en&sa=X&ei=IQoUUdTMN8aQ0QGdvIDYCg&ved=0CDYQ6AEwAQ#v=onepage&q="Future Shock" decentralization&f=false), ISBN 9780920057377

[21] John Naisbitt biography (http://www.naisbitt.com/index.php?id=5) at personal website, *accessed February 10, 2013*.

[22] Sam Inkinen, *Mediapolis: Aspects of Texts, Hypertexts and Multimedial Communication,* Volume 25 of Research in Text Theory, Walter de Gruyter, 1999, p. 272 (http://books.google.com/books?id=UdOg403f5ykC&pg=PA272&dq=Naisbitt+Megatrends+decentralization&hl=en&sa=X&ei=JRcUUfDQDO630QHN2YHoDg&ved=0CGsQ6AEwCw#v=onepage&q=Naisbitt Megatrends decentralization&f=false), ISBN 9783110807059

[23] Public Administration Review, Vol. 56, No. 3 (May - June, 1996), pp. 247-255.

[24] Stephen Cummings, *ReCreating Strategy*, SAGE, 2002, p. 157 (http://books.google.com/books?id=ZymtYIEUZj0C&pg=PA157&dq=Stephen+Cummings+decentralization+Megatrend&hl=en&sa=X&ei=nBMUUfT3BOT00QGg-IGIDA&ved=0CDAQ6AEwAA#v=onepage&q=Stephen Cummings decentralization Megatrend&f=false), ISBN 9780857026514

[25] Diana Conyers, "Decentralization: The latest fashion in development administration?" (http://onlinelibrary.wiley.com/doi/10.1002/pad.4230030202/abstract), Public Administration and Development, Volume 3, Issue 2, pages 97–109, April/June 1983, via Wiley Online Library, *accessed February 4, 2013*.

[26] Decentralization (http://government.cce.cornell.edu/doc/viewpage_r.asp?ID=Decentralization), article at the " Restructuring local government project (http://government.cce.cornell.edu/)" of Dr. Mildred Warner, Cornell University, *accessed February 4, 2013*.

[27] "Decentralization: A Sampling of Definitions", 1999, p. 11-12.

[28] "Decentralization: A Sampling of Definitions",1999, p. 13.

[29] PACE Project "What is Decentralization?" page (http://www.isr.uci.edu/projects/pace/decentralization.html), University of California, Irvine's Institute for Software Research, Last Updated - May 10, 2006.

[30] Holger Daun, *School Decentralization in the Context of Globalizing Governance: International Comparison of Grassroots Responses,* Springer, 2007, p. 28-29 (http://books.google.com/books?id=nYHXnyzGggkC&pg=PA28&dq=goals+of+decentralization&hl=en&sa=X&ei=fq0aUYHIEZC30QHyy4G4CQ&ved=0CEsQ6AEwBQ#v=onepage&q=goals of decentralization&f=false), ISBN 9781402047008

[31] "Decentralization: A Sampling of Definitions", 1999, p. 2, 16, 26.

[32] Subhabrata Dutta, *Democratic decentralization and grassroot leadership in India* (http://books.google.com/books?id=O8X_NxDWDq4C&printsec=frontcover&dq=decentralization+Dutta&hl=en&sa=X&ei=SaoZUdHRB4iw0AHVv4Fo&ved=0CDAQ6AEwAA), Mittal Publications, 2009, p. 5-8, ISBN 9788183242738

[33] Robert Charles Vipond, *Liberty & Community: Canadian Federalism and the Failure of the Constitution* (http://books.google.com/books?id=_iTJdiemMoUC&pg=PA252&dq=Vipond+Liberty+and+Community&hl=en&sa=X&ei=rasZUZGEDJK40QGqwYGIBw&ved=0CDAQ6AEwAA), SUNY Press, 1991, p. 145, ISBN 9780791404669

[34] "Decentralization: A Sampling of Definitions", 1999, p. 12-13.

[35] Political Decentralization (http://www1.worldbank.org/publicsector/decentralization/political.htm#2), Decentralization and Subnational Economies project, World Bank website, *accessed February 9, 2013*.

[36] Therese A McCarty, Demographic diversity and the size of the public sector (http://onlinelibrary.wiley.com/doi/10.1111/j.1467-6435.1993.tb02422.x/abstract), Kyklos, 1993, via Wiley Online Library. Quote: "...if demographic diversity promotes greater decentralization, the size of the public sector is not affected 10 consequently."

[37] "Decentralization: A Sampling of Definitions", 1999, p. 11.

[38] Jerry M. Silverman, *Public Sector Decentralization: Economic Policy and Sector Investment Programs, Volume 188*, World Bank Publications, 1992, p. 4 (http://books.google.com/books?id=XtoXuSHMclIC&pg=PA4&dq=efficiency+decentralization&hl=en&sa=X&ei=Q90ZUZC3F4rV0gG86IGIBw&ved=0CEUQ6AEwBA#v=onepage&q=efficiency decentralization&f=false), ISBN 9780821322796

[39] Don R. Hansen, Maryanne M. Mowen, Liming Guan, *Cost Management: Accounting & Control*, Cengage Learning, 2009, p. 338 (http://books.google.com/books?id=10eIDgRH3EgC&pg=PA338&dq=reason+for++decentralization&hl=en&sa=X&ei=pRgbUb7nOJOC9gSbmYCABg&ved=0CDAQ6AEwAA#v=onepage&q=reason for decentralization&f=false), ISBN 9780324559675

[40] David R. Cameron, Gustav Ranis, Annalisa Zinn, *Globalization and Self-Determination: Is the Nation-State Under Siege?*, Taylor & Francis, 2006, p. 203 (http://books.google.com/books?id=bO-FSUEPO6MC&pg=PA202&dq="conflict+resolution"+decentralization&hl=en&sa=X&ei=uDMbUZuvNIua9QTH-oGQBA&ved=0CH0Q6AEwDA#v=onepage&q="conflict resolution" decentralization&f=false), ISBN 9780203086636

[41] Dawn Brancati, Peace by Design:Managing Intrastate Conflict through Decentralization (http://books.google.com/books/about/Peace_by_Design_Managing_Intrastate_Conf.html?id=g7eLksrA8LAC), Oxford University Press, 2009, ISBN 9780191615221

[42] "Decentralization: A Sampling of Definitions", 1999, p. 12, chart p. 27.

[43] Ariunaa Lkhagvadorj, *Fiscal federalism and decentralization in Mongolia*, University of Potsdam, Germany, 2010, p. 23 (http://books.google.com/books?id=gDK4fsCj7ZYC&pg=PA23&dq=top+down+bottom+up+decentralization&hl=en&sa=X&

ei=MVUZUZCkE9C70AGn44HwBA&ved=0CEcQ6AEwAw#v=onepage&q=top down bottom up decentralization&f=false), ISBN 9783869560533

[44] Karin E. Kemper, Ariel Dinar, *Integrated River Basin Management Through Decentralization*, Springer, 2007, p. 36 (http://books.google.com/books?id=k0BMK_M9gN4C&pg=PA36&dq=top+down+bottom+up+decentralization&hl=en&sa=X&ei=MVUZUZCkE9C70AGn44HwBA&ved=0CEwQ6AEwBA#v=onepage&q=top down bottom up decentralization&f=false), ISBN 9783540283553.

[45] "Decentralization: A Sampling of Definitions", 1999, p. 12.

[46] Robert J. Taylor, Susan B. Taylor, *The Aupha Manual of Health Services Management*, Jones & Bartlett Learning, 1994, p. 33, ISBN 9780834203631

[47] Frannie Frank Humplick, Azadeh Moini Araghi, "Is There an Optimal Structure for Decentralized Provision of Roads?", World Bank Policy Research Working Paper, 1996, p. 35.

[48] Abbass F. Alkhafaji, *Strategic Management: Formulation, Implementation, and Control in a Dynamic Environment*, Psychology Press, 2003, p. 184, ISBN 9780789018106

[49] Ehtisham Ahmad, Vito Tanzi, *Managing Fiscal Decentralization*, Routledge, 2003, p. 182 (http://books.google.com/books?id=qzcVmaqpitwC&pg=PA182&dq="appropriate+size"+government+decentralization&hl=en&sa=X&ei=Gr0bUerHCofB0QG6v4Ag&ved=0CE0Q6AEwBQ#v=onepage&q="appropriate size" government decentralization&f=false), ISBN 9780203219997

[50] Aaron Tesfaye, *Political Power and Ethnic Federalism: The Struggle for Democracy in Ethiopa*, University Press of America, 2002, p. 44 (http://books.google.com/books?id=XD9oFjvFurAC&pg=PA44&dq="appropriate+size"+government+decentralization&hl=en&sa=X&ei=Gr0bUerHCofB0QG6v4Ag&ved=0CEcQ6AEwBA#v=onepage&q="appropriate size" government decentralization&f=false), ISBN 9780761822394

[51] Harry Ward Richardson, *Urban economics*, Dryden Press, 1978, p. 107, 133, 159 (http://books.google.com/books?ei=Gr0bUerHCofB0QG6v4Ag&id=d9YpAQAAMAAJ&dq="appropriate+size"+government+decentralization&q="appropriate+size"+#search_anchor).

[52] Allen G Noble, Frank J. Costa, *Preserving the Legacy: Concepts in Support of Sustainability*, Lexington Books, 1999, p. 214 (http://books.google.com/books?id=QVDd6tQLZ9kC&pg=PA214&dq=Bioregionalism+appropriate+size&hl=en&sa=X&ei=PMIbUfeZJvSx0AH2rIDQDA&ved=0CDwQ6AEwAg#v=onepage&q=Bioregionalism appropriate size&f=false), ISBN 9780739100158

[53] "Decentralization: A Sampling of Definitions", 1999, p. 21.

[54] H.F.W. Dubois and G. Fattore, Definitions and typologies in public administration research: the case of decentralization (http://www.tandfonline.com/doi/abs/10.1080/01900690902908760), *International Journal of Public Administration*, Volume 32, Issue 8, 2009, pp. 704–727.

[55] "Decentralization: A Sampling of Definitions", 1999, p. 19–20.

[56] "Decentralization: A Sampling of Definitions", 1999, p. 19.

[57] OECD, 2013, Fiscal Federalism Studies Measuring Fiscal Decentralisation: Concepts and Policies (http://books.google.co.in/books?id=4__JMipPEPoC&pg=PA96&lpg=PA96), OECD, Korea Institute of Public Finance

[58] Chanchal Kumar Sharma, Decentralization Dilemma: Measuring the Degree and Evaluating the Outcomes (http://papers.ssrn.com/sol3/papers.cfm?abstract_id=955113), *The Indian Journal of Political Science*, Vol. 67, No. 1, 2006, pp. 49–64.

[59] Henry Cabot Lodge, Volume 1 of *The History of Nations,* H. W. Snow, 1910, p. 164 (http://books.google.com/books?id=GvMnAQAAMAAJ&pg=PA164-IA2&dq=history+of+empires+centralization&hl=en&sa=X&ei=6soSUfniMs6E0QHm94DIDA&ved=0CEgQ6AEwBA#v=onepage&q=history of empires centralization&f=false).

[60] Stephen K. Sanderson, *Civilizations and World Systems: Studying World-Historical Change,* Rowman & Littlefield, 1995, p. 118-119 (http://books.google.com/books?id=8DJKxgdJsWMC&pg=PA117&dq=centralization+decentralization+rise+fall+empires&hl=en&sa=X&ei=p9USUYu2MYLF0QG40IG4CA&ved=0CDYQ6AEwAA#v=onepage&q=centralization decentralization rise fall empires&f=false) , ISBN 9780761991052

[61] Yildiz Atasoy, *Hegemonic Transitions, the State and Crisis in Neoliberal Capitalism,* Volume 7 of *Routledge Studies in Governance and Change in the Global Era, Taylor & Francis US, 2009, p. 65-67* (http://books.google.com/books?id=VcUA3MKuN5IC&pg=PA65&dq=centralization+decentralization+rise+fall+empires&hl=en&sa=X&ei=p9USUYu2MYLF0QG40IG4CA&ved=0CDwQ6AEwAQ#v=onepage&q=centralization decentralization rise fall empires&f=false)*, ISBN 9780415473842*

[62] Christopher K. Chase-Dunn, Thomas D. Hall, *Rise and Demise: Comparing World Systems*, Westview Press, 1997, p. 20, 33 (http://books.google.com/books?id=sIpw_9oI0RgC&printsec=frontcover&dq=Chase-Dunn+Hall+decentralization&hl=en&sa=X&ei=8Q0TUcCODofA9QTR-oGYCw&ved=0CDAQ6AEwAA#v=onepage&q=decentralization&f=false), ISBN 9780813310060

[63] Editors: S. N. Mishra, Anil Dutta Mishra, Sweta Mishra, *Public Governance and Decentralisation: Essays in Honour of T.N. Chaturvedi*, Mittal Publications, 2003, p. 229 (http://books.google.com/books?id=kpohGPGIyYMC&pg=PA229&dq=neo-liberalism+Mishra+decentralization&hl=en&sa=X&ei=2xYpUeuBNsXw0QGC8oEI&ved=0CDAQ6AEwAA#v=onepage&q=neo-liberalism Mishra decentralization&f=false), ISBN 9788170999188

[64] "Decentralization: A Sampling of Definitions", 1999, p. 5-8.

[65] *Managing Decentralisation: A New Role for Labour Market Policy*, Organisation for Economic Co-operation and Development, Local Economic and Employment Development (Program), OECD Publishing, 2003, p 135 (http://books.google.com/books?id=LJh1onzfJMIC&pg=PA135&dq=New+Public+Management+Decentralisation&hl=en&sa=X&ei=6WgUUcjVJevW0gHs3oH4AQ&

ved=0CEcQ6AEwBA#v=onepage&q=New Public Management Decentralisation&f=false), ISBN 9789264104709

[66] ”Decentralization: A Sampling of Definitions”, 1999, p. 8.

[67] Summary of Janet Kodras (http://government.cce.cornell.edu/doc/summary.asp?id=kodras1997), "Restructuring the State: Devolution, Privatization, and the Geographic Redistribution of Power and Capacity in Governance", from *State Devolution in America: Implications for a Diverse Society*, Edited by Lynn Staeheli, Janet Kodras, and Colin Flint, Urban Affairs Annual Reviews 48, SAGE, 1997, p. 79-68 at Restructuring local government project (http://government.cce.cornell.edu/) website.

[68] John Stossel, “ Private charity would do much more — if government hadn't crowded it out (http://www.jewishworldreview.com/0805/stossel082405.php3)”, Jewish World Review, August 24, 2005.

[69] David King, *Fiscal Tiers: The Economics of Multilevel Government,* George Allen and Unwin, 1984.

[70] Nico Groenendijk, "Fiscal federalism Revisited" paper presented at Institutions in Transition Conference organized by IMAD, Slovania Ljublijana.

[71] ”Decentralization: A Sampling of Definitions”, 1999, p. 18.

[72] Intergovernmental Fiscal Relations (http://www1.worldbank.org/publicsector/decentralization/fiscal.htm), Decentralization and Subnational Economies project, World Bank website, *accessed February 9, 2013*.

[73] Jerry Taylor and Peter Van Doren, Short-Circuited (http://www.cato.org/publications/commentary/shortcircuited), Wall Street Journal, August 30, 2007, reprinted at Cato Institute website.

[74] Mark A. Calabria, Did Deregulation Cause the Financial Crisis? (http://www.cato.org/sites/cato.org/files/serials/files/policy-report/2009/7/cpr31n4.pdf), *Cato Policy Report*, July/August 2009, Vol. XXXI No. 4p 1, 6-8.

[75] Emmanuelle Auriol and Michel Benaim, Standardization in Decentralized Economies (http://ideas.repec.org/a/aea/aecrev/v90y2000i3p550-570.html), The American Economic Review, June 2000, p. 1, 5.

[76] Terry L. Anderson, Vernon L. Smith, and Emily Simmons, How and Why to Privatize Federal Lands (http://perc.org/sites/default/files/pa1.pdf), Cato Institute policy paper No. 363, November 9, 1999.

[77] I. Scoones, *Beyond Farmer First*, London: Intermediate technology publications.

[78] Binkley, Robert C. Realism and Nationalism 1852-1871. Read Books. p. 118

[79] "The revolution abolishes private ownership of the means of production and distribution, and with it goes capitalistic business. Personal possession remains only in the things you use. Thus, your watch is your own, but the watch factory belongs to the people." [[Alexander Berkman (http://www.theanarchistlibrary.org/HTML/Alexander_Berkman__What_Is_Communist_Anarchism_.html)]. "What Is Communist Anarchism?"]

[80] As Noam Chomsky put it, a consistent libertarian "must oppose private ownership of the means of production and the wage slavery, which is a component of this system, as incompatible with the principle that labor must be freely undertaken and under the control of the producer". Chomsky (2003) p. 26 (http://books.google.com/books?id=Y75K660UfzsC&pg=PA26)

[81] Paul Zarembka. Transitions in Latin America and in Poland and Syria. Emerald Group Publishing, 2007. p. 25

[82] Guerin, Daniel. *Anarchism: A Matter of Words*: "Some contemporary anarchists have tried to clear up the misunderstanding by adopting a more explicit term: they align themselves with libertarian socialism or communism." Faatz, Chris, *Towards a Libertarian Socialism*.

[83] Ostergaard, Geoffrey. "Anarchism". *A Dictionary of Marxist Thought*. Blackwell Publishing, 1991. p. 21.

[84] Chomsky (2004) p. 739

[85] Ross, Dr. Jeffery Ian. *Controlling State Crime* (http://books.google.com/books?id=hX412xS2mkoC&printsec=frontcover), Transaction Publishers (2000) p. 400 ISBN 0-7658-0695-9

[86] Spiegel, Henry. *The Growth of Economic Thought* Duke University Press (1991) p.446

[87] Paul, Ellen Frankel et al. *Problems of Market Liberalism* Cambridge University Press (1998) p.305

[88] However, libertarian socialism retains respect for personal property.

[89] A Mutualist FAQ: A.4. Are Mutualists Socialists? (http://www.mutualist.org/id32.html). Mutualist.org. Retrieved on 2011-12-28.

[90] Murray Bookchin, *Ghost of Anarcho-Syndicalism*; Robert Graham, *The General Idea of Proudhon's Revolution*

[91] Kent Bromley, in his preface to Peter Kropotkin's book *The Conquest of Bread*, considered early French utopian socialist Charles Fourier to be the founder of the libertarian branch of socialist thought, as opposed to the authoritarian socialist ideas of Babeuf and Buonarroti." Kropotkin, Peter. *The Conquest of Bread*, preface by Kent Bromley, New York and London, G. P. Putnam's Sons, 1906.

[92] "(Benjamin) Tucker referred to himself many times as a socialist and considered his philosophy to be "Anarchistic socialism." *An Anarchist FAQ* by Various Authors

[93] French individualist anarchist Émile Armand shows clearly opposition to capitalism and centralized economies when he said that the individualist anarchist "inwardly he remains refractory – fatally refractory – morally, intellectually, economically (The capitalist economy and the directed economy, the speculators and the fabricators of single are equally repugnant to him.)" "Anarchist Individualism as a Life and Activity" by Emile Armand (http://www.spaz.org/~dan/individualist-anarchist/library/emile-armand/life-activity.html)

[94] Anarchist Peter Sabatini reports that In the United States "of early to mid-19th century, there appeared an array of communal and "utopian" counterculture groups (including the so-called free love movement). William Godwin's anarchism exerted an ideological influence on some of this, but more so the socialism of Robert Owen and Charles Fourier. After success of his British venture, Owen himself established a cooperative community within the United States at New Harmony, Indiana during 1825. One member of this commune was Josiah Warren (1798–1874), considered to be the first individualist anarchist" Peter Sabatini. "Libertarianism: Bogus Anarchy" (http://www.theanarchistlibrary.org/HTML/Peter_Sabatini__Libertarianism__Bogus_Anarchy.html)

[95] Lewis Herber. (Murray Bookchin) "Ecology and Revolutionary Thought" (http://www.theanarchistlibrary.org/HTML/Lewis_Herber__Murray_Bookchin___Ecology_and_Revolutionary_Thought.html). Theanarchistlibrary.org (2009-04-27). Retrieved on 2011-12-28.

[96] Sheehan, Sean. *Anarchism*, London: Reaktion Books, 2004. p. 85.

[97] Guerin, Daniel (ed.) *No Gods, No Masters*, AK Press, vol. 1, p. 62

[98] *The General Idea of the Revolution*, Pluto Press, p. 277 and p. 281

[99] Peter Kropotkin. "Anarchist Communism: Its Basis and Principles" (http://theanarchistlibrary.org/library/petr-kropotkin-anarchist-communism-its-basis-and-principles)

[100] George Woodcock. *Anarchism: a history of libertarian movements*. Pg. 357

[101] George Woodcock. *Anarchism: a history of libertarian movements*. Pg. 357

[102] "Anarchism" at the *[[Encyclopedia Britannica* (http://www.britannica.com/EBchecked/topic/22753/anarchism/66525/Anarchism-in-Spain#ref539322)] *online.]*

[103] "This process of education and class organization, more than any single factor in Spain, produced the collectives. And to the degree that the CNT-FAI (for the two organizations became fatally coupled after July 1936) exercised the major influence in an area, the collectives proved to be generally more durable, communist and resistant to Stalinist counterrevolution than other republican-held areas of Spain." [[Murray Bookchin (http://www.theanarchistlibrary.org/HTML/Murray_Bookchin__To_Remember_Spain__The_Anarchist_and_Syndicalist_Revolution_of_1936.html)]. *To Remember Spain: The Anarchist and Syndicalist Revolution of 1936*]

[104] [[Murray Bookchin (http://www.theanarchistlibrary.org/HTML/Murray_Bookchin__To_Remember_Spain__The_Anarchist_and_Syndicalist_Revolution_of_1936.html)]. *To Remember Spain: The Anarchist and Syndicalist Revolution of 1936*]

[105] Noam Chomsky Notes on Anarchism (http://www.chomsky.info/articles/1970----.htm)

[106] Marvin Zelkowitz, Editor, *Advances in Computers, Volume 82*, Academic Press, 2011, p. 3 (http://books.google.com/books?id=eFGi_414Y-cC&pg=PA3&dq="free+markets"+are+decentralized&hl=en&sa=X&ei=kiQkUa3EMc6s0AGc6YHgBg&ved=0CEoQ6AEwBQ#v=onepage&q="free markets" are decentralized&f=false), ISBN 9780123855138

[107] Eleanor Doyle, *The Economic System*, John Wiley & Sons, 2005, p. 61 (http://books.google.com/books?id=Ckz-4WbzlsgC&pg=PA61&dq="free+markets"+are+decentralized&hl=en&sa=X&ei=kiQkUa3EMc6s0AGc6YHgBg&ved=0CDAQ6AEwAA#v=onepage&q="free markets" are decentralized&f=false), ISBN 9780470015179

[108] Friedrich von Hayek, *The Road to Serfdom: Text and documents - the definitive edition; Volume 2 of Collected Works of F. A. Hayek*, edited by Bruce Caldwell, University of Chicago Press, 2009, p. 94 (http://books.google.com/books?id=qg61T_I1mwsC&pg=PA95&dq=F.+A.+Hayek+Decentralization&hl=en&sa=X&ei=G_8wUa6eKMKB0AGc9oHQCQ&ved=0CDsQ6AEwAQ#v=onepage&q=F.A.Hayek Decentralization&f=false), ISBN 9780226320533

[109] Bruce M. Owen, Antecedents to Net Neutrality (http://www.cato.org/sites/cato.org/files/serials/files/regulation/2007/7/v30n3-3.pdf), Cato Institute publication "Regulation", Fall, 2007, p. 16.

- Tibor R. Machan, *Private Rights & Public Illusions,* Transaction Publishers, 1995, p. 99 (http://books.google.com/books?id=W1WFgQLS-RcC&pg=PA99&dq="Big+Business+and+the+Rise+of+American+Statism"&hl=en&sa=X&ei=htkjUZ2bIMHB0AGC6oGYCQ&ved=0CD4Q6AEwAw#v=onepage&q="Big Business and the Rise of American Statism"&f=false), ISBN 9781412831925
- Tibor R. Machan, editor, *The Libertarian Alternative,* Nelson-Hall, 1974 included Yale Brozen's, "Is Government the source of monopoly? and other essays", Cato Institute, 1980; and Roy Childs' "Big Business and the Rise of American Statism", 1971, Reason.

[111] Gabriel Kolko, *The Triumph of Conservatism: A Reinterpretation of American History, 1900–1916*, Chapter Two: "Competition and Decentralization: The Failure to Rationalize Industry", Simon and Schuster, 2008, p. 26-56, 141, 220, 243, 351 (http://books.google.com/books?id=jTyfQk1zMTYC&pg=PA309&dq="Gabriel+Kolko"+decentralization&hl=en&sa=X&ei=60UkUeCWGbG80QHOjICICQ&ved=0CDAQ6AEwAA#v=onepage&q=decentralization&f=false), ISBN 9781439118726

[112] Roy Childs, " Big Business and the Rise of American Statism (http://praxeology.net/RC-BRS.htm)", Reason, 1971.

[113] D. T. Armentano, Antitrust Policy: Reform or Repeal? (http://www.cato.org/pubs/pas/pa021.html), Cato Institute Policy Analysis No. 21, January 18, 1983

[114] John Montgomery, *The New Wealth of Cities: City Dynamics and the Fifth Wave*, Ashgate Publishing, Ltd., 2008, p. 2 (http://books.google.com/books?id=5CCsHETenvMC&pg=PA2&dq=Jane+Jacobs+decentralization&hl=en&sa=X&ei=utkbUcHyBsfp0gHsxoEY&ved=0CEYQ6AEwBA#v=onepage&q=Jane Jacobs decentralization&f=false), ISBN 9780754674153

[115] Jane Jacobs, *The Question of Separatism: Quebec and the Struggle over Sovereignty*, (1980 Random House and 2011 Baraka Books), ISBN 978-1-926824-06-2

[116] Gopal Balakrishnan, *Mapping the Nation*, Verso, 1996, p. 277 (http://books.google.com/books?id=hdrfDqF3fLoC&pg=PA277&dq=Jane+Jacobs+Cities+and+the+wealth+of+nations+multiplication+of+sovereignties&hl=en&sa=X&ei=y-EbUdzvOcLi0gHkvoCIAQ&ved=0CDsQ6AEwAg#v=onepage&q=Jane Jacobs Cities and the wealth of nations multiplication of sovereignties&f=false), ISBN 9781859840603

[117] Jane Jacobs, *Cities and the Wealth of Nations: Principles of Economic Life,* Vintage Books, 1984, ISBN 0-394-72911-0

[118] Jeremy Magliaro, Amory Lovins, Valuing Decentralized Wastewater Technologies: A Catalog of Benefits, Costs, and Economic Analysis Techniques (http://www.rmi.org/Knowledge-Center/Library/W04-21_ValuingDecentralizedWastewater), Rocky Mountain Institute,

2004.

[119] Lawrence D. Smith, *Reform and Decentralization of Agricultural Services: A Policy Framework*, United Nationa Food and Agriculture Organization, 2001, p. 2010-211 (http://books.google.com/books?id=PtIqRuW0SVYC&pg=PA211&dq=decentralization+of+water+delivery&hl=en&sa=X&ei=rSAtUdO-M9PU0gGPzoBA&ved=0CEEQ6AEwAQ#v=onepage&q=decentralization of water delivery&f=false), ISBN 9789251046449

[120] Lawrence D. Smith, *Reform and Decentralization of Agricultural Services: A Policy Framework*, 2001.

[121] Maggie Koerth-Baker, What We Talk About When We Talk About the Decentralization of Energy (http://www.theatlantic.com/technology/archive/2012/04/what-we-talk-about-when-we-talk-about-the-decentralization-of-energy/255873/), The Atlantic, April 16, 2012.

[122] Fred E. Foldvary, Daniel Bruce Klein, Editors, *The Half-Life of Policy Rationales: How New Technology Affects Old Policy Issues* NYU Press, 2003, p. 1, 184 (http://books.google.com/books?id=pEg2pC6entUC&pg=PA184&dq=Half-Life+Policy+Rationales+Decentralization&hl=en&sa=X&ei=wRwtUY7NKeqw0AHElIGACA&ved=0CDAQ6AEwAA#v=onepage&q=Half-Life Policy Rationales Decentralization&f=false), ISBN 9780814747773

[123] John Baschab, Jon Piot, *The Executive's Guide to Information Technology*, John Wiley & Sons, 2007, p. 119 (http://books.google.com/books?id=YokHUkRBZoAC&pg=PA119&dq=decentralization+information+technology&hl=en&sa=X&ei=OtwqUYr3CMzH0AHs94CIBA&ved=0CHsQ6AEwCw#v=onepage&q=decentralization information technology&f=false), ISBN 9780470135914

[124] G. David Garson, *Modern Public Information Technology Systems: Issues and Challenges*, IGI Global, 2007, p. 115-120 (http://books.google.com/books?id=mf_kb6MKrOYC&pg=PT128&dq=information+technology+decentralization&hl=en&sa=X&ei=D7cqUbnwLtCr0AGw0IDgBw&ved=0CFgQ6AEwBQ), ISBN 9781599040530

[125] Thomas W. Malone, Robert Laubacher, Michael S. Scott Morton, *Inventing Organizations 21st Century*, MIT Press, 2003, 65-66 (http://books.google.com/books?id=az_UcjtGrSgC&dq=internet+is+decentralized++structure&source=gbs_navlinks_s), ISBN 9780262632737

[126] Chris DiBona, Mark Stone, Danese Cooper, *Open Sources 2.0: The Continuing Evolution*, O'Reilly Media, Inc., 2008, p. 316 (http://books.google.com/books?id=q9GnNrq3e5EC&pg=PA316&dq=wikipedia+decentralization&hl=en&sa=X&ei=JMsqUbfjJJC70AGyjIHYDw&ved=0CG0Q6AEwCg#v=onepage&q=wikipedia decentralization&f=false), ISBN 9780596553890

[127] Axel Bruns, *Blogs, Wikipedia, Second Life, and Beyond: From Production to Produsage*, Peter Lang, 2008, p. 80 (http://books.google.com/books?id=xWxmFNMKXhEC&pg=PA231&dq=wikipedia+decentralization&hl=en&sa=X&ei=JMsqUbfjJJC70AGyjIHYDw&ved=0CEsQ6AEwBA), ISBN 9780820488660

[128] Joseph Nye, The politics of the information age (http://www.praguepost.com/opinion/15511-the-politics-of-the-information-age.html), Prague Post, February 13, 2010.

[129] Adi Kamdar and Peter Eckersley, Can the FCC Create Public "Super WiFi Networks"? (https://www.eff.org/deeplinks/2013/02/fcc-public-super-wifi-networks), Electronic Frontier Foundation, February 5, 2013.

[130] Jennifer Holt, Alisa Perren, *Media Industries: History, Theory, and Method*, John Wiley & Sons, 2011, p. 1995-1997 (http://books.google.com/books?id=SJ1ZYY-8kj4C&pg=RA4-PA1996&dq="free+market"+decentralization+of+corporations+libertarian&hl=en&sa=X&ei=R9QjUfyxJNO20AGUnYA4&ved=0CDoQ6AEwAg#v=onepage&q="free market" decentralization of corporations libertarian&f=false), ISBN 9781444360233

[131] David G. Post and David R. Johnson, 'Chaos Prevailing on Every Continent': Towards a New Theory of Decentralized Decision-Making in Complex Systems (http://papers.ssrn.com/sol3/papers.cfm?abstract_id=157692), Chicago-Kent Law Review, Vol. 73, No. 4, p. 1055, 1998, full version at David G. Post's Temple University website (http://www.temple.edu/lawschool/dpost/chaos/chaos.htm).

[132] National Center for Appropriate Technology (http://www.ncat.org/regional_offices.php) website.

[133] Chapter 2. Decentralization and environmental issues (http://www.fao.org/docrep/005/y4256e/y4256e05.htm), "Environment in decentralized development", United Nations Food and Agriculture Organization ("FAO"), *accessed February 23, 2013*; also see Environment in Decentralized Decision Making, An Overview (http://www.fao.org/docs/up/easypol/346/en_decen_dec_making_016en.pdf), Agricultural Policy Support Service, Policy Assistance Division, FAO, Rome, Italy, November 2005.

[134] Summary of Remy Prud'homme, " The Dangers of Decentralization (http://government.cce.cornell.edu/doc/summary.asp?id=prudhomme1995)", World Bank Research Observer, 10(2):201, 1995, linked from Decentralization (http://government.cce.cornell.edu/doc/viewpage_r.asp?ID=Decentralization), article “Restructuring local government project” of Dr. Mildred Warner.

Further reading

General

- Furniss, Norman (1974). "The Practical Significance of Decentralization". *The Journal of Politics* **36** (4): 958–82. doi: 10.2307/2129402 (http://dx.doi.org/10.2307/2129402). ISSN 0022-3816 (http://www.worldcat.org/issn/0022-3816).
- Merilee Serrill Grindle, *Going Local: Decentralization, Democratization, And The Promise of Good Governance*, Princeton University Press, 2007, ISBN 9780691129075
- Daniel Treisman, *The Architecture of Government: Rethinking Political Decentralization*, *Cambridge University Press*, 2007, ISBN 9780521872294
- Richard M. Burton, Børge Obel, *Design Models for Hierarchical Organizations: Computation, Information, and Decentralization*, Springer, 1995, ISBN 9780792396093
- Dubois, H.F.W. & Fattore, G. (2009), Definitions and typologies in public administration research: the case of decentralization', International Journal of Public Administration (http://www.tandfonline.com/doi/abs/10.1080/01900690902908760#preview), 32(8): pp. 704–727.
- Kauffman, Bill (2008). "Decentralism" (http://books.google.com/books?id=yxNgXs3TkJYC). In Hamowy, Ronald. *The Encyclopedia of Libertarianism*. Thousand Oaks, CA: SAGE; Cato Institute. pp. 111–3. ISBN 978-1-4129-6580-4. LCCN 2008009151 (http://lccn.loc.gov/2008009151). OCLC 750831024 (http://www.worldcat.org/oclc/750831024).
- Miller, Michelle Ann & Tim Bunnell. (2012), guest editors. 'Asian Cities in an Era of Decentralisation', *Space and Polity*, Vol.16, No.1.
- Sharma, Chanchal Kumar (2006), Decentralization Dilemma: Measuring the Degree and Evaluating the Outcomes (http://papers.ssrn.com/sol3/papers.cfm?abstract_id=955113), The Indian Journal of Political Science, Vol.67, No.1, pp. 49–64.
- Sharma, Chanchal Kumar (2008), Emerging Dimensions of Decentralization Debate in the Age of Glocalization (http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1369943), MPRA Paper 6734, University Library of Munich, Germany; revised version Published as "Emerging Dimensions of Decentralization Debate in the Age of Globalization" in Indian Journal of Federal Studies Vol. 19 No.1 pp 47–65(year 2009).
- Schakel, Arjan H. (2008), Validation of the Regional Authority Index (http://www.unc.edu/~gwmarks/assets/doc/RFS.III.Validation of regional authority index.pdf), Regional and Federal Studies, Routlege, Vol. 18 (2).
- Decentralization (http://government.cce.cornell.edu/doc/viewpage_r.asp?ID=Decentralization), article at the " Restructuring local government project (http://government.cce.cornell.edu/)" of Dr. Mildred Warner, Cornell University includes a number of articles on decentralization trends and theories.
- Robert J. Bennett, ed., *Decentralization, Intergovernmental Relations and Markets: Towards a Post-Welfare Agenda,* Clarendon, 1990, p. 1-26. ISBN 978-0198286875

Case studies

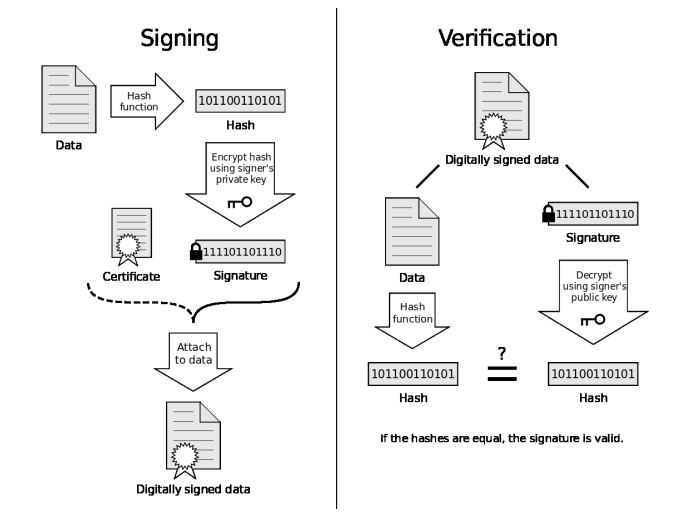
- Peter Aucoin, Herman Bakvis, *The Centralization-Decentralization Conundrum: Organization and Management in the Canadian Government*, IRPP, 1988, ISBN 9780886450700
- Jean-Paul Faguet, *Decentralization and Popular Democracy: Governance from Below in Bolivia*, University of Michigan Press, 2012, ISBN 9780472118199
- Harvey S. Rosen, Editor, *Fiscal Federalism: Quantitative Studies National Bureau of Economic Research Project Report*, NBER-Project Report, University of Chicago Press, 2008, ISBN 9780226726236
- Tim Campbell, *Quiet Revolution: Decentralization and the Rise of Political Participation in Latin American Cities*, University of Pittsburgh Press, 2003, ISBN 9780822957966
- Fisman, Raymond and Roberta Gatti (2002), Decentralization and Corruption: Evidence Across Countries (http://elibrary.worldbank.org/content/workingpaper/10.1596/1813-9450-2290), Journal of Public Economics, Vol.83, No.3, pp. 325–45.

- Frischmann, Eva (2010), Decentralization and Corruption. A Cross-Country Analysis (http://www.imf.org/external/pubs/ft/wp), Grin Verlag, 978-3640710959.
- Miller, Michelle Ann, ed. (2012). *Autonomy and Armed Separatism in South and Southeast Asia* (Singapore: ISEAS).
- Michelle Ann Miller (2009). *Rebellion and Reform in Indonesia. Jakarta's Security and Autonomy Policies in Aceh* (London and New York: Routledge).

Digital signature

This article is about cryptographic signatures. For simple signatures in digital form, see Electronic signature.

A **digital signature** is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, such that the sender cannot deny having sent the message (authentication and non-repudiation) and that the message was not altered in transit (integrity). Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering.



A diagram showing how a digital signature is applied and then verified.

Explanation

Digital signatures are often used to implement electronic signatures, a broader term that refers to any electronic data that carries the intent of a signature,[1] but not all electronic signatures use digital signatures.[2][3] In some countries, including the United States, India, Brazil, and members of the European Union, electronic signatures have legal significance.

Digital signatures employ a type of asymmetric cryptography. For messages sent through a nonsecure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital signatures are equivalent to traditional handwritten signatures in many respects, but properly implemented digital signatures are more difficult to forge than the handwritten type. Digital signature schemes, in the sense used here, are cryptographically based, and must be implemented properly to be effective. Digital signatures can also provide non-repudiation, meaning that the signer cannot successfully claim they did not sign a message, while also claiming their private key remains secret; further, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is valid. Digitally signed messages may be anything representable as a bitstring: examples include electronic mail, contracts, or a message sent via some other cryptographic protocol.

Definition

Main article: Public-key cryptography

A digital signature scheme typically consists of three algorithms:

- A *key generation* algorithm that selects a *private key* uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding *public key*.
- A *signing* algorithm that, given a message and a private key, produces a signature.
- A *signature verifying* algorithm that, given a message, public key and a signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key.

History

In 1976, Whitfield Diffie and Martin Hellman first described the notion of a digital signature scheme, although they only conjectured that such schemes existed.[4][5] Soon afterwards, Ronald Rivest, Adi Shamir, and Len Adleman invented the RSA algorithm, which could be used to produce primitive digital signatures (although only as a proof-of-concept – "plain" RSA signatures are not secure[6]). The first widely marketed software package to offer digital signature was Lotus Notes 1.0, released in 1989, which used the RSA algorithm.Wikipedia:Citation needed

Other digital signature schemes were soon developed after RSA, the earliest being Lamport signatures,[7] Merkle signatures (also known as "Merkle trees" or simply "Hash trees"),[8] and Rabin signatures.[9]

In 1988, Shafi Goldwasser, Silvio Micali, and Ronald Rivest became the first to rigorously define the security requirements of digital signature schemes.[10] They described a hierarchy of attack models for signature schemes, and also present the GMR signature scheme, the first that can be proven to prevent even an existential forgery against a chosen message attack.

How they work

To create RSA signature keys, generate an RSA key pair containing a modulus N that is the product of two large primes, along with integers e and d such that $e\,d \equiv 1 \pmod{\varphi(N)}$, where φ is the Euler phi-function. The signer's public key consists of N and e, and the signer's secret key contains d.

To sign a message m, the signer computes $\sigma \equiv m^d \pmod{N}$. To verify, the receiver checks that $\sigma^e \equiv m \pmod{N}$.

As noted earlier, this basic scheme is not very secure. To prevent attacks, one can first apply a cryptographic hash function to the message m and then apply the RSA algorithm described above to the result. This approach can be proven secure in the so-called random oracle modelWikipedia:Please clarify. Most early signature schemes were of a similar type: they involve the use of a trapdoor permutation, such as the RSA function, or in the case of the Rabin signature scheme, computing square modulo composite n. A trapdoor permutation family is a family of permutations, specified by a parameter, that is easy to compute in the forward direction, but is difficult to compute in the reverse direction without already knowing the private key. However, for every parameter there is a "trapdoor" (private key) which when known, easily decrypts the message. Trapdoor permutations can be viewed as public-key encryption systems, where the parameter is the public key and the trapdoor is the secret key, and where encrypting corresponds to computing the forward direction of the permutation, while decrypting corresponds to the reverse direction. Trapdoor permutations can also be viewed as digital signature schemes, where computing the reverse direction with the secret key is thought of as signing, and computing the forward direction is done to verify signatures. Because of this correspondence, digital signatures are often described as based on public-key cryptosystems, where signing is equivalent to decryption and verification is equivalent to encryption, but this is not

the only way digital signatures are computed.

Used directly, this type of signature scheme is vulnerable to a key-only existential forgery attack. To create a forgery, the attacker picks a random signature σ and uses the verification procedure to determine the message *m* corresponding to that signature.[11] In practice, however, this type of signature is not used directly, but rather, the message to be signed is first hashed to produce a short digest that is then signed. This forgery attack, then, only produces the hash function output that corresponds to σ, but not a message that leads to that value, which does not lead to an attack. In the random oracle model, this hash-then-sign form of signature is existentially unforgeable, even against a chosen-plaintext attack.Wikipedia:Please clarify

There are several reasons to sign such a hash (or message digest) instead of the whole document.

- **For efficiency:** The signature will be much shorter and thus save time since hashing is generally much faster than signing in practice.
- **For compatibility:** Messages are typically bit strings, but some signature schemes operate on other domains (such as, in the case of RSA, numbers modulo a composite number *N*). A hash function can be used to convert an arbitrary input into the proper format.
- **For integrity:** Without the hash function, the text "to be signed" may have to be split (separated) in blocks small enough for the signature scheme to act on them directly. However, the receiver of the signed blocks is not able to recognize if all the blocks are present and in the appropriate order.

Notions of security

In their foundational paper, Goldwasser, Micali, and Rivest lay out a hierarchy of attack models against digital signatures:

1. In a *key-only* attack, the attacker is only given the public verification key.
2. In a *known message* attack, the attacker is given valid signatures for a variety of messages known by the attacker but not chosen by the attacker.
3. In an *adaptive chosen message* attack, the attacker first learns signatures on arbitrary messages of the attacker's choice.

They also describe a hierarchy of attack results:

1. A *total break* results in the recovery of the signing key.
2. A universal forgery attack results in the ability to forge signatures for any message.
3. A selective forgery attack results in a signature on a message of the adversary's choice.
4. An existential forgery merely results in some valid message/signature pair not already known to the adversary.

The strongest notion of security, therefore, is security against existential forgery under an adaptive chosen message attack.

Applications of digital signatures

As organizations move away from paper documents with ink signatures or authenticity stamps, digital signatures can provide added assurances of the evidence to provenance, identity, and status of an electronic document as well as acknowledging informed consent and approval by a signatory. The United States Government Printing Office (GPO) publishes electronic versions of the budget, public and private laws, and congressional bills with digital signatures. Universities including Penn State, University of Chicago, and Stanford are publishing electronic student transcripts with digital signatures.

Below are some common reasons for applying a digital signature to communications:

Authentication

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

Integrity

In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to *change* an encrypted message without understanding it. (Some encryption algorithms, known as nonmalleable ones, prevent this, but others do not.) However, if a message is digitally signed, any change in the message after signature invalidates the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions (see collision resistance).

Non-repudiation

Non-repudiation, or more specifically *non-repudiation of origin*, is an important aspect of digital signatures. By this property, an entity that has signed some information cannot at a later time deny having signed it. Similarly, access to the public key only does not enable a fraudulent party to fake a valid signature.

Note that these authentication, non-repudiation etc. properties rely on the secret key *not having been revoked* prior to its usage. Public revocation of a key-pair is a required ability, else leaked secret keys would continue to implicate the claimed owner of the key-pair. Checking revocation status requires an "online" check, e.g. checking a "Certificate Revocation List" or via the "Online Certificate Status Protocol". Very roughly this is analogous to a vendor who receives credit-cards first checking online with the credit-card issuer to find if a given card has been reported lost or stolen. Of course, with stolen key pairs, the theft is often discovered only after the secret key's use, e.g., to sign a bogus certificate for espionage purposes.

Additional security precautions

Putting the private key on a smart card

All public key / private key cryptosystems depend entirely on keeping the private key secret. A private key can be stored on a user's computer, and protected by a local password, but this has two disadvantages:

- the user can only sign documents on that particular computer
- the security of the private key depends entirely on the security of the computer

A more secure alternative is to store the private key on a smart card. Many smart cards are designed to be tamper-resistant (although some designs have been broken, notably by Ross Anderson and his students). In a typical digital signature implementation, the hash calculated from the document is sent to the smart card, whose CPU signs the hash using the stored private key of the user, and then returns the signed hash. Typically, a user must activate his smart card by entering a personal identification number or PIN code (thus providing two-factor authentication). It can be arranged that the private key never leaves the smart card, although this is not always implemented. If the smart card is stolen, the thief will still need the PIN code to generate a digital signature. This reduces the security of the scheme to that of the PIN system, although it still requires an attacker to possess the card. A mitigating factor is

that private keys, if generated and stored on smart cards, are usually regarded as difficult to copy, and are assumed to exist in exactly one copy. Thus, the loss of the smart card may be detected by the owner and the corresponding certificate can be immediately revoked. Private keys that are protected by software only may be easier to copy, and such compromises are far more difficult to detect.

Using smart card readers with a separate keyboard

Entering a PIN code to activate the smart card commonly requires a numeric keypad. Some card readers have their own numeric keypad. This is safer than using a card reader integrated into a PC, and then entering the PIN using that computer's keyboard. Readers with a numeric keypad are meant to circumvent the eavesdropping threat where the computer might be running a keystroke logger, potentially compromising the PIN code. Specialized card readers are also less vulnerable to tampering with their software or hardware and are often EAL3 certified.

Other smart card designs

Smart card design is an active field, and there are smart card schemes which are intended to avoid these particular problems, though so far with little security proofs.

Using digital signatures only with trusted applications

One of the main differences between a digital signature and a written signature is that the user does not "see" what he signs. The user application presents a hash code to be signed by the digital signing algorithm using the private key. An attacker who gains control of the user's PC can possibly replace the user application with a foreign substitute, in effect replacing the user's own communications with those of the attacker. This could allow a malicious application to trick a user into signing any document by displaying the user's original on-screen, but presenting the attacker's own documents to the signing application.

To protect against this scenario, an authentication system can be set up between the user's application (word processor, email client, etc.) and the signing application. The general idea is to provide some means for both the user application and signing application to verify each other's integrity. For example, the signing application may require all requests to come from digitally signed binaries.

WYSIWYS

Main article: WYSIWYS

Technically speaking, a digital signature applies to a string of bits, whereas humans and applications "believe" that they sign the semantic interpretation of those bits. In order to be semantically interpreted, the bit string must be transformed into a form that is meaningful for humans and applications, and this is done through a combination of hardware and software based processes on a computer system. The problem is that the semantic interpretation of bits can change as a function of the processes used to transform the bits into semantic content. It is relatively easy to change the interpretation of a digital document by implementing changes on the computer system where the document is being processed. From a semantic perspective this creates uncertainty about what exactly has been signed. WYSIWYS (What You See Is What You Sign) [12] means that the semantic interpretation of a signed message cannot be changed. In particular this also means that a message cannot contain hidden information that the signer is unaware of, and that can be revealed after the signature has been applied. WYSIWYS is a necessary requirement for the validity of digital signatures, but this requirement is difficult to guarantee because of the increasing complexity of modern computer systems.

Digital signatures vs. ink on paper signatures

An ink signature could be replicated from one document to another by copying the image manually or digitally, but to have credible signature copies that can resist some scrutiny is a significant manual or technical skill, and to produce ink signature copies that resist professional scrutiny is very difficult.

Digital signatures cryptographically bind an electronic identity to an electronic document and the digital signature cannot be copied to another document. Paper contracts sometimes have the ink signature block on the last page, and the previous pages may be replaced after a signature is applied. Digital signatures can be applied to an entire document, such that the digital signature on the last page will indicate tampering if any data on any of the pages have been altered, but this can also be achieved by signing with ink and numbering all pages of the contract.

Some digital signature algorithms

- RSA-based signature schemes, such as RSA-PSS
- DSA and its elliptic curve variant ECDSA
- ElGamal signature scheme as the predecessor to DSA, and variants Schnorr signature and Pointcheval–Stern signature algorithm
- Rabin signature algorithm
- Pairing-based schemes such as BLS
- Undeniable signatures
- Aggregate signature - a signature scheme that supports aggregation: Given n signatures on n messages from n users, it is possible to aggregate all these signatures into a single signature whose size is constant in the number of users. This single signature will convince the verifier that the n users did indeed sign the n original messages.
- Signatures with efficient protocols - are signature schemes that facilitate efficient cryptographic protocols such as zero-knowledge proofs or secure computation.

The current state of use – legal and practical

All digital signature schemes share the following basic prerequisites regardless of cryptographic theory or legal provision:

Quality algorithms

Some public-key algorithms are known to be insecure, practical attacks against them having been discovered.

Quality implementations

An implementation of a good algorithm (or protocol) with mistake(s) will not work.

The private key must remain private

If the private key becomes known to any other party, that party can produce *perfect* digital signatures of anything whatsoever.

The public key owner must be verifiable

A public key associated with Bob actually came from Bob. This is commonly done using a public key infrastructure (PKI) and the public key↔user association is attested by the operator of the PKI (called a certificate authority). For 'open' PKIs in which anyone can request such an attestation (universally embodied in a cryptographically protected identity certificate), the possibility of mistaken attestation is nontrivial. Commercial PKI operators have suffered several publicly known problems. Such mistakes could lead to falsely signed, and thus wrongly attributed, documents. 'Closed' PKI systems are more expensive, but less easily subverted in this way.

Users (and their software) must carry out the signature protocol properly.

Only if all of these conditions are met will a digital signature actually be any evidence of who sent the message, and therefore of their assent to its contents. Legal enactment cannot change this reality of the existing engineering possibilities, though some such have not reflected this actuality.

Legislatures, being importuned by businesses expecting to profit from operating a PKI, or by the technological avant-garde advocating new solutions to old problems, have enacted statutes and/or regulations in many jurisdictions authorizing, endorsing, encouraging, or permitting digital signatures and providing for (or limiting) their legal effect. The first appears to have been in Utah in the United States, followed closely by the states Massachusetts and California. Other countries have also passed statutes or issued regulations in this area as well and the UN has had an active model law project for some time. These enactments (or proposed enactments) vary from place to place, have typically embodied expectations at variance (optimistically or pessimistically) with the state of the underlying cryptographic engineering, and have had the net effect of confusing potential users and specifiers, nearly all of whom are not cryptographically knowledgeable. Adoption of technical standards for digital signatures have lagged behind much of the legislation, delaying a more or less unified engineering position on interoperability, algorithm choice, key lengths, and so on what the engineering is attempting to provide.

See also: ABA digital signature guidelines

Industry standards

Some industries have established common interoperability standards for the use of digital signatures between members of the industry and with regulators. These include the Automotive Network Exchange for the automobile industry and the SAFE-BioPharma Association for the healthcare industry.

Using separate key pairs for signing and encryption

In several countries, a digital signature has a status somewhat like that of a traditional pen and paper signature, like in the EU digital signature legislation [13]. Generally, these provisions mean that anything digitally signed legally binds the signer of the document to the terms therein. For that reason, it is often thought best to use separate key pairs for encrypting and signing. Using the encryption key pair, a person can engage in an encrypted conversation (*e.g.*, regarding a real estate transaction), but the encryption does not legally sign every message he sends. Only when both parties come to an agreement do they sign a contract with their signing keys, and only then are they legally bound by the terms of a specific document. After signing, the document can be sent over the encrypted link. If a signing key is lost or compromised, it can be revoked to mitigate any future transactions. If an encryption key is lost, a backup or key escrow should be utilized to continue viewing encrypted content. Signing keys should never be backed up or escrowed.

Notes

[1] US ESIGN Act of 2000 (http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=106_cong_public_laws&docid=f:publ229.106.pdf)

[2] State of WI (http://enterprise.state.wi.us/home/strategic/esig.htm)

[3] National Archives of Australia (http://www.naa.gov.au/recordkeeping/er/Security/6-glossary.html)

[4] "New Directions in Cryptography", IEEE Transactions on Information Theory, IT-22(6):644–654, Nov. 1976.

[5] " Signature Schemes and Applications to Cryptographic Protocol Design (http://theory.lcs.mit.edu/~cis/theses/anna-phd.pdf)", Anna Lysyanskaya, PhD thesis, MIT, 2002.

[6] For example any integer r "signs" $m=r^e$ and the product $s_1 s_2$ of any two valid signatures s_1, s_2 of m_1, m_2 is a valid signature of the product $m_1 m_2$.

[7] "Constructing digital signatures from a one-way function.", Leslie Lamport, Technical Report CSL-98, SRI International, Oct. 1979.

[8] "A certified digital signature", Ralph Merkle, In Gilles Brassard, ed., Advances in Cryptology – CRYPTO '89, vol. 435 of Lecture Notes in Computer Science, pp. 218–238, Spring Verlag, 1990.

[9] "Digitalized signatures as intractable as factorization." Michael O. Rabin, Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Jan. 1979

[10] "A digital signature scheme secure against adaptive chosen-message attacks.", Shafi Goldwasser, Silvio Micali, and Ronald Rivest. SIAM Journal on Computing, 17(2):281–308, Apr. 1988.
[11] "Modern Cryptography: Theory & Practice", Wenbo Mao, Prentice Hall Professional Technical Reference, New Jersey, 2004, pg. 308. ISBN 0-13-066943-1
[12] A. Jøsang, D. Povey and A. Ho. "What You See is Not Always What You Sign". Proceedings of the Australian Unix User Group Symposium (AUUG2002), Melbourne, September 2002. PDF (http://www.unik.no/people/josang/papers/JPH2002-AUUG.pdf)
[13] http://europa.eu/legislation_summaries/information_society/l24118_en.htm

Further reading

- J. Katz and Y. Lindell, "Introduction to Modern Cryptography" (Chapman & Hall/CRC Press, 2007)
- Stephen Mason, Electronic Signatures in Law (3rd edition, Cambridge University Press, 2012)
- Lorna Brazell, Electronic Signatures and Identities Law and Regulation (2nd edn, London: Sweet & Maxwell, 2008);
- Dennis Campbell, editor, E-Commerce and the Law of Digital Signatures (Oceana Publications, 2005).
- M. H. M Schellenkens, Electronic Signatures Authentication Technology from a Legal Perspective, (TMC Asser Press, 2004).
- Jeremiah S. Buckley, John P. Kromer, Margo H. K. Tank, and R. David Whitaker, The Law of Electronic Signatures (3rd Edition, West Publishing, 2010).
- *Digital Evidence and Electronic Signature Law Review* (http://journals.sas.ac.uk/deeslr/) Free open source
- OpenCertificateSigning server (https://trust1t.atlassian.net/wiki/display/OCSPublic/01.+Digital+signatures+in+PDF) Open-source PDF Signing (EU, ETSI)

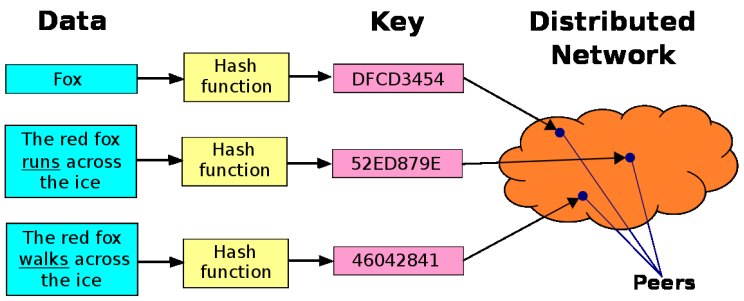
Distributed hash table

A **distributed hash table** (**DHT**) is a class of a decentralized distributed system that provides a lookup service similar to a hash table; (*key*, *value*) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

DHTs form an infrastructure that can be used to build more complex services, such as anycast, cooperative Web caching, distributed file systems, domain name services, instant messaging, multicast, and also peer-to-peer file sharing and content distribution systems. Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the Coral Content Distribution Network, the Kad network, the Storm botnet, Freenet and the YaCy search engine.

History

DHT research was originally motivated, in part, by peer-to-peer systems such as Freenet, gnutella, BitTorrent and Napster, which took advantage of resources distributed across the Internet to provide a single useful application. In particular, they



Distributed hash tables

took advantage of increased bandwidth and hard disk capacity to provide a file-sharing service.

These systems differed in how they *found* the data their peers contained:

- Napster, the first large-scale P2P content delivery system, required a central index server: each node, upon joining, would send a list of locally held files to the server, which would perform searches and refer the querier to the nodes that held the results. This central component left the system vulnerable to attacks and lawsuits.Wikipedia:Citation needed
- Gnutella and similar networks moved to a flooding query model – in essence, each search would result in a message being broadcast to every other machine in the network. While avoiding a single point of failure, this method was significantly less efficient than Napster. Later versions of Gnutella clients moved to a dynamic querying model which vastly improved efficiency.Wikipedia:Citation needed
- Finally, Freenet is fully distributed, but employs a heuristic key-based routing in which each file is associated with a key, and files with similar keys tend to cluster on a similar set of nodes. Queries are likely to be routed through the network to such a cluster without needing to visit many peers. However, Freenet does not guarantee that data will be found.

Distributed hash tables use a more structured key-based routing in order to attain both the decentralization of Freenet and gnutella, and the efficiency and guaranteed results of Napster. One drawback is that, like Freenet, DHTs only directly support exact-match search, rather than keyword search, although Freenet's routing algorithm can be generalized to any key type where a closeness operation can be defined.

In 2001, four systems—CAN, Chord,[1] Pastry, and Tapestry—ignited DHTs as a popular research topic. A project called the Infrastructure for Resilient Internet Systems (Iris) was funded by a $12 million grant from the US National Science Foundation in 2002. Researchers included Hari Balakrishnan and Scott Shenker. Outside academia, DHT technology has been adopted as a component of BitTorrent and in the Coral Content Distribution Network.

Properties

DHTs characteristically emphasize the following properties:

- Autonomy and Decentralization: the nodes collectively form the system without any central coordination.
- Fault tolerance: the system should be reliable (in some sense) even with nodes continuously joining, leaving, and failing.
- Scalability: the system should function efficiently even with thousands or millions of nodes.

A key technique used to achieve these goals is that any one node needs to coordinate with only a few other nodes in the system – most commonly, O(log n) of the n participants (see below) – so that only a limited amount of work needs to be done for each change in membership.

Some DHT designs seek to be secure against malicious participants[2] and to allow participants to remain anonymous, though this is less common than in many other peer-to-peer (especially file sharing) systems; see anonymous P2P.

Finally, DHTs must deal with more traditional distributed systems issues such as load balancing, data integrity, and performance (in particular, ensuring that operations such as routing and data storage or retrieval complete quickly).

Structure

The structure of a DHT can be decomposed into several main components.[3][4] The foundation is an abstract **keyspace**, such as the set of 160-bit strings. A **keyspace partitioning** scheme splits ownership of this keyspace among the participating nodes. An **overlay network** then connects the nodes, allowing them to find the owner of any given key in the keyspace.

Once these components are in place, a typical use of the DHT for storage and retrieval might proceed as follows. Suppose the keyspace is the set of 160-bit strings. To store a file with given $filename$ and $data$ in the DHT, the SHA-1 hash of $filename$ is generated, producing a 160-bit key k, and a message $put(k, data)$ is sent to any node participating in the DHT. The message is forwarded from node to node through the overlay network until it reaches the single node responsible for key k as specified by the keyspace partitioning. That node then stores the key and the data. Any other client can then retrieve the contents of the file by again hashing $filename$ to produce k and asking any DHT node to find the data associated with k with a message $get(k)$. The message will again be routed through the overlay to the node responsible for k, which will reply with the stored $data$.
The keyspace partitioning and overlay network components are described below with the goal of capturing the principal ideas common to most DHTs; many designs differ in the details.

Keyspace partitioning

Most DHTs use some variant of consistent hashing to map keys to nodes. This technique employs a function $\delta(k_1, k_2)$ that defines an abstract notion of the *distance* between the keys k_1 and k_2, which is unrelated to geographical distance or network latency. Each node is assigned a single key called its *identifier* (ID). A node with ID i_x owns all the keys k_m for which i_x is the closest ID, measured according to $\delta(k_m, i_x)$.

> **Example.** The Chord DHT treats keys as points on a circle, and $\delta(k_1, k_2)$ is the distance traveling clockwise around the circle from k_1 to k_2. Thus, the circular keyspace is split into contiguous segments whose endpoints are the node identifiers. If i_1 and i_2 are two adjacent IDs, with a shorter clockwise distance from i_1 to i_2, then the node with ID i_2 owns all the keys that fall between i_1 and i_2.

Consistent hashing has the essential property that removal or addition of one node changes only the set of keys owned by the nodes with adjacent IDs, and leaves all other nodes unaffected. Contrast this with a traditional hash table in which addition or removal of one bucket causes nearly the entire keyspace to be remapped. Since any change in ownership typically corresponds to bandwidth-intensive movement of objects stored in the DHT from one node to another, minimizing such reorganization is required to efficiently support high rates of churn (node arrival and failure).

Locality-preserving hashing ensures that similar keys are assigned to similar objects. This can enable a more efficient execution of range queries. Self-Chord [5] decouples object keys from peer IDs and sorts keys along the ring with a statistical approach based on the swarm intelligence paradigm. Sorting ensures that similar keys are stored by neighbour nodes and that discovery procedures, including range queries, can be performed in logarithmic time.

Overlay network

Each node maintains a set of links to other nodes (its *neighbors* or routing table) . Together, these links form the overlay network. A node picks its neighbors according to a certain structure, called the network's topology.

All DHT topologies share some variant of the most essential property: for any key k, each node either has a node ID that owns k or has a link to a node whose node ID is *closer* to k, in terms of the keyspace distance defined above. It is then easy to route a message to the owner of any key k using the following greedy algorithm (that is not necessarily globally optimal): at each step, forward the message to the neighbor whose ID is closest to k. When there is no such neighbor, then we must have arrived at the closest node, which is the owner of k as defined above.

This style of routing is sometimes called key-based routing.

Beyond basic routing correctness, two important constraints on the topology are to guarantee that the maximum number of hops in any route (route length) is low, so that requests complete quickly; and that the maximum number of neighbors of any node (maximum node degree) is low, so that maintenance overhead is not excessive. Of course, having shorter routes requires higher maximum degree. Some common choices for maximum degree and route length are as follows, where n is the number of nodes in the DHT, using Big O notation:

Max. degree	Route length	Used in	Note
$O(1)$	$O(n)$		
$O(\log n)$	$O(\log n / \log(\log n))$	Koorde	
$O(\log n)$	$O(\log n)$	Chord	most common, but not optimal (degree/route length)
$O(1)$	$O(\log n)$		
$O(\sqrt{n})$	$O(1)$		

The most common choice, $O(\log n)$ degree/route length, is not optimal in terms of degree/route length tradeoff, as such topologies typically allow more flexibility in choice of neighbors. Many DHTs use that flexibility to pick neighbors that are close in terms of latency in the physical underlying network.

Maximum route length is closely related to diameter: the maximum number of hops in any shortest path between nodes. Clearly, the network's worst case route length is at least as large as its diameter, so DHTs are limited by the degree/diameter tradeoff that is fundamental in graph theory. Route length can be greater than diameter, since the greedy routing algorithm may not find shortest paths.[6]

Algorithms for overlay networks

Aside from routing, there exist many algorithms that exploit the structure of the overlay network for sending a message to all nodes, or a subset of nodes, in a DHT.[7] These algorithms are used by applications to do overlay multicast, range queries, or to collect statistics. Two systems that are based on this approach are Structella,[8] which implements flooding and random walks on a Pastry overlay, and DQ-DHT,[9] which implements a dynamic querying search algorithm over a Chord network.

DHT implementations

Most notable differences encountered in practical instances of DHT implementations include at least the following:

- The address space is a parameter of DHT. Several real world DHTs use 128-bit or 160-bit key space
- Some real-world DHTs use hash functions other than SHA-1.
- In the real world the key k could be a hash of a file's *content* rather than a hash of a file's *name* to provide content-addressable storage, so that renaming of the file does not prevent users from finding it.
- Some DHTs may also publish objects of different types. For example, key k could be the node ID and associated data could describe how to contact this node. This allows publication-of-presence information and often used in IM applications, etc. In the simplest case, ID is just a random number that is directly used as key k (so in a 160-bit DHT ID will be a 160-bit number, usually randomly chosen). In some DHTs, publishing of nodes IDs is also used to optimize DHT operations.
- Redundancy can be added to improve reliability. The $(k, data)$ key pair can be stored in more than one node corresponding to the key. Usually, rather than selecting just one node, real world DHT algorithms select i suitable nodes, with i being an implementation-specific parameter of the DHT. In some DHT designs, nodes agree to handle a certain keyspace range, the size of which may be chosen dynamically, rather than hard-coded.
- Some advanced DHTs like Kademlia perform iterative lookups through the DHT first in order to select a set of suitable nodes and send $put(k, data)$ messages only to those nodes, thus drastically reducing useless traffic,

since published messages are only sent to nodes that seem suitable for storing the key k ; and iterative lookups cover just a small set of nodes rather than the entire DHT, reducing useless forwarding. In such DHTs, forwarding of $put(k, data)$ messages may only occur as part of a self-healing algorithm: if a target node receives a $put(k, data)$ message that k is out of its handled range and a closer node (in terms of DHT keyspace) is known, the message is forwarded to that node. Otherwise, data are indexed locally. This leads to a somewhat self-balancing DHT behavior. Of course, such an algorithm requires nodes to publish their presence data in the DHT so the iterative lookups can be performed.

Examples

- DHT protocols and implementations
  - Apache Cassandra
  - BATON Overlay
  - Mainline DHT - Standard DHT used by BitTorrent (based on Kademlia as provided by Khashmir[10])
  - CAN (Content Addressable Network)
  - Chord
  - Koorde
  - Kademlia
  - Pastry
  - P-Grid
  - Tapestry
  - TomP2P
- Applications employing DHTs
  - BTDigg: BitTorrent DHT search engine
  - cjdns: routing engine for mesh-based networks
  - CloudSNAP: a decentralized web application deployment platform
  - Codeen: web caching
  - Coral Content Distribution Network
  - FAROO: peer-to-peer Web search engine
  - Freenet: a censorship-resistant anonymous network
  - GlusterFS: a distributed file system used for storage virtualization
  - GNUnet: Freenet-like distribution network including a DHT implementation
  - JXTA: open-source P2P platform
  - Oracle Coherence: an in-memory data grid built on top of a Java DHT implementation
  - Retroshare: a Friend-to-friend network[11]
  - YaCy: a distributed search engine
  - Twister: a microblogging peer-to-peer platform
  - Perfect Dark: a peer-to-peer file-sharing application from Japan

References

[1] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in P2P systems (http://www.cs.berkeley.edu/~istoica/papers/2003/cacm03.pdf). In Communications of the ACM, February 2003.

[2] Guido Urdaneta, Guillaume Pierre and Maarten van Steen. A Survey of DHT Security Techniques (http://www.globule.org/publi/SDST_acmcs2009.html). ACM Computing Surveys 43(2), January 2011.

[3] Moni Naor and Udi Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach (http://www.wisdom.weizmann.ac.il/~naor/PAPERS/dh.pdf). Proc. SPAA, 2003.

[4] Gurmeet Singh Manku. Dipsea: A Modular Distributed Hash Table (http://www-db.stanford.edu/~manku/phd/index.html). Ph. D. Thesis (Stanford University), August 2004.

[5] Agostino Forestiero, Emilio Leonardi, Carlo Mastroianni and Michela Meo. Self-Chord: a Bio-Inspired P2P Framework for Self-Organizing Distributed Systems (http://dx.doi.org/10.1109/TNET.2010.2046745). IEEE/ACM Transactions on Networking, 2010.

[6] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy Neighbor's Neighbor: the Power of Lookahead in Randomized P2P Networks (http://citeseer.ist.psu.edu/naor04know.html). Proc. STOC, 2004.

[7] Ali Ghodsi. Distributed k-ary System: Algorithms for Distributed Hash Tables (http://www.sics.se/~ali/thesis/). KTH-Royal Institute of Technology, 2006.

[8] Miguel Castro, Manuel Costa, and Antony Rowstron. Should we build Gnutella on a structured overlay? (http://dx.doi.org/10.1145/972374.972397). Computer Communication Review, 2004.

[9] Domenico Talia and Paolo Trunfio. Enabling Dynamic Querying over Distributed Hash Tables (http://dx.doi.org/10.1016/j.jpdc.2010.08.012). Journal of Parallel and Distributed Computing, 2010.

[10] Tribler wiki (http://www.tribler.org/trac/wiki/Khashmir) retrieved January 2010.

[11] Retroshare FAQ (http://retroshare.sourceforge.net/wiki/index.php/Frequently_Asked_Questions#4-1_How_does_RetroShare_know_my_friend.27s_IP_address_and_port.3F_Why_don.

27t_I_need_a_static_IP_address.3F_What_is_DHT_for.3F) retrieved December 2011

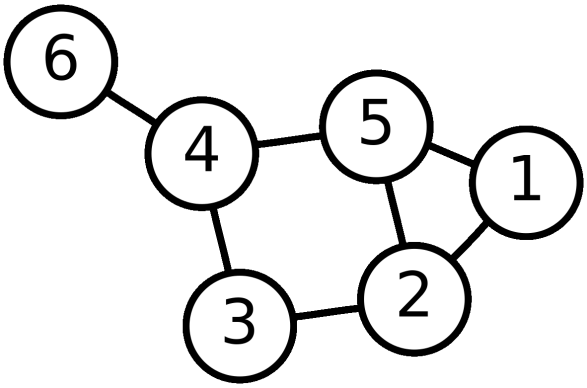
External links

- Distributed Hash Tables, Part 1 (http://linuxjournal.com/article/6797) by Brandon Wiley.
- Distributed Hash Tables links (http://deim.urv.cat/~cpairot/dhts.html) Carles Pairot's Page on DHT and P2P research
- kademlia.scs.cs.nyu.edu (http://web.archive.org/web/*/http://kademlia.scs.cs.nyu.edu/) Archive.org snapshots of kademlia.scs.cs.nyu.edu
- Hazelcast (http://hazelcast.com): open source DHT implementation
- Eng-Keong Lua; Crowcroft, Jon; Pias, Marcelo; Sharma, Ravi; Lim, Steve. *IEEE Survey on overlay network schemes*. CiteSeerX: 10.1.1.111.4197 (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.4197): covering unstructured and structured decentralized overlay networks including DHTs (Chord, Pastry, Tapestry and others).
- Mainline DHT Measurement (http://www.cs.helsinki.fi/u/jakangas/MLDHT/) at Department of Computer Science, University of Helsinki, Finland.

Graph (mathematics)

This article is about sets of vertices connected by edges. For graphs of mathematical functions, see Graph of a function. For other uses, see Graph (disambiguation).

In mathematics, and more specifically in graph theory, a **graph** is a representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by mathematical abstractions called *vertices*, and the links that connect some pairs of vertices are called *edges*. Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges. Graphs are one of the objects of study in discrete mathematics.



A drawing of a labeled graph on 6 vertices and 7 edges.

The edges may be directed or undirected. For example, if the vertices represent people at a party, and there is an edge between two people if they shake hands, then this is an undirected graph, because if person A shook hands with person B, then person B also shook hands with person A. In contrast, if there is an edge from person A to person B when person A knows of person B, then this graph is directed, because knowledge of someone is not necessarily a symmetric relation (that is, one person knowing another person does not necessarily imply the reverse; for example, many fans may know of a celebrity, but the celebrity is unlikely to know of all their fans). This latter type of graph is called a *directed* graph and the edges are called *directed edges* or *arcs*.

Vertices are also called *nodes* or *points*, and **edges** are also called *arcs* or *lines*. Graphs are the basic subject studied by graph theory. The word "graph" was first used in this sense by J.J. Sylvester in 1878.[1]

Definitions

Definitions in graph theory vary. The following are some of the more basic ways of defining graphs and related mathematical structures.

Graph

In the most common sense of the term,[2] a **graph** is an ordered pair $G = (V, E)$ comprising a set V of **vertices** or **nodes** together with a set E of **edges** or **lines**, which are 2-element subsets of V (i.e., an edge is related with two vertices, and the relation is represented as an unordered pair of the vertices with respect to the particular edge). To avoid ambiguity, this type of graph may be described precisely as undirected and simple**.**

Other senses of *graph* stem from different conceptions of the edge set. In one more generalized notion,[3] E is a set together with a relation of **incidence** that associates with each edge two vertices. In another generalized notion, E is a multiset of unordered pairs of (not necessarily distinct) vertices. Many authors call this type of object a multigraph or pseudograph.

All of these variants and others are described more fully below.

The vertices belonging to an edge are called the **ends**, **endpoints**, or **end vertices** of the edge. A vertex may exist in a graph and not belong to an edge.

V and E are usually taken to be finite, and many of the well-known results are not true (or are rather different) for **infinite graphs** because many of the arguments fail in the infinite case. The **order** of a graph is $|V|$ (the number of vertices). A graph's **size** is $|E|$, the number of edges. The **degree** of a vertex is the number of edges that connect to it, where an edge that connects to the vertex at both ends (a loop) is counted twice.
For an edge $\{u, v\}$, graph theorists usually use the somewhat shorter notation uv.

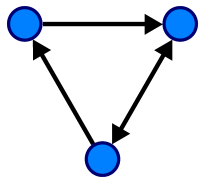
Adjacency relation

The edges E of an undirected graph G induce a symmetric binary relation ~ on V that is called the **adjacency** relation of G. Specifically, for each edge $\{u, v\}$ the vertices u and v are said to be **adjacent** to one another, which is denoted $u \sim v$.

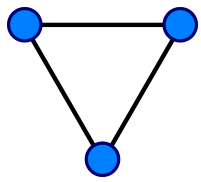
Types of graphs

Distinction in terms of the main definition

As stated above, in different contexts it may be useful to define the term *graph* with different degrees of generality. Whenever it is necessary to draw a strict distinction, the following terms are used. Most commonly, in modern texts in graph theory, unless stated otherwise, *graph* means "undirected simple finite graph" (see the definitions below).



A directed graph.



A simple undirected graph with three vertices and three edges. Each vertex has degree two, so this is also a regular graph.

Undirected graph

An undirected graph is one in which edges have no orientation. The edge (a, b) is identical to the edge (b, a), i.e., they are not ordered pairs, but sets $\{u, v\}$ (or 2-multisets) of vertices. The maximum number of edges in an undirected graph without a self-loop is $n(n - 1)/2$.

Directed graph

Main article: Directed graph

A **directed graph** or **digraph** is an ordered pair $D = (V, A)$ with

- V a set whose elements are called **vertices** or **nodes**, and
- A a set of ordered pairs of vertices, called **arcs**, **directed edges**, or **arrows**.

An arc $a = (x, y)$ is considered to be directed **from** x **to** y; y is called the **head** and x is called the **tail** of the arc; y is said to be a **direct successor** of x, and x is said to be a **direct predecessor** of y. If a path leads from x to y, then y is said to be a **successor** of x and **reachable** from x, and x is said to be a **predecessor** of y. The arc (y, x) is called the arc (x, y) **inverted**.

A directed graph D is called **symmetric** if, for every arc in D, the corresponding inverted arc also belongs to D. A symmetric loopless directed graph $D = (V, A)$ is equivalent to a simple undirected graph $G = (V, E)$, where the pairs of inverse arcs in A correspond 1-to-1 with the edges in E; thus the edges in G number $|E| = |A|/2$, or half the number of arcs in D.

A variation on this definition is the **oriented graph**, in which at most one of (x, y) and (y, x) may be arcs.

Mixed graph

Main article: Mixed graph

A **mixed graph** G is a graph in which some edges may be directed and some may be undirected. It is written as an ordered triple $G = (V, E, A)$ with V, E, and A defined as above. Directed and undirected graphs are special cases.

Multigraph

A loop is an edge (directed or undirected) which starts and ends on the same vertex; these may be permitted or not permitted according to the application. In this context, an edge with two different ends is called a **link**.

The term "multigraph" is generally understood to mean that multiple edges (and sometimes loops) are allowed. Where graphs are defined so as to *allow* loops and multiple edges, a multigraph is often defined to mean a graph *without* loops,[4] however, where graphs are defined so as to *disallow* loops and multiple edges, the term is often defined to mean a "graph" which can have both multiple edges *and* loops,[5] although many use the term "pseudograph" for this meaning.[6]

Quiver

A **quiver** or "multidigraph" is a directed graph which may have more than one arrow from a given source to a given target. A quiver may also have directed loops in it.

Simple graph

As opposed to a multigraph, a simple graph is an undirected graph that has no loops (edges connected at both ends to the same vertex) and no more than one edge between any two different vertices. In a simple graph the edges of the graph form a set (rather than a multiset) and each edge is a *distinct* pair of vertices. In a simple graph with n vertices every vertex has a degree that is less than n (the converse, however, is not true — there exist non-simple graphs with

n vertices in which every vertex has a degree smaller than n).

Weighted graph

A graph is a weighted graph if a number (weight) is assigned to each edge. Such weights might represent, for example, costs, lengths or capacities, etc. depending on the problem at hand. Some authors call such a graph a network. Weighted correlation networks can be defined by soft-thresholding the pairwise correlations among variables (e.g. gene measurements).

Half-edges, loose edges

In certain situations it can be helpful to allow edges with only one end, called **half-edges**, or no ends (**loose edges**); see for example signed graphs and biased graphs.

Important graph classes

Regular graph

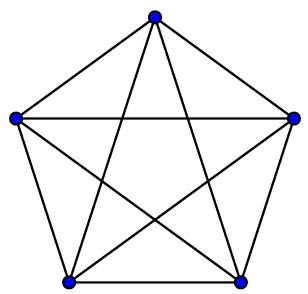
Main article: Regular graph

A regular graph is a graph where each vertex has the same number of neighbours, i.e., every vertex has the same degree or valency. A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k.

Complete graph

Main article: Complete graph

Complete graphs have the feature that each pair of vertices has an edge connecting them.



A complete graph with 5 vertices. Each vertex has an edge to every other vertex.

Finite and infinite graphs

A finite graph is a graph $G = (V, E)$ such that V and E are finite sets. An infinite graph is one with an infinite set of vertices or edges or both.

Most commonly in graph theory it is implied that the graphs discussed are finite. If the graphs are infinite, that is usually specifically stated.

Graph classes in terms of connectivity

Main article: Connectivity (graph theory)

In an undirected graph G, two vertices u and v are called **connected** if G contains a path from u to v. Otherwise, they are called **disconnected**. A graph is called **connected** if every pair of distinct vertices in the graph is connected; otherwise, it is called **disconnected**.

A graph is called ***k*-vertex-connected** or ***k*-edge-connected** if no set of $k-1$ vertices (respectively, edges) exists that, when removed, disconnects the graph. A k-vertex-connected graph is often called simply ***k*-connected**.

A directed graph is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. It is **strongly connected** or **strong** if it contains a directed path from u to v and a directed path from v to u for every pair of vertices u, v.

Category of all graphs

The category of all graphs is the slice category $\mathbf{Set} \downarrow D$ where $D : \mathbf{Set} \to \mathbf{Set}$ is the functor taking a set s to $s \times s$.

Properties of graphs

See also: Glossary of graph theory and Graph property

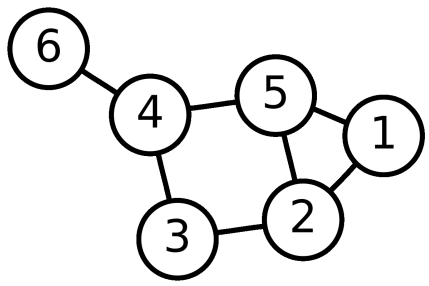
Two edges of a graph are called **adjacent** if they share a common vertex. Two arrows of a directed graph are called **consecutive** if the head of the first one is at the nock (notch end) of the second one. Similarly, two vertices are called **adjacent** if they share a common edge (**consecutive** if they are at the notch and at the head of an arrow), in which case the common edge is said to **join** the two vertices. An edge and a vertex on that edge are called **incident**.

The graph with only one vertex and no edges is called the **trivial graph**. A graph with only vertices and no edges is known as an **edgeless graph**. The graph with no vertices and no edges is sometimes called the **null graph** or **empty graph**, but the terminology is not consistent and not all mathematicians allow this object.

In a **weighted** graph or digraph, each edge is associated with some value, variously called its *cost*, *weight*, *length* or other term depending on the application; such graphs arise in many contexts, for example in optimal routing problems such as the traveling salesman problem.

Normally, the vertices of a graph, by their nature as elements of a set, are distinguishable. This kind of graph may be called **vertex-labeled**. However, for many questions it is better to treat vertices as indistinguishable; then the graph may be called **unlabeled**. (Of course, the vertices may be still distinguishable by the properties of the graph itself, e.g., by the numbers of incident edges). The same remarks apply to edges, so graphs with labeled edges are called **edge-labeled** graphs. Graphs with labels attached to edges or vertices are more generally designated as **labeled**. Consequently, graphs in which vertices are indistinguishable and edges are indistinguishable are called *unlabeled*. (Note that in the literature the term *labeled* may apply to other kinds of labeling, besides that which serves only to distinguish different vertices or edges.)

Examples



A graph with six nodes.

- The diagram at right is a graphic representation of the following graph:

  $V = \{1, 2, 3, 4, 5, 6\}$

  $E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$.
- In category theory a small category can be represented by a directed multigraph in which the objects of the category represented as vertices and the morphisms as directed edges. Then, the functors between categories induce some, but not necessarily all, of the digraph morphisms of the graph.
- In computer science, directed graphs are used to represent knowledge (e.g., Conceptual graph), finite state machines, and many other discrete structures.
- A binary relation R on a set X defines a directed graph. An element x of X is a direct predecessor of an element y of X iff xRy.
- A directed edge can model information networks such as Twitter, with one user following another [7]

Important graphs

Basic examples are:

- In a complete graph, each pair of vertices is joined by an edge; that is, the graph contains all possible edges.
- In a bipartite graph, the vertex set can be partitioned into two sets, W and X, so that no two vertices in W are adjacent and no two vertices in X are adjacent. Alternatively, it is a graph with a chromatic number of 2.
- In a complete bipartite graph, the vertex set is the union of two disjoint sets, W and X, so that every vertex in W is adjacent to every vertex in X but there are no edges within W or X.
- In a *linear graph* or path graph of length n, the vertices can be listed in order, $v_0, v_1, ..., v_n$, so that the edges are $v_{i-1}v_i$ for each $i = 1, 2, ..., n$. If a linear graph occurs as a subgraph of another graph, it is a path in that graph.
- In a cycle graph of length $n \geq 3$, vertices can be named $v_1, ..., v_n$ so that the edges are $v_{i-1}v_i$ for each $i = 2,...,n$ in addition to $v_n v_1$. Cycle graphs can be characterized as connected 2-regular graphs. If a cycle graph occurs as a subgraph of another graph, it is a *cycle* or *circuit* in that graph.
- A planar graph is a graph whose vertices and edges can be drawn in a plane such that no two of the edges intersect (i.e., *embedded* in a plane).
- A tree is a connected graph with no cycles.
- A *forest* is a graph with no cycles (i.e. the disjoint union of one or more *trees*).

More advanced kinds of graphs are:

- The Petersen graph and its generalizations
- Perfect graphs
- Cographs
- Chordal graphs
- Other graphs with large automorphism groups: vertex-transitive, arc-transitive, and distance-transitive graphs.
- Strongly regular graphs and their generalization distance-regular graphs.

Operations on graphs

Main article: Operations on graphs

There are several operations that produce new graphs from old ones, which might be classified into the following categories:

- Elementary operations, sometimes called "editing operations" on graphs, which create a new graph from the original one by a simple, local change, such as addition or deletion of a vertex or an edge, merging and splitting of vertices, etc.
- Graph rewrite operations replacing the occurrence of some pattern graph within the host graph by an instance of the corresponding replacement graph.
- Unary operations, which create a significantly new graph from the old one. Examples:
  - Line graph
  - Dual graph
  - Complement graph
- Binary operations, which create new graph from two initial graphs. Examples:
  - Disjoint union of graphs
  - Cartesian product of graphs
  - Tensor product of graphs
  - Strong product of graphs
  - Lexicographic product of graphs

Generalizations

In a hypergraph, an edge can join more than two vertices.

An undirected graph can be seen as a simplicial complex consisting of 1-simplices (the edges) and 0-simplices (the vertices). As such, complexes are generalizations of graphs since they allow for higher-dimensional simplices.

Every graph gives rise to a matroid.

In model theory, a graph is just a structure. But in that case, there is no limitation on the number of edges: it can be any cardinal number, see continuous graph.

In computational biology, power graph analysis introduces power graphs as an alternative representation of undirected graphs.

In geographic information systems, geometric networks are closely modeled after graphs, and borrow many concepts from graph theory to perform spatial analysis on road networks or utility grids.

Notes

[1] See:

- J. J. Sylvester (February 7, 1878) "Chemistry and algebra," (http://books.google.com/books?id=KcoKAAAAYAAJ&vq=Sylvester&pg=PA284#v=onepage&q&f=false) *Nature*, **17** : 284. From page 284: "Every invariant and covariant thus becomes expressible by a *graph* precisely identical with a Kekuléan diagram or chemicograph."
- J. J. Sylvester (1878) "On an application of the new atomic theory to the graphical representation of the invariants and covariants of binary quantics, — with three appendices," (http://books.google.com/books?id=1q0EAAAAYAAJ&pg=PA64#v=onepage&q&f=false) *American Journal of Mathematics, Pure and Applied*, **1** (1) : 64-90. The term "graph" first appears in this paper on page 65.

[2] See, for instance, Iyanaga and Kawada, **69 J**, p. 234 or Biggs, p. 4.

[3] See, for instance, Graham et al., p. 5.

[4] For example, see Balakrishnan, p. 1, Gross (2003), p. 4, and Zwillinger, p. 220.

[5] For example, see. Bollobás, p. 7 and Diestel, p. 25.

[6] Gross (1998), p. 3, Gross (2003), p. 205, Harary, p.10, and Zwillinger, p. 220.

[7] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF: The who-to-follow system at Twitter (http://dl.acm.org/citation.cfm?id=2488433), Proceedings of the 22nd international conference on World Wide Web

References

- Balakrishnan, V. K. (1997-02-01). *Graph Theory* (1st ed.). McGraw-Hill. ISBN 0-07-005489-4.
- Berge, Claude (1958). *Théorie des graphes et ses applications* (in French). Dunod, Paris: Collection Universitaire de Mathématiques, II. pp. viii+277. Translation: -. Dover, New York: Wiley. 2001 [1962].
- Biggs, Norman (1993). *Algebraic Graph Theory* (2nd ed.). Cambridge University Press. ISBN 0-521-45897-8.
- Bollobás, Béla (2002-08-12). *Modern Graph Theory* (1st ed.). Springer. ISBN 0-387-98488-7.
- Bang-Jensen, J.; Gutin, G. (2000). *Digraphs: Theory, Algorithms and Applications* (http://www.cs.rhul.ac.uk/books/dbook/). Springer.
- Diestel, Reinhard (2005). *Graph Theory* (http://diestel-graph-theory.com/GrTh.html) (3rd ed.). Berlin, New York: Springer-Verlag. ISBN 978-3-540-26183-4..
- Graham, R.L., Grötschel, M., and Lovász, L, ed. (1995). *Handbook of Combinatorics*. MIT Press. ISBN 0-262-07169-X.
- Gross, Jonathan L.; Yellen, Jay (1998-12-30). *Graph Theory and Its Applications*. CRC Press. ISBN 0-8493-3982-0.
- Gross, Jonathan L., & Yellen, Jay, ed. (2003-12-29). *Handbook of Graph Theory*. CRC. ISBN 1-58488-090-2.
- Harary, Frank (January 1995). *Graph Theory*. Addison Wesley Publishing Company. ISBN 0-201-41033-8.
- Iyanaga, Shôkichi; Kawada, Yukiyosi (1977). *Encyclopedic Dictionary of Mathematics*. MIT Press. ISBN 0-262-09016-3.

- Zwillinger, Daniel (2002-11-27). *CRC Standard Mathematical Tables and Formulae* (31st ed.). Chapman & Hall/CRC. ISBN 1-58488-291-3.

Further reading

- Trudeau, Richard J. (1993). *Introduction to Graph Theory* (http://store.doverpublications.com/0486678709.html) (Corrected, enlarged republication. ed.). New York: Dover Publications. ISBN 978-0-486-67870-2. Retrieved 8 August 2012.

External links

Library resources about **Graph(mathematics)**
• Resources in your library (http://tools.wmflabs.org/ftl/cgi-bin/ftl?st=wp&su=Graph+(mathematics))

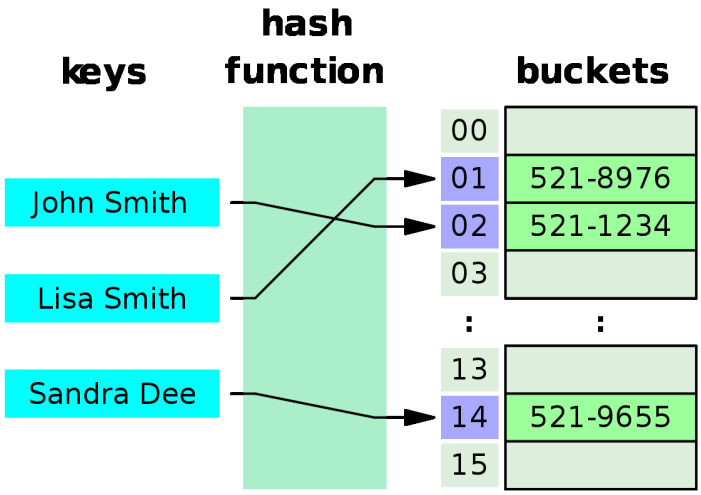
- Weisstein, Eric W., "Graph" (http://mathworld.wolfram.com/Graph.html), *MathWorld*.

Hash table

Not to be confused with Hash list or Hash tree.

Hash table		
Type	Unordered associative array	
Invented	1953	
Time complexity in big O notation		
	Average	Worst case
Space	O(*n*)	O(*n*)
Search	O(1)	O(*n*)
Insert	O(1)	O(*n*)
Delete	O(1)	O(*n*)

In computing, a **hash table** (also **hash map**) is a data structure used to implement an associative array, a structure that can map keys to values. A hash table uses a hash function to compute an *index* into an array of *buckets* or *slots*, from which the correct value can be found.



A small phone book as a hash table

Ideally, the hash function will assign each key to a unique bucket, but this situation is rarely achievable in practice (usually some keys will hash to the same bucket). Instead, most hash table designs assume that *hash collisions*—different keys that are assigned by the hash function to the same bucket—will occur and must be accommodated in some way.

In a well-dimensioned hash table, the average cost (number of instructions) for each lookup is independent of the number of elements stored in the table. Many hash table designs also allow arbitrary insertions and deletions of key-value pairs, at (amortized[1]) constant average cost per operation.

In many situations, hash tables turn out to be more efficient than search trees or any other table lookup structure. For this reason, they are widely used in many kinds of computer software, particularly for associative arrays, database indexing, caches, and sets.

Hashing

Main article: Hash function

The idea of hashing is to distribute the entries (key/value pairs) across an array of *buckets*. Given a key, the algorithm computes an *index* that suggests where the entry can be found:

```
index = f(key, array_size)
```

Often this is done in two steps:

```
hash = hashfunc(key)
index = hash % array_size
```

In this method, the *hash* is independent of the array size, and it is then *reduced* to an index (a number between `0` and `array_size - 1`) using the modulus operator (`%`).

In the case that the array size is a power of two, the remainder operation is reduced to masking, which improves speed, but can increase problems with a poor hash function.

Choosing a good hash function

A good hash function and implementation algorithm are essential for good hash table performance, but may be difficult to achieve.

A basic requirement is that the function should provide a uniform distribution of hash values. A non-uniform distribution increases the number of collisions and the cost of resolving them. Uniformity is sometimes difficult to ensure by design, but may be evaluated empirically using statistical tests, e.g. a Pearson's chi-squared test for discrete uniform distributions.

The distribution needs to be uniform only for table sizes that occur in the application. In particular, if one uses dynamic resizing with exact doubling and halving of the table size *s*, then the hash function needs to be uniform only when *s* is a power of two. On the other hand, some hashing algorithms provide uniform hashes only when *s* is a prime number.[2]

For open addressing schemes, the hash function should also avoid *clustering*, the mapping of two or more keys to consecutive slots. Such clustering may cause the lookup cost to skyrocket, even if the load factor is low and collisions are infrequent. The popular multiplicative hash is claimed to have particularly poor clustering behavior.

Cryptographic hash functions are believed to provide good hash functions for any table size *s*, either by modulo reduction or by bit masking. They may also be appropriate if there is a risk of malicious users trying to sabotage a network service by submitting requests designed to generate a large number of collisions in the server's hash tables. However, the risk of sabotage can also be avoided by cheaper methods (such as applying a secret salt to the data, or using a universal hash function).

Perfect hash function

If all keys are known ahead of time, a perfect hash function can be used to create a perfect hash table that has no collisions. If minimal perfect hashing is used, every location in the hash table can be used as well.

Perfect hashing allows for constant time lookups in the worst case. This is in contrast to most chaining and open addressing methods, where the time for lookup is low on average, but may be very large (proportional to the number of entries) for some sets of keys.

Key statistics

A critical statistic for a hash table is called the *load factor*. This is simply the number of entries divided by the number of buckets, that is, n/k where n is the number of entries and k is the number of buckets.

If the load factor is kept reasonable, the hash table should perform well, provided the hashing is good. If the load factor grows too large, the hash table will become slow, or it may fail to work (depending on the method used). The expected constant time property of a hash table assumes that the load factor is kept below some bound. For a *fixed* number of buckets, the time for a lookup grows with the number of entries and so does not achieve the desired constant time.

Second to that, one can examine the variance of number of entries per bucket. For example, two tables both have 1000 entries and 1000 buckets; one has exactly one entry in each bucket, the other has all entries in the same bucket. Clearly the hashing is not working in the second one.

A low load factor is not especially beneficial. As the load factor approaches 0, the proportion of unused areas in the hash table increases, but there is not necessarily any reduction in search cost. This results in wasted memory.

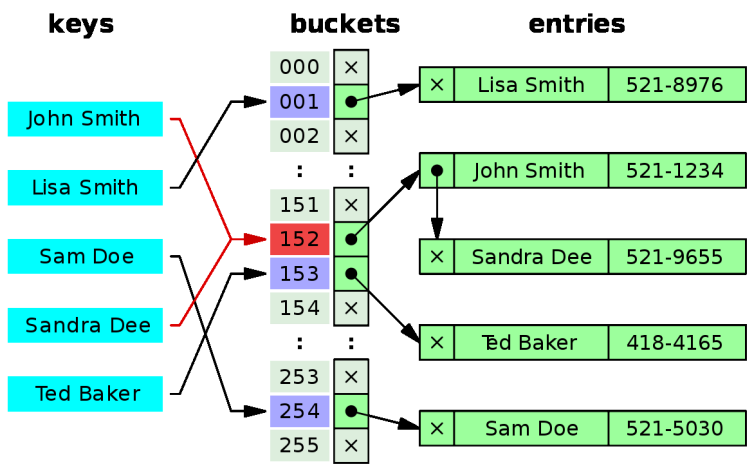
Collision resolution

Hash collisions are practically unavoidable when hashing a random subset of a large set of possible keys. For example, if 2,450 keys are hashed into a million buckets, even with a perfectly uniform random distribution, according to the birthday problem there is approximately a 95% chance of at least two of the keys being hashed to the same slot.

Therefore, most hash table implementations have some collision resolution strategy to handle such events. Some common strategies are described below. All these methods require that the keys (or pointers to them) be stored in the table, together with the associated values.

Separate chaining

In the method known as *separate chaining*, each bucket is independent, and has some sort of list of entries with the same index. The time for hash table operations is the time to find the bucket (which is constant) plus the time for the list operation. (The technique is also called *open hashing* or *closed addressing*.)



Hash collision resolved by separate chaining.

In a good hash table, each bucket has zero or one entries, and sometimes two or three, but rarely more than that. Therefore, structures that are efficient in time and space for these cases are preferred. Structures that are efficient for a fairly large number of entries are not needed or desirable. If these cases happen often, the hashing is not working well, and this needs to be fixed.

Separate chaining with linked lists

Chained hash tables with linked lists are popular because they require only basic data structures with simple algorithms, and can use simple hash functions that are unsuitable for other methods.

The cost of a table operation is that of scanning the entries of the selected bucket for the desired key. If the distribution of keys is sufficiently uniform, the *average* cost of a lookup depends only on the average number of keys per bucket—that is, on the load factor.

Chained hash tables remain effective even when the number of table entries *n* is much higher than the number of slots. Their performance degrades more gracefully (linearly) with the load factor. For example, a chained hash table with 1000 slots and 10,000 stored keys (load factor 10) is five to ten times slower than a 10,000-slot table (load factor 1); but still 1000 times faster than a plain sequential list, and possibly even faster than a balanced search tree.Wikipedia:Citation needed

For separate-chaining, the worst-case scenario is when all entries are inserted into the same bucket, in which case the hash table is ineffective and the cost is that of searching the bucket data structure. If the latter is a linear list, the lookup procedure may have to scan all its entries, so the worst-case cost is proportional to the number *n* of entries in the table.

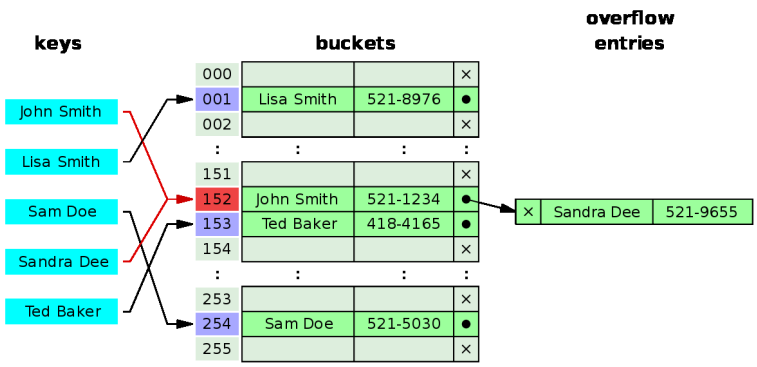
The bucket chains are often implemented as ordered lists, sorted by the key field; this choice approximately halves the average cost of unsuccessful lookups, compared to an unordered listWikipedia:Citation needed. However, if some keys are much more likely to come up than others, an unordered list with move-to-front heuristic may be more effective. More sophisticated data structures, such as balanced search trees, are worth considering only if the load factor is large (about 10 or more), or if the hash distribution is likely to be very non-uniform, or if one must guarantee good performance even in a worst-case scenario. However, using a larger table and/or a better hash function may be even more effective in those cases.

Chained hash tables also inherit the disadvantages of linked lists. When storing small keys and values, the space overhead of the `next` pointer in each entry record can be significant. An additional disadvantage is that traversing a linked list has poor cache performance, making the processor cache ineffective.

Separate chaining with list head cells

Some chaining implementations store the first record of each chain in the slot array itself. The number of pointer traversals is decreased by one for most cases. The purpose is to increase cache efficiency of hash table access.

The disadvantage is that an empty bucket takes the same space as a bucket with one entry. To save memory space, such hash tables often have about as many slots as stored entries, meaning that many slots have two or more entries.



Hash collision by separate chaining with head records in the bucket array.

Separate chaining with other structures

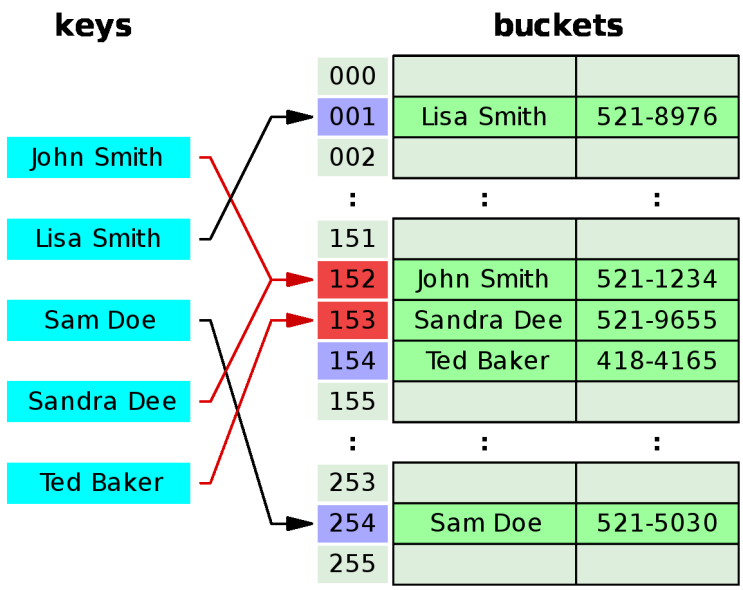
Instead of a list, one can use any other data structure that supports the required operations. For example, by using a self-balancing tree, the theoretical worst-case time of common hash table operations (insertion, deletion, lookup) can be brought down to O(log *n*) rather than O(*n*). However, this approach is only worth the trouble and extra memory cost if long delays must be avoided at all costs (e.g. in a real-time application), or if one must guard against many entries hashed to the same slot (e.g. if one expects extremely non-uniform distributions, or in the case of web sites or other publicly accessible services, which are vulnerable to malicious key distributions in requests).

The variant called array hash table uses a dynamic array to store all the entries that hash to the same slot. Each newly inserted entry gets appended to the end of the dynamic array that is assigned to the slot. The dynamic array is resized in an *exact-fit* manner, meaning it is grown only by as many bytes as needed. Alternative techniques such as growing the array by block sizes or *pages* were found to improve insertion performance, but at a cost in space. This variation makes more efficient use of CPU caching and the translation lookaside buffer (TLB), because slot entries are stored in sequential memory positions. It also dispenses with the `next` pointers that are required by linked lists, which saves space. Despite frequent array resizing, space overheads incurred by operating system such as memory fragmentation, were found to be small.

An elaboration on this approach is the so-called dynamic perfect hashing,[3] where a bucket that contains k entries is organized as a perfect hash table with k^2 slots. While it uses more memory (n^2 slots for n entries, in the worst case and $n*k$ slots in the average case), this variant has guaranteed constant worst-case lookup time, and low amortized time for insertion.

Open addressing

In another strategy, called open addressing, all entry records are stored in the bucket array itself. When a new entry has to be inserted, the buckets are examined, starting with the hashed-to slot and proceeding in some *probe sequence*, until an unoccupied slot is found. When searching for an entry, the buckets are scanned in the same sequence, until either the target record is found, or an unused array slot is found, which indicates that there is no such key in the table. The name "open addressing" refers to the fact that the location ("address") of the item is not determined by its hash value. (This method is also called **closed hashing**; it should not be confused with "open hashing" or "closed addressing" that usually mean separate chaining.)



Hash collision resolved by open addressing with linear probing (interval=1). Note that "Ted Baker" has a unique hash, but nevertheless collided with "Sandra Dee", that had previously collided with "John Smith".

Well-known probe sequences include:

- Linear probing, in which the interval between probes is fixed (usually 1)
- Quadratic probing, in which the interval between probes is increased by adding the successive outputs of a quadratic polynomial to the starting value given by the original hash computation

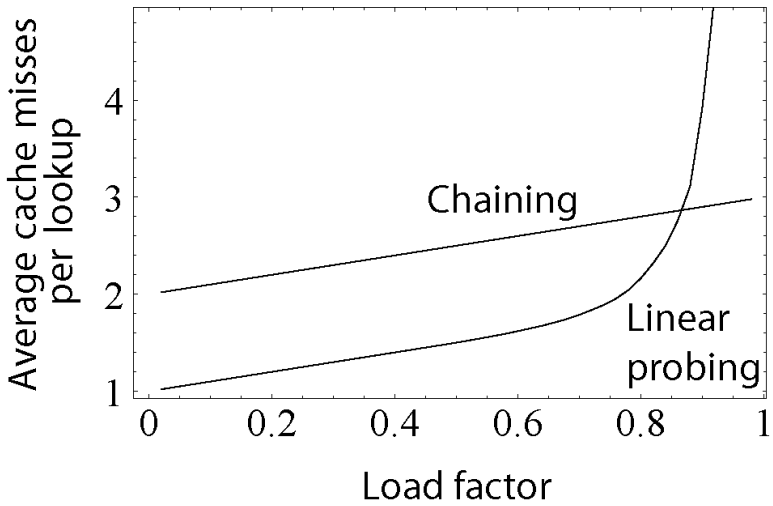
- Double hashing, in which the interval between probes is computed by another hash function

A drawback of all these open addressing schemes is that the number of stored entries cannot exceed the number of slots in the bucket array. In fact, even with good hash functions, their performance dramatically degrades when the load factor grows beyond 0.7 or so. Thus a more aggressive resize scheme is needed. Separate linking works correctly with any load factor, although performance is likely to be reasonable if it is kept below 2 or so. For many applications, these restrictions mandate the use of dynamic resizing, with its attendant costs.

Open addressing schemes also put more stringent requirements on the hash function: besides distributing the keys more uniformly over the buckets, the function must also minimize the clustering of hash values that are consecutive in the probe order. Using separate chaining, the only concern is that too many objects map to the *same* hash value; whether they are adjacent or nearby is completely irrelevant.

Open addressing only saves memory if the entries are small (less than four times the size of a pointer) and the load factor is not too small. If the load factor is close to zero (that is, there are far more buckets than stored entries), open addressing is wasteful even if each entry is just two words.

Open addressing avoids the time overhead of allocating each new entry record, and can be implemented even in the absence of a memory allocator. It also avoids the extra indirection required to access the first entry of each bucket (that is, usually the only one). It also has better locality of reference, particularly with linear probing. With small record sizes, these factors can yield better performance than chaining, particularly for lookups. Hash tables with open addressing are also easier to serialize, because they do not use pointers.



This graph compares the average number of cache misses required to look up elements in tables with chaining and linear probing. As the table passes the 80%-full mark, linear probing's performance drastically degrades.

On the other hand, normal open addressing is a poor choice for large elements, because these elements fill entire CPU cache lines (negating the cache advantage), and a large amount of space is wasted on large empty table slots. If the open addressing table only stores references to elements (external storage), it uses space comparable to chaining even for large records but loses its speed advantage.

Generally speaking, open addressing is better used for hash tables with small records that can be stored within the table (internal storage) and fit in a cache line. They are particularly suitable for elements of one word or less. If the table is expected to have a high load factor, the records are large, or the data is variable-sized, chained hash tables often perform as well or better.

Ultimately, used sensibly, any kind of hash table algorithm is usually fast *enough*; and the percentage of a calculation spent in hash table code is low. Memory usage is rarely considered excessive. Therefore, in most cases the differences between these algorithms are marginal, and other considerations typically come into play.Wikipedia:Citation needed

Coalesced hashing

A hybrid of chaining and open addressing, coalesced hashing links together chains of nodes within the table itself. Like open addressing, it achieves space usage and (somewhat diminished) cache advantages over chaining. Like chaining, it does not exhibit clustering effects; in fact, the table can be efficiently filled to a high density. Unlike chaining, it cannot have more elements than table slots.

Cuckoo hashing

Another alternative open-addressing solution is cuckoo hashing, which ensures constant lookup time in the worst case, and constant amortized time for insertions and deletions. It uses two or more hash functions, which means any key/value pair could be in two or more locations. For lookup, the first hash function is used; if the key/value is not found, then the second hash function is used, and so on. If a collision happens during insertion, then the key is re-hashed with the second hash function to map it to another bucket. If all hash functions are used and there is still a collision, then the key it collided with is removed to make space for the new key, and the old key is re-hashed with one of the other hash functions, which maps it to another bucket. If that location also results in a collision, then the process repeats until there is no collision or the process traverses all the buckets, at which point the table is resized. By combining multiple hash functions with multiple cells per bucket, very high space utilisation can be achieved.

Robin Hood hashing

One interesting variation on double-hashing collision resolution is Robin Hood hashing. The idea is that a new key may displace a key already inserted, if its probe count is larger than that of the key at the current position. The net effect of this is that it reduces worst case search times in the table. This is similar to ordered hash tables except that the criterion for bumping a key does not depend on a direct relationship between the keys. Since both the worst case and the variation in the number of probes is reduced dramatically, an interesting variation is to probe the table starting at the expected successful probe value and then expand from that position in both directions. External Robin Hashing is an extension of this algorithm where the table is stored in an external file and each table position corresponds to a fixed-sized page or bucket with B records.

2-choice hashing

2-choice hashing employs 2 different hash functions, $h_1(x)$ and $h_2(x)$, for the hash table. Both hash functions are used to compute two table locations. When an object is inserted in the table, then it is placed in the table location that contains fewer objects (with the default being the $h_1(x)$ table location if there is equality in bucket size). 2-choice hashing employs the principle of the power of two choices.

Hopscotch hashing

Another alternative open-addressing solution is hopscotch hashing, which combines the approaches of cuckoo hashing and linear probing, yet seems in general to avoid their limitations. In particular it works well even when the load factor grows beyond 0.9. The algorithm is well suited for implementing a resizable concurrent hash table.

The hopscotch hashing algorithm works by defining a neighborhood of buckets near the original hashed bucket, where a given entry is always found. Thus, search is limited to the number of entries in this neighborhood, which is logarithmic in the worst case, constant on average, and with proper alignment of the neighborhood typically requires one cache miss. When inserting an entry, one first attempts to add it to a bucket in the neighborhood. However, if all buckets in this neighborhood are occupied, the algorithm traverses buckets in sequence until an open slot (an unoccupied bucket) is found (as in linear probing). At that point, since the empty bucket is outside the neighborhood, items are repeatedly displaced in a sequence of hops. (This is similar to cuckoo hashing, but with the difference that in this case the empty slot is being moved into the neighborhood, instead of items being moved out with the hope of eventually finding an empty slot.) Each hop brings the open slot closer to the original neighborhood, without

invalidating the neighborhood property of any of the buckets along the way. In the end, the open slot has been moved into the neighborhood, and the entry being inserted can be added to it.

Dynamic resizing

To keep the load factor under a certain limit, e.g. under 3/4, many table implementations expand the table when items are inserted. For example, in Java's `HashMap` class the default load factor threshold for table expansion is 0.75.

Since buckets are usually implemented on top of a dynamic array and any constant proportion for resizing greater than 1 will keep the load factor under the desired limit, the exact choice of the constant is determined by the same space-time tradeoff as for dynamic arrays.

Resizing is accompanied by a full or incremental table **rehash** whereby existing items are mapped to new bucket locations.

To limit the proportion of memory wasted due to empty buckets, some implementations also shrink the size of the table—followed by a rehash—when items are deleted. From the point of space-time tradeoffs, this operation is similar to the deallocation in dynamic arrays.

Resizing by copying all entries

A common approach is to automatically trigger a complete resizing when the load factor exceeds some threshold r_{max}. Then a new larger table is allocated, all the entries of the old table are removed and inserted into this new table, and the old table is returned to the free storage pool. Symmetrically, when the load factor falls below a second threshold r_{min}, all entries are moved to a new smaller table.

If the table size increases or decreases by a fixed percentage at each expansion, the total cost of these resizings, amortized over all insert and delete operations, is still a constant, independent of the number of entries n and of the number m of operations performed.

For example, consider a table that was created with the minimum possible size and is doubled each time the load ratio exceeds some threshold. If m elements are inserted into that table, the total number of extra re-insertions that occur in all dynamic resizings of the table is at most $m - 1$. In other words, dynamic resizing roughly doubles the cost of each insert or delete operation.

Incremental resizing

Some hash table implementations, notably in real-time systems, cannot pay the price of enlarging the hash table all at once, because it may interrupt time-critical operations. If one cannot avoid dynamic resizing, a solution is to perform the resizing gradually:

- During the resize, allocate the new hash table, but keep the old table unchanged.
- In each lookup or delete operation, check both tables.
- Perform insertion operations only in the new table.
- At each insertion also move r elements from the old table to the new table.
- When all elements are removed from the old table, deallocate it.

To ensure that the old table is completely copied over before the new table itself needs to be enlarged, it is necessary to increase the size of the table by a factor of at least $(r + 1)/r$ during resizing.

Monotonic keys

If it is known that key values will always increase (or decrease) monotonically, then a variation of consistent hashing can be achieved by keeping a list of the single most recent key value at each hash table resize operation. Upon lookup, keys that fall in the ranges defined by these list entries are directed to the appropriate hash function—and indeed hash table—both of which can be different for each range. Since it is common to grow the overall number of entries by doubling, there will only be O(lg(N)) ranges to check, and binary search time for the redirection would be O(lg(lg(N))). As with consistent hashing, this approach guarantees that any key's hash, once issued, will never change, even when the hash table is later grown.

Other solutions

Linear hashing is a hash table algorithm that permits incremental hash table expansion. It is implemented using a single hash table, but with two possible look-up functions.

Another way to decrease the cost of table resizing is to choose a hash function in such a way that the hashes of most values do not change when the table is resized. This approach, called consistent hashing, is prevalent in disk-based and distributed hashes, where rehashing is prohibitively costly.

Performance analysis

In the simplest model, the hash function is completely unspecified and the table does not resize. For the best possible choice of hash function, a table of size k with open addressing has no collisions and holds up to k elements, with a single comparison for successful lookup, and a table of size k with chaining and n keys has the minimum $\max(0, n-k)$ collisions and $O(1 + n/k)$ comparisons for lookup. For the worst choice of hash function, every insertion causes a collision, and hash tables degenerate to linear search, with $\Omega(n)$ amortized comparisons per insertion and up to n comparisons for a successful lookup.

Adding rehashing to this model is straightforward. As in a dynamic array, geometric resizing by a factor of b implies that only n/b^i keys are inserted i or more times, so that the total number of insertions is bounded above by $bn/(b-1)$, which is $O(n)$. By using rehashing to maintain $n < k$, tables using both chaining and open addressing can have unlimited elements and perform successful lookup in a single comparison for the best choice of hash function.

In more realistic models, the hash function is a random variable over a probability distribution of hash functions, and performance is computed on average over the choice of hash function. When this distribution is uniform, the assumption is called "simple uniform hashing" and it can be shown that hashing with chaining requires $\Theta(1 + n/k)$ comparisons on average for an unsuccessful lookup, and hashing with open addressing requires $\Theta(1/(1 - n/k))$.[4] Both these bounds are constant, if we maintain $n/k < c$ using table resizing, where c is a fixed constant less than 1.

Features

Advantages

The main advantage of hash tables over other table data structures is speed. This advantage is more apparent when the number of entries is large. Hash tables are particularly efficient when the maximum number of entries can be predicted in advance, so that the bucket array can be allocated once with the optimum size and never resized.

If the set of key-value pairs is fixed and known ahead of time (so insertions and deletions are not allowed), one may reduce the average lookup cost by a careful choice of the hash function, bucket table size, and internal data structures. In particular, one may be able to devise a hash function that is collision-free, or even perfect (see below). In this case the keys need not be stored in the table.

Drawbacks

Although operations on a hash table take constant time on average, the cost of a good hash function can be significantly higher than the inner loop of the lookup algorithm for a sequential list or search tree. Thus hash tables are not effective when the number of entries is very small. (However, in some cases the high cost of computing the hash function can be mitigated by saving the hash value together with the key.)

For certain string processing applications, such as spell-checking, hash tables may be less efficient than tries, finite automata, or Judy arrays. Also, if each key is represented by a small enough number of bits, then, instead of a hash table, one may use the key directly as the index into an array of values. Note that there are no collisions in this case.

The entries stored in a hash table can be enumerated efficiently (at constant cost per entry), but only in some pseudo-random order. Therefore, there is no efficient way to locate an entry whose key is *nearest* to a given key. Listing all n entries in some specific order generally requires a separate sorting step, whose cost is proportional to $\log(n)$ per entry. In comparison, ordered search trees have lookup and insertion cost proportional to $\log(n)$, but allow finding the nearest key at about the same cost, and *ordered* enumeration of all entries at constant cost per entry.

If the keys are not stored (because the hash function is collision-free), there may be no easy way to enumerate the keys that are present in the table at any given moment.

Although the *average* cost per operation is constant and fairly small, the cost of a single operation may be quite high. In particular, if the hash table uses dynamic resizing, an insertion or deletion operation may occasionally take time proportional to the number of entries. This may be a serious drawback in real-time or interactive applications.

Hash tables in general exhibit poor locality of reference—that is, the data to be accessed is distributed seemingly at random in memory. Because hash tables cause access patterns that jump around, this can trigger microprocessor cache misses that cause long delays. Compact data structures such as arrays searched with linear search may be faster, if the table is relatively small and keys are compact. The optimal performance point varies from system to system.

Hash tables become quite inefficient when there are many collisions. While extremely uneven hash distributions are extremely unlikely to arise by chance, a malicious adversary with knowledge of the hash function may be able to supply information to a hash that creates worst-case behavior by causing excessive collisions, resulting in very poor performance, e.g. a denial of service attack.[5] In critical applications, universal hashing can be used; a data structure with better worst-case guarantees may be preferable.[6]

Uses

Associative arrays

Hash tables are commonly used to implement many types of in-memory tables. They are used to implement associative arrays (arrays whose indices are arbitrary strings or other complicated objects), especially in interpreted programming languages like AWK, Perl, and PHP.

When storing a new item into a multimap and a hash collision occurs, the multimap unconditionally stores both items.

When storing a new item into a typical associative array and a hash collision occurs, but the actual keys themselves are different, the associative array likewise stores both items. However, if the key of the new item exactly matches the key of an old item, the associative array typically erases the old item and overwrites it with the new item, so every item in the table has a unique key.

Database indexing

Hash tables may also be used as disk-based data structures and database indices (such as in dbm) although B-trees are more popular in these applications.

Caches

Hash tables can be used to implement caches, auxiliary data tables that are used to speed up the access to data that is primarily stored in slower media. In this application, hash collisions can be handled by discarding one of the two colliding entries—usually erasing the old item that is currently stored in the table and overwriting it with the new item, so every item in the table has a unique hash value.

Sets

Besides recovering the entry that has a given key, many hash table implementations can also tell whether such an entry exists or not.

Those structures can therefore be used to implement a set data structure, which merely records whether a given key belongs to a specified set of keys. In this case, the structure can be simplified by eliminating all parts that have to do with the entry values. Hashing can be used to implement both static and dynamic sets.

Object representation

Several dynamic languages, such as Perl, Python, JavaScript, and Ruby, use hash tables to implement objects. In this representation, the keys are the names of the members and methods of the object, and the values are pointers to the corresponding member or method.

Unique data representation

Hash tables can be used by some programs to avoid creating multiple character strings with the same contents. For that purpose, all strings in use by the program are stored in a single *string pool* implemented as a hash table, which is checked whenever a new string has to be created. This technique was introduced in Lisp interpreters under the name hash consing, and can be used with many other kinds of data (expression trees in a symbolic algebra system, records in a database, files in a file system, binary decision diagrams, etc.)

String interning

Main article: String interning

Implementations

In programming languages

Many programming languages provide hash table functionality, either as built-in associative arrays or as standard library modules. In C++11, for example, the `unordered_map` class provides hash tables for keys and values of arbitrary type.

In PHP 5, the Zend 2 engine uses one of the hash functions from Daniel J. Bernstein to generate the hash values used in managing the mappings of data pointers stored in a hash table. In the PHP source code, it is labelled as `DJBX33A` (Daniel J. Bernstein, Times 33 with Addition).

Python's built-in hash table implementation, in the form of the `dict` type, as well as Perl's hash type (%) are used internally to implement namespaces and therefore need to pay more attention to security, i.e. collision attacks.

In the .NET Framework, support for hash tables is provided via the non-generic `Hashtable` and generic `Dictionary` classes, which store key-value pairs, and the generic `HashSet` class, which stores only values.

Independent packages

- SparseHash [7] (formerly Google SparseHash) An extremely memory-efficient hash_map implementation, with only 2 bits/entry of overhead. The SparseHash library has several C++ hash map implementations with different performance characteristics, including one that optimizes for memory use and another that optimizes for speed.
- SunriseDD [8] An open source C library for hash table storage of arbitrary data objects with lock-free lookups, built-in reference counting and guaranteed order iteration. The library can participate in external reference counting systems or use its own built-in reference counting. It comes with a variety of hash functions and allows the use of runtime supplied hash functions via callback mechanism. Source code is well documented.
- uthash [9] This is an easy-to-use hash table for C structures.

History

The idea of hashing arose independently in different places. In January 1953, H. P. Luhn wrote an internal IBM memorandum that used hashing with chaining. G. N. Amdahl, E. M. Boehme, N. Rochester, and Arthur Samuel implemented a program using hashing at about the same time. Open addressing with linear probing (relatively prime stepping) is credited to Amdahl, but Ershov (in Russia) had the same idea.

References

[1] Charles E. Leiserson, *Amortized Algorithms, Table Doubling, Potential Method* (http://videolectures.net/mit6046jf05_leiserson_lec13/) Lecture 13, course MIT 6.046J/18.410J Introduction to Algorithms—Fall 2005

[2] Thomas Wang (1997), Prime Double Hash Table (http://www.concentric.net/~Ttwang/tech/primehash.htm). Retrieved April 27, 2012

[3] Erik Demaine, Jeff Lind. 6.897: Advanced Data Structures. MIT Computer Science and Artificial Intelligence Laboratory. Spring 2003. http://courses.csail.mit.edu/6.897/spring03/scribe_notes/L2/lecture2.pdf

[4] Doug Dunham. CS 4521 Lecture Notes (http://www.duluth.umn.edu/~ddunham/cs4521s09/notes/ch11.txt). University of Minnesota Duluth. Theorems 11.2, 11.6. Last modified April 21, 2009.

[5] Alexander Klink and Julian Wälde's *Efficient Denial of Service Attacks on Web Application Platforms* (http://events.ccc.de/congress/2011/Fahrplan/attachments/2007_28C3_Effective_DoS_on_web_application_platforms.pdf), December 28, 2011, 28th Chaos Communication Congress. Berlin, Germany.

[6] Crosby and Wallach's *Denial of Service via Algorithmic Complexity Attacks* (http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach_UsenixSec2003.pdf).

[7] http://code.google.com/p/sparsehash/

[8] http://www.sunrisetel.net/software/devtools/sunrise-data-dictionary.shtml

[9] http://uthash.sourceforge.net/

Further reading

- Tamassia, Roberto; Goodrich, Michael T. (2006). "Chapter Nine: Maps and Dictionaries". *Data structures and algorithms in Java : [updated for Java 5.0]* (4th ed.). Hoboken, NJ: Wiley. pp. 369–418. ISBN 0-471-73884-0.
- McKenzie, B. J.; Harries, R.; Bell, T. (Feb 1990). "Selecting a hashing algorithm". *Software Practice & Experience* **20** (2): 209–224.

External links

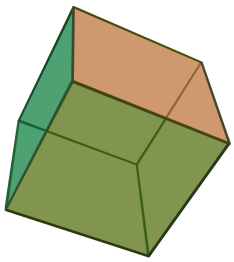
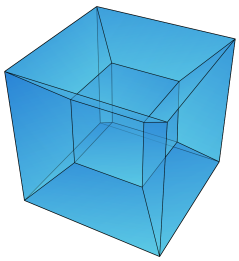
- A Hash Function for Hash Table Lookup (http://www.burtleburtle.net/bob/hash/doobs.html) by Bob Jenkins.
- Hash Tables (http://www.sparknotes.com/cs/searching/hashtables/summary.html) by SparkNotes—explanation using C
- Hash functions (http://www.azillionmonkeys.com/qed/hash.html) by Paul Hsieh

- Design of Compact and Efficient Hash Tables for Java (http://blog.griddynamics.com/2011/03/ultimate-sets-and-maps-for-java-part-i.html) link not working
- Libhashish (http://libhashish.sourceforge.net/) hash library
- NIST entry on hash tables (http://www.nist.gov/dads/HTML/hashtab.html)
- Open addressing hash table removal algorithm from ICI programming language, *ici_set_unassign* in set.c (http://ici.cvs.sourceforge.net/ici/ici/set.c?view=markup) (and other occurrences, with permission).
- A basic explanation of how the hash table works by Reliable Software (http://www.relisoft.com/book/lang/pointer/8hash.html)
- Lecture on Hash Tables (http://compgeom.cs.uiuc.edu/~jeffe/teaching/373/notes/06-hashing.pdf)
- Hash-tables in C (http://task3.cc/308/hash-maps-with-linear-probing-and-separate-chaining/)—two simple and clear examples of hash tables implementation in C with linear probing and chaining
- Open Data Structures - Chapter 5 - Hash Tables (http://opendatastructures.org/versions/edition-0.1e/ods-java/5_Hash_Tables.html)
- MIT's Introduction to Algorithms: Hashing 1 (http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-7-hashing-hash-functions/) MIT OCW lecture Video
- MIT's Introduction to Algorithms: Hashing 2 (http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-8-universal-hashing-perfect-hashing/) MIT OCW lecture Video
- Java HashMap example (http://talkera.org.cp-in-1.webhostbox.net/wp/?p=45)
- How to sort a HashMap (Java) and keep the duplicate entries (http://www.lampos.net/sort-hashmap)

Hypercube

This article is about the mathematical concept. For the film, see Cube 2: Hypercube. For the computer architecture, see Connection Machine.

Perspective projections

	
Cube (3-cube)	**Tesseract (4-cube)**

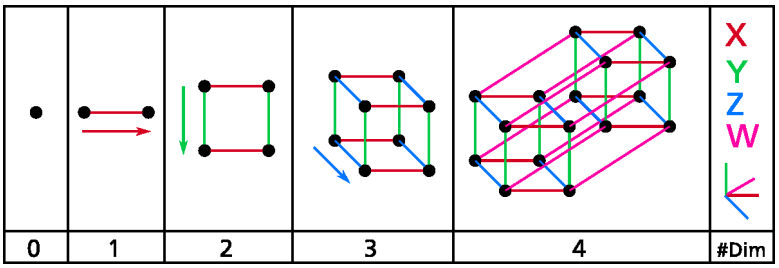
In geometry, a **hypercube** is an *n*-dimensional analogue of a square (*n* = 2) and a cube (*n* = 3). It is a closed, compact, convex figure whose 1-skeleton consists of groups of opposite parallel line segments aligned in each of the space's dimensions, perpendicular to each other and of the same length. A unit hypercube's longest diagonal in n-dimensions is equal to $\sqrt{n}$.

An *n*-dimensional hypercube is also called an **n-cube** or an **n-dimensional cube**. The term "measure polytope" is also used, notably in the work of H.S.M. Coxeter (originally from Elte, 1912[1]), but it has now been superseded.

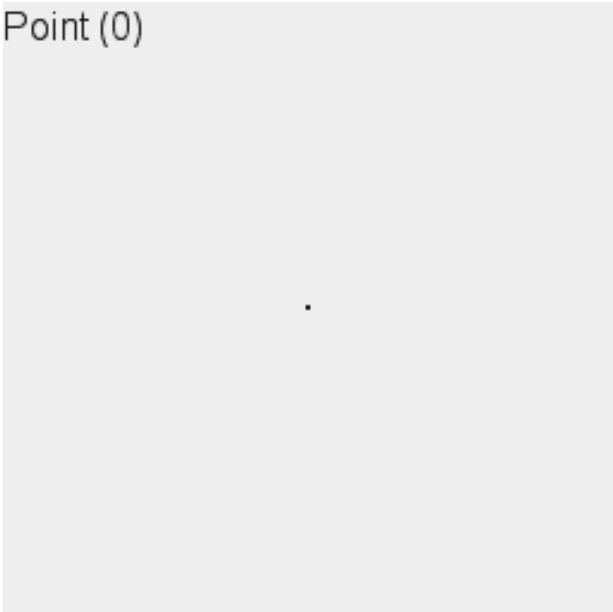
The hypercube is the special case of a hyperrectangle (also called an *orthotope*).

A **unit hypercube** is a hypercube whose side has length one unit. Often, the hypercube whose corners (or **vertices**) are the 2^n points in R^n with coordinates equal to 0 or 1 is called **"the" unit hypercube**.

Construction



A diagram showing how to create a tesseract from a point.



An animation showing how to create a tesseract from a point.

0 – A point is a hypercube of dimension zero.

1 – If one moves this point one unit length, it will sweep out a line segment, which is a unit hypercube of dimension one.

2 – If one moves this line segment its length in a perpendicular direction from itself; it sweeps out a 2-dimensional square.

3 – If one moves the square one unit length in the direction perpendicular to the plane it lies on, it will generate a 3-dimensional cube.

4 – If one moves the cube one unit length into the fourth dimension, it generates a 4-dimensional unit hypercube (a unit tesseract).

This can be generalized to any number of dimensions. This process of sweeping out volumes can be formalized mathematically as a Minkowski sum: the *d*-dimensional hypercube is the Minkowski sum of *d* mutually perpendicular unit-length line segments, and is therefore an example of a zonotope.

The 1-skeleton of a hypercube is a hypercube graph.

Coordinates

A unit hypercube of *n* dimensions is the convex hull of the points given by all sign permutations of the Cartesian coordinates $\left(\pm\frac{1}{2}, \pm\frac{1}{2}, \cdots, \pm\frac{1}{2}\right)$. It has an edge length of 1 and an *n*-dimensional volume of 1.

An *n*-dimensional hypercube is also often regarded as the convex hull of all sign permutations of the coordinates $(\pm 1, \pm 1, \cdots, \pm 1)$. This form is often chosen due to ease of writing out the coordinates. Its edge length is 2, and its *n*-dimensional volume is 2^n.

Elements

Every n-cube of n > 0 is composed of elements, or n-cubes of a lower dimension, on the (n-1)-dimensional surface on the parent hypercube. A side is any element of (n-1) dimension of the parent hypercube. A hypercube of dimension n has 2n sides (a 1-dimensional line has 2 end points; a 2-dimensional square has 4 sides or edges; a 3-dimensional cube has 6 2-dimensional faces; a 4-dimensional tesseract has 8 cells). The number of vertices (points) of a hypercube is 2^n (a cube has 2^3 vertices, for instance).

A simple formula to calculate the number of *"n-2"*-faces in an *n*-dimensional hypercube is: $2n^2 - 2n$

The number of *m*-dimensional hypercubes (just referred to as m-cube from here on) on the boundary of an *n*-cube is

$$E_{m,n} = 2^{n-m}\binom{n}{m}, \quad \text{where } \binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$ and *n*! denotes the factorial of *n*.

For example, the boundary of a 4-cube (n=4) contains 8 cubes (3-cubes), 24 squares (2-cubes), 32 lines (1-cubes) and 16 vertices (0-cubes).

This identity can be proved by combinatorial arguments; each of the 2^n vertices defines a vertex in a m -dimensional boundary. There are $\binom{n}{m}$ ways of choosing which lines ("sides") that defines the subspace that the boundary is in. But, each side is counted 2^m times since it has that many vertices, we need to divide with this number.

This identity can also be used to generate the formula for the n-dimensional cube surface area. The surface area of a hypercube is: $2ns^{n-1}$.

These numbers can also be generated by the linear recurrence relation

$E_{m,n} = 2E_{m,n-1} + E_{m-1,n-1}$, with $E_{0,0} = 1$, and undefined elements (where $m < n$, $m < 0$, or $n < 0$) = 0.

For example, extending a square via its 4 vertices adds one extra line (edge) per vertex, and also adds the final second square, to form a cube, giving $E_{1,3}$= 12 lines in total.

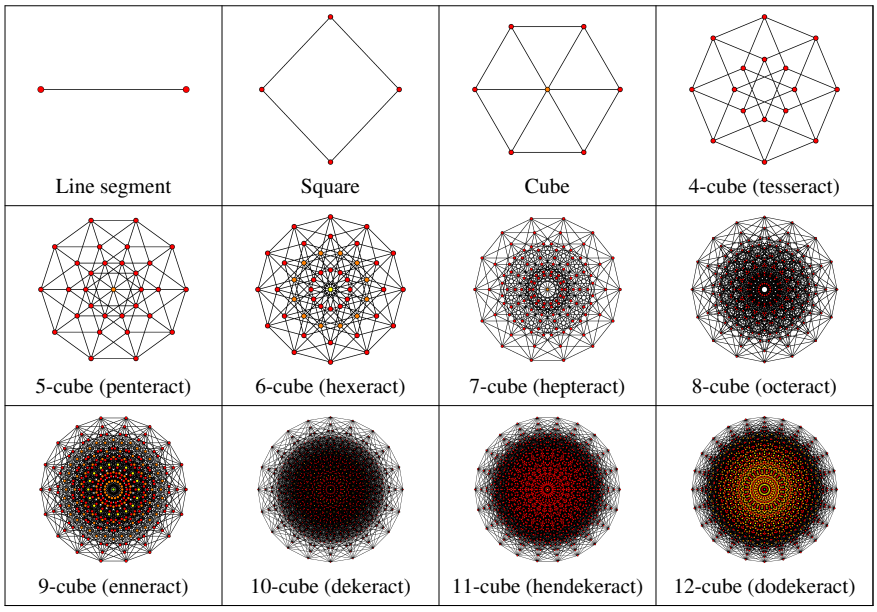
			m	0	1	2	3	4	5	6	7	8	9	10
n	**γ_n**	**n-cube**	**Names** **Schläfli symbol** **Coxeter-Dynkin**	**Vertices**	**Edges**	**Faces**	**Cells** **(3-faces)**	***4*-faces**	***5*-faces**	***6*-faces**	***7*-faces**	***8*-faces**	***9*-faces**	***10*-faces**
0	**γ_0**	0-cube	Point -	1										
1	**γ_1**	1-cube	Line segment {} ◉	2	1									
2	**γ_2**	2-cube	Square **Tetragon** {4} ◉$_{\overline{4}}$●	4	4	1								
3	**γ_3**	3-cube	Cube **Hexahedron** {4,3} ◉$_{\overline{4}}$●−●	8	12	6	1							
4	**γ_4**	4-cube	Tesseract **Octachoron** {4,3,3} ◉$_{\overline{4}}$●−●−●	16	32	24	8	1						

5	γ_5	5-cube	Penteract **Decateron** {4,3,3,3}	32	80	80	40	10	1					
6	γ_6	6-cube	Hexeract **Dodecapeton** {4,3,3,3,3}	64	192	240	160	60	12	1				
7	γ_7	7-cube	Hepteract **Tetradeca-7-tope** {4,3,3,3,3,3}	128	448	672	560	280	84	14	1			
8	γ_8	8-cube	Octeract **Hexadeca-8-tope** {4,3,3,3,3,3,3}	256	1024	1792	1792	1120	448	112	16	1		
9	γ_9	9-cube	Enneract **Octadeca-9-tope** {4,3,3,3,3,3,3,3}	512	2304	4608	5376	4032	2016	672	144	18	1	
10	γ_{10}	10-cube	Dekeract **icosa-10-tope** {4,3,3,3,3,3,3,3,3}	1024	5120	11520	15360	13440	8064	3360	960	180	20	1

Graphs

An **n-cube** can be projected inside a regular *2n*-gonal polygon by a skew orthogonal projection, shown here from the line segment to the 12-cube.

Petrie polygon Orthographic projections

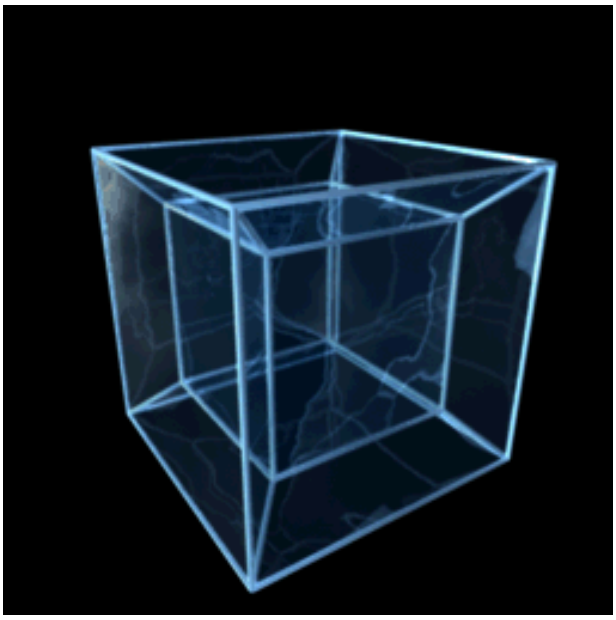


Related families of polytopes

The hypercubes are one of the few families of regular polytopes that are represented in any number of dimensions.

The **hypercube (offset)** family is one of three regular polytope families, labeled by Coxeter as γ_n. The other two are the hypercube dual family, the **cross-polytopes**, labeled as β_n, and the **simplices**, labeled as α_n. A fourth family, the infinite tessellations of hypercubes, he labeled as δ_n.

Another related family of semiregular and uniform polytopes is the **demihypercubes**, which are constructed from hypercubes with alternate vertices deleted and simplex facets added in the gaps, labeled as $h\gamma_n$.



Projection of a rotating tesseract.

Relation to *n*-simplices

The graph of the *n*-hypercube's edges is isomorphic to the Hasse diagram of the (*n*-1)-simplex's face lattice. This can be seen by orienting the *n*-hypercube so that two opposite vertices lie vertically, corresponding to the (*n*-1)-simplex itself and the null polytope, respectively. Each vertex connected to the top vertex then uniquely maps to one of the (*n*-1)-simplex's facets (*n*-2 faces), and each vertex connected to those vertices maps to one of the simplex's *n*-3 faces, and so forth, and the vertices connected to the bottom vertex map to the simplex's vertices.

This relation may be used to generate the face lattice of an (*n-1*)-simplex efficiently, since face lattice enumeration algorithms applicable to general polytopes are more computationally expensive.

Notes

[1] Chapter IV, five dimensional semiregular polytope (http://www.amazon.com/Semiregular-Polytopes-Hyperspaces-Emanuel-Lodewijk/dp/141817968X)

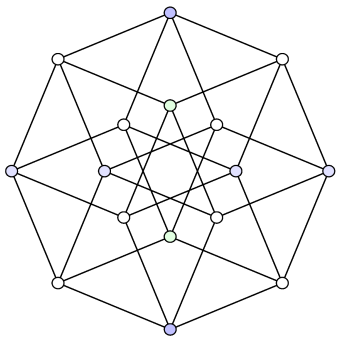
References

- Bowen, J. P. (April 1982). "Hypercubes" (https://web.archive.org/web/20080630081518/www.jpbowen.com/publications/ndcubes.html). *Practical Computing* **5** (4): 97–99.
- Coxeter, H. S. M. (1973). *Regular Polytopes* (3rd ed.). Dover. p. 123. ISBN 0-486-61480-8. p. 296, Table I (iii): Regular Polytopes, three regular polytopes in *n* dimensions (*n* ≥ 5)
- Hill, Frederick J.; Gerald R. Peterson. *Introduction to Switching Theory and Logical Design: Second Edition*. NY: John Wiley & Sons. ISBN 0-471-39882-9. Cf Chapter 7.1 "Cubical Representation of Boolean Functions" wherein the notion of "hypercube" is introduced as a means of demonstrating a distance-1 code (Gray code) as the vertices of a hypercube, and then the hypercube with its vertices so labelled is squashed into two dimensions to form either a Veitch diagram or Karnaugh map.

External links

- Weisstein, Eric W., "Hypercube" (http://mathworld.wolfram.com/Hypercube.html), *MathWorld*.
- Weisstein, Eric W., "Hypercube graphs" (http://mathworld.wolfram.com/HypercubeGraph.html), *MathWorld*.
- Olshevsky, George, *Measure polytope* (http://web.archive.org/web/20070204075028/members.aol.com/Polycell/glossary.html#Measure) at *Glossary for Hyperspace*.
- www.4d-screen.de (http://www.4d-screen.de/related-space/) (Rotation of 4D – 7D-Cube)
- *Rotating a Hypercube* (http://demonstrations.wolfram.com/RotatingAHypercube/) by Enrique Zeleny, Wolfram Demonstrations Project.
- Stereoscopic Animated Hypercube (http://dogfeathers.com/java/hyprcube.html)
- Rudy Rucker and Farideh Dormishian's Hypercube Downloads (http://www.cs.sjsu.edu/~rucker/hypercube.htm)

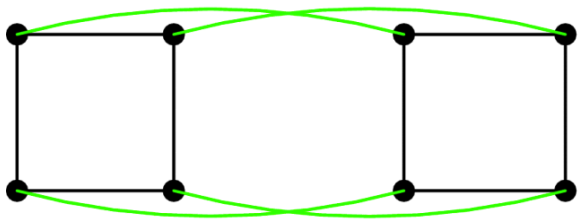
Hypercube graph

Hypercube graph	
 The hypercube graph Q_4	
Vertices	2^n
Edges	$2^{n-1}n$
Diameter	n
Girth	4 if $n \geq 2$
Automorphisms	$n!\ 2^n$
Chromatic number	2
Spectrum	$\{(n-2k)^{\binom{n}{k}}; k = 0, \ldots, n\}$
Properties	Symmetric Distance regular Unit distance Hamiltonian Bipartite
Notation	Q_n
• v • t • e [1]	

In graph theory, the **hypercube graph** Q_n is a regular graph with 2^n vertices, $2^{n-1}n$ edges, and n edges touching each vertex. It can be obtained as the one-dimensional skeleton of the geometric hypercube; for instance, Q_3 is the graph formed by the 8 vertices and 12 edges of a three-dimensional cube. Alternatively, it can be obtained from the family of subsets of a set with n elements, by making a vertex for each possible subset and joining two vertices by an edge whenever the corresponding subsets differ in a single element.

Hypercube graphs should not be confused with cubic graphs, which are graphs that have exactly three edges touching each vertex. The only hypercube that is a cubic graph is Q_3.

Construction



Construction of Q_3 by connecting pairs of corresponding vertices in two copies of Q_2

The hypercube graph Q_n may be constructed from the family of subsets of a set with n elements, by making a vertex for each possible subset and joining two vertices by an edge whenever the corresponding subsets differ in a single element. Equivalently, it may be constructed using 2^n vertices labeled with n-bit binary numbers and connecting two vertices by an edge whenever the Hamming distance of their labels is 1. These two constructions are closely related: a binary number may be interpreted as a set (the set of positions where it has a 1 digit), and two such sets differ in a single element whenever the corresponding two binary numbers have Hamming distance 1.

Alternatively, Q_{n+1} may be constructed from the disjoint union of two hypercubes Q_n, by adding an edge from each vertex in one copy of Q_n to the corresponding vertex in the other copy, as shown in the figure. The joining edges form a perfect matching.

Another definition of Q_n is the Cartesian product of n two-vertex complete graphs K_2.

Examples

The graph Q_0 consists of a single vertex, while Q_1 is the complete graph on two vertices and Q_2 is a cycle of length 4.

The graph Q_3 is the 1-skeleton of a cube, a planar graph with eight vertices and twelve edges.

The graph Q_4 is the Levi graph of the Möbius configuration.

Properties

Bipartiteness

Every hypercube graph is bipartite: it can be colored with only two colors. The two colors of this coloring may be found from the subset construction of hypercube graphs, by giving one color to the subsets that have an even number of elements and the other color to the subsets with an odd number of elements.

Hamiltonicity

Every hypercube Q_n with $n > 1$ has a Hamiltonian cycle, a cycle that visits each vertex exactly once. Additionally, a Hamiltonian path exists between two vertices u,v if and only if have different colors in a 2-coloring of the graph. Both facts are easy to prove using the principle of induction on the dimension of the hypercube, and the construction of the hypercube graph by joining two smaller hypercubes with a matching.

Hamiltonicity of the hypercube is tightly related to the theory of Gray codes. More precisely there is a bijective correspondence between the set of n-bit cyclic Gray codes and the set of Hamiltonian cycles in the hypercube Q_n. An analogous property holds for acyclic n-bit Gray codes and Hamiltonian paths.

A lesser known fact is that every perfect matching in the hypercube extends to a Hamiltonian cycle. The question whether every matching extends to a Hamiltonian cycle remains an open problem.[2]

Other properties

The hypercube graph Q_n (n > 1) :

- is the Hasse diagram of a finite Boolean algebra.
- is a median graph. Every median graph is an isometric subgraph of a hypercube, and can be formed as a retraction of a hypercube.
- has more than 2^2n-2 perfect matchings. (this is another consequence that follows easily from the inductive construction.)
- is arc transitive and symmetric. The symmetries of hypercube graphs can be represented as signed permutations.
- contains all the cycles of length 4, 6, ..., 2^n and is thus a bipancyclic graph.
- can be drawn as a unit distance graph in the Euclidean plane by choosing a unit vector for each set element and placing each vertex corresponding to a set *S* at the sum of the vectors in *S*.
- is a *n*-vertex-connected graph, by Balinski's theorem
- is planar (can be drawn with no crossings) if and only if **n** ≤ 3. For larger values of *n*, the hypercube has genus $(n-4)2^{n-3}+1$.
- has exactly $2^{2^n-n-1}\prod_{k=2}^{n} k^{\binom{n}{k}}$ spanning trees.
- The family Q_n (n > 1) is a Lévy family of graphs
- The achromatic number of Q_n is known to be proportional to $\sqrt{n2^n}$, but the constant of proportionality is not known precisely.
- The bandwidth of Q_n is exactly $\sum_{i=0}^{n}\binom{n}{\lfloor n/2\rfloor}$.[3]
- The eigenvalues of the adjacency matrix are (-n,-n+2,-n+4,...,n-4,n-2,n) and the eigenvalues of its Laplacian are (0,2,...,2n). The k-th eigenvalue has multiplicity $\binom{n}{k}$ in both cases.
- The isoperimetric number is h(G)=1

Problems

The problem of finding the longest path or cycle that is an induced subgraph of a given hypercube graph is known as the snake-in-the-box problem.

Szymanski's conjecture concerns the suitability of a hypercube as an network topology for communications. It states that, no matter how one chooses a permutation connecting each hypercube vertex to another vertex with which it should be connected, there is always a way to connect these pairs of vertices by paths that do not share any directed edge.

Notes

[1] http://en.wikipedia.org/w/index.php?title=Template:Infobox_graph&action=edit

[2] Ruskey, F. and Savage, C. Matchings extend to Hamiltonian cycles in hypercubes (http://garden.irmacs.sfu.ca/?q=op/matchings_extends_to_hamilton_cycles_in_hypercubes) on Open Problem Garden. 2007.

[3] Optimal Numberings and Isoperimetric Problems on Graphs, L.H. Harper, Journal of Combinatorial Theory, 1, 385–393,

References

- Harary, F.; Hayes, J. P.; Wu, H.-J. (1988), "A survey of the theory of hypercube graphs", *Computers & Mathematics with Applications* **15** (4): 277–289, doi: 10.1016/0898-1221(88)90213-1 (http://dx.doi.org/10.1016/0898-1221(88)90213-1).

Kademlia

Kademlia is a distributed hash table for decentralized peer-to-peer computer networks designed by Petar Maymounkov and David Mazières in 2002.[1][2] It specifies the structure of the network and the exchange of information through node lookups. Kademlia nodes communicate among themselves using UDP. A virtual or overlay network is formed by the participant nodes. Each node is identified by a number or *node ID*. The *node ID* serves not only as identification, but the Kademlia algorithm uses the *node ID* to locate values (usually file hashes or keywords). In fact, the *node ID* provides a direct map to file hashes and that node stores information on where to obtain the file or resource.

When searching for some value, the algorithm needs to know the associated key and explores the network in several steps. Each step will find nodes that are closer to the key until the contacted node returns the value or no more closer nodes are found. This is very efficient: Like many other DHTs, Kademlia contacts only $O(\log(n))$ nodes during the search out of a total of n nodes in the system.

Further advantages are found particularly in the decentralized structure, which increases the resistance against a denial of service attack. Even if a whole set of nodes is flooded, this will have limited effect on network availability, since the network will recover itself by knitting the network around these "holes".

System details

The first generation peer-to-peer file sharing networks, such as Napster, relied on a central database to co-ordinate look ups on the network. Second generation peer-to-peer networks, such as Gnutella, used flooding to locate files, searching every node on the network. Third generation peer-to-peer networks use Distributed hash tables to look up files in the network. *Distributed hash tables* store resource **locations** throughout the network. A major criterion for these protocols is locating the desired nodes quickly.

Kademlia uses a "distance" calculation between two nodes. This distance is computed as the exclusive or of the two node IDs, taking the result as an integer number. Keys and Node IDs have the same format and length, so distance can be calculated among them in exactly the same way. The node ID is typically a large random number that is chosen with the goal of being unique for a particular node (see UUID). It can and does happen that geographically widely separated nodes—from Germany and Australia, for instance—can be "neighbors" if they have chosen similar random node IDs.

Exclusive Or was chosen because it shares some properties with the geometric distance formula. Specifically:

- the distance between a node and itself is zero
- it is symmetric: the "distance" calculated from A to B and from B to A are the same
- it follows the triangle inequality: given A, B and C are vertices(points) of a triangle, then the distance from A to B is shorter than (or equal to) the sum of the distance from A to C and the distance from C to B.

These properties provide most of the important behaviors of measuring a real distance with a significantly lower amount of computation to calculate it.

Each Kademlia search iteration comes one bit closer to the target. A **basic** Kademlia network with 2^n nodes will only take *n* steps (in the worst case) to find that node.

Routing tables

This section is simplified to use a single bit, see the section accelerated lookups for more information on real routing tables.

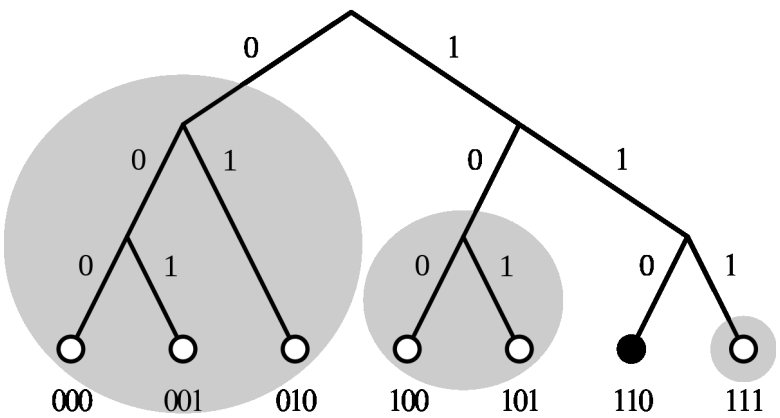
Kademlia routing tables consist of a *list* for each bit of the node ID. (e.g. if a node ID consists of 128 bits, a node will keep 128 such *lists*.) A list has many entries. Every entry in a *list* holds the necessary data to locate another node. The data in each *list* entry is typically the *IP address*, *port*, and *node ID* of another node. Every *list* corresponds to a specific distance from the node. Nodes that can go in the nth *list* must have a differing n^{th} bit from the node's ID; the first n-1 bits of the candidate ID must match those of the node's ID. This means that it is very easy to fill the first *list* as 1/2 of the nodes in the network are far away candidates. The next *list* can use only 1/4 of the nodes in the network (one bit closer than the first), etc.

With an ID of 128 bits, every node in the network will classify other nodes in one of 128 different distances, one specific distance per bit.

As nodes are encountered on the network, they are added to the *lists*. This includes store and retrieval operations and even when helping other nodes to find a key. Every node encountered will be considered for inclusion in the *lists*. Therefore the knowledge that a node has of the network is very dynamic. This keeps the network constantly updated and adds resilience to failures or attacks.

In the Kademlia literature, the *lists* are referred to as *k-buckets*. *k* is a system wide number, like 20. Every k-bucket is a *list* having up to *k* entries inside; i.e. for a network with k=20, each node will have *lists* containing up to 20 nodes for a particular bit (a particular distance from itself).

Since the possible nodes for each *k-bucket* decreases quickly (because there will be very few nodes that are that close), the lower bit *k-buckets* will fully map all nodes in that section of the network. Since the quantity of possible IDs is much larger than any node population can ever be, some of the k-buckets corresponding to very short distances will remain empty.



Network partition for node **110**

Consider the simple network to the right. The network size is 2^3 or eight maximum keys and nodes. There are seven nodes participating; the small circles at the bottom. The node under consideration is node six (binary 110) in black. There are three *k-buckets* for each node in this network. Nodes zero, one and two (binary 000, 001, and 010) are candidates for the farthest *k-bucket* . Node three (binary 011, not shown) is not participating in the network. In the middle *k-bucket*, nodes four and five (binary 100 and 101) are placed. Finally, the third *k-bucket* can only contain node seven (binary 111). Each of the three *k-buckets* is enclosed in a gray circle. If the size of the *k-bucket* was two, then the farthest *2-bucket* can only contain two of the three nodes. For example if node six has node one and two in the farthest 2-bucket, it would have

to request a node ID lookup to these nodes to find the location (ip address) of node zero. Each node *knows* its neighbourhood well and has contact with a few nodes far away which can help locate other nodes far away.

It is known that nodes that have been connected for a long time in a network will probably remain connected for a long time in the future.[3][4] Because of this statistical distribution, Kademlia selects long connected nodes to remain stored in the k-buckets. This increases the number of known valid nodes at some time in the future and provides for a more stable network.

When a *k-bucket* is full and a new node is discovered for that *k-bucket*, the least recently seen node in the *k-bucket* is PINGed. If the node is found to be still alive, the new node is place in a secondary list, a replacement cache. The replacement cache is used only if a node in the *k-bucket* stops responding. In other words: new nodes are used only when older nodes disappear.

Protocol messages

Kademlia has four messages.

- PING — used to verify that a node is still alive.
- STORE — Stores a (key, value) pair in one node.
- FIND_NODE — The recipient of the request will return the k nodes in his own buckets that are the closest ones to the requested key.
- FIND_VALUE — Same as FIND_NODE, but if the recipient of the request has the requested key in its store, it will return the corresponding value.

Each RPC message includes a random value from the initiator. This ensures that when the response is received it corresponds to the request previously sent. (see Magic cookie)

Locating nodes

Node lookups can proceed asynchronously. The quantity of simultaneous lookups is denoted by α and is typically three. A node initiates a FIND_NODE request by querying to the α nodes in its own *k-buckets* that are the closest ones to the desired key. When these recipient nodes receive the request, they will look in their *k-buckets* and return the *k* closest nodes to the desired key that they know. The requester will update a results list with the results (node ID's) he receives, keeping the *k* best ones (the k nodes that are closer to the searched key) that respond to queries. Then the requester will select these *k* best results and issue the request to them, and iterate this process again and again. Because every node has a better knowledge of his own surroundings than any other node has, the received results will be other nodes that are every time closer and closer to the searched key. The iterations continue until no nodes are returned that are closer than the best previous results. When the iterations stop, the best k nodes in the results list are the ones in the whole network that are the closest to the desired key.

The node information can be augmented with round trip times, or RTT. This information will be used to choose a time-out specific for every consulted node. When a query times out, another query can be initiated, never surpassing α queries at the same time.

Locating resources

Information is located by mapping it to a key. A hash is typically used for the map. The storer nodes will have information due to a previous STORE message. Locating a value follows the same procedure as locating the closest nodes to a key, except the search terminates when a node has the requested value in his store and returns this value.

The values are stored at several nodes (k of them) to allow for nodes to come and go and still have the value available in some node. Periodically, a node that stores a value will explore the network to find the k nodes that are close to the key value and replicate the value onto them. This compensates for disappeared nodes.

Also, for popular values that might have many requests, the load in the storer nodes is diminished by having a retriever store this value in some node near, but outside of, the k closest ones. This new storing is called a cache. In this way the value is stored farther and farther away from the key, depending on the quantity of requests. This allows popular searches to find a storer more quickly. Because the value is returned from nodes farther away from the key, this alleviates possible "hot spots". Caching nodes will drop the value after a certain time depending on their distance from the key.

Some implementations (e.g. Kad) do not have replication nor caching. The purpose of this is to remove old information quickly from the system. The node that is providing the file will periodically refresh the information onto the network (perform FIND_NODE and STORE messages). When all of the nodes having the file go offline, nobody will be refreshing its values (sources and keywords) and the information will eventually disappear from the network.

Joining the network

A node that would like to join the net must first go through a bootstrap process. In this phase, the joining node needs to know the IP address and port of another node—a bootstrap node (obtained from the user, or from a stored list)—that is already participating in the Kademlia network. If the joining node has not yet participated in the network, it computes a random ID number that is supposed not to be already assigned to any other node. It uses this ID until leaving the network.

The joining node inserts the bootstrap node into one of its *k-buckets*. The joining node then does a FIND_NODE of its own ID against the bootstrap node (the only other node it knows). The "self-lookup" will populate other nodes' *k-buckets* with the new node ID, and will populate the joining node's k-buckets with the nodes in the path between it and the bootstrap node. After this, the joining node refreshes all *k-buckets* further away than the k-bucket the bootstrap node falls in. This refresh is just a lookup of a random key that is within that *k-bucket* range.

Initially, nodes have one *k-bucket*. When the *k-bucket* becomes full, it can be split. The split occurs if the range of nodes in the *k-bucket* spans the node's own id (values to the left and right in a binary tree). Kademlia relaxes even this rule for the one "closest nodes" *k-bucket*, because typically one single bucket will correspond to the distance where all the nodes that are the closest to this node are, they may be more than k, and we want it to know them all. It may turn out that a highly unbalanced binary sub-tree exists near the node. If *k* is 20, and there are 21+ nodes with a prefix "xxx0011....." and the new node is "xxx0000*11001*", the new node can contain multiple *k-buckets* for the other 21+ nodes. This is to guarantee that the network knows about all nodes in the closest region.

Accelerated lookups

Kademlia uses an *XOR metric* to define distance. Two node ID's or a node ID and a key are XORed and the result is the distance between them. For each bit, the XOR function returns zero if the two bits are equal and one if the two bits are different. XOR metric distances hold the triangle inequality: given A, B and C are vertices(points) of a triangle, then the distance from A to B is shorter than (or equal to) the sum of the distance from A to C to B.

The *XOR metric* allows Kademlia to extend routing tables beyond single bits. Groups of bits can be placed in *k-buckets*. The group of bits are termed a prefix. For an *m-bit* prefix, there will be 2^m-1 *k-buckets*. The missing *k-bucket* is a further extension of the routing tree that contains the node ID. An *m-bit* prefix reduces the maximum number of lookups from $log_2\ n$ to $log_2 m\ n$. These are **maximum** values and the average value will be far less, increasing the chance of finding a node in a *k-bucket* that shares more bits than just the prefix with the target key.

Nodes can use mixtures of prefixes in their routing table, such as the Kad Network used by eMule. Wikipedia:Citation needed The Kademlia network could even be heterogeneous in routing table implementations, at the expense of complicating the analysis of lookups.

Academic significance

While the XOR metric is not needed to understand Kademlia, it is critical in the analysis of the protocol. The XOR arithmetic forms an abelian group allowing closed analysis. Other DHT protocols and algorithms required simulation or complicated formal analysis in order to predict network behavior and correctness. Using groups of bits as routing information also simplifies the algorithms.

Use in file sharing networks

Kademlia is used in file sharing networks. By making Kademlia keyword searches, one can find information in the file-sharing network so it can be downloaded. Since there is no central instance to store an index of existing files, this task is divided evenly among all clients: If a node wants to share a file, it processes the contents of the file, calculating from it a number (hash) that will identify this file within the file-sharing network. The hashes and the node IDs must be of the same length. It then searches for several nodes whose ID is close to the hash, and has its own IP address stored at those nodes. i.e. it publishes itself as a source for this file. A searching client will use Kademlia to search the network for the node whose ID has the smallest distance to the file hash, then will retrieve the sources list that is stored in that node.

Since a key can correspond to many values, e.g. many sources of the same file, every storing node may have different information. Then, the sources are requested from all k nodes close to the key.

The file hash is usually obtained from a specially formed Internet link found elsewhere, or included within an indexing file obtained from other sources.

Filename searches are implemented using keywords. The filename is divided into its constituent words. Each of these keywords is hashed and stored in the network, together with the corresponding filename and file hash. A search involves choosing one of the keywords, contacting the node with an ID closest to that keyword hash, and retrieving the list of filenames that contain the keyword. Since every filename in the list has its hash attached, the chosen file can then be obtained in the normal way.

Implementations

Networks

Public networks using the Kademlia algorithm (these networks are incompatible with one another):

- Kad Network — developed originally by the eMule community to replace the server-based architecture of the eDonkey2000 network.
- Overnet network: With KadC [5] a C library for handling its Kademlia is available. (development of Overnet is discontinued)
- BitTorrent Uses a DHT based on an implementation of the Kademlia algorithm, for trackerless torrents.
- Osiris sps (all version): used to manage distributed and anonymous web portal.
- Retroshare — F2F decentralised communication platform with secure VOIP, instant messaging, file transfer etc.
- Gnutella DHT — Originally by LimeWire[6][7] to augment the Gnutella protocol for finding alternate file locations, now in use by other gnutella clients.[8]

Libraries

- Khashmir [9] — Python implementation of Kademlia. Used in the mainline BitTorrent, with some modifications.
- Entangled [10] — Python implementation of Kademlia, also providing a distributed tuple space. LGPL licensed
- Mojito — a Java Kademlia library written for the LimeWire project. See the Class interface documentation [11]Wikipedia:Link rot for more information.
- maidsafe-dht [12] — c++ implementation of Kademlia (Commercial or GPL license), with NAT traversal and crypto libraries.
- Plan-x [13] — Java implementation.
- Daylight [14] — C#/.NET implementation.
- BitDHT [15] — C++ implementation of Kademlia. Uses BitTorrent DHT network (LGPL license)
- Ardverk-DHT [16] — Java implementation (ASL 2.0 license)
- TomP2P [17] - Java implementation with UPNP support for NAT traversal (ASL 2.0 license)
- openkad [18] - Java implementation with distributed caching (GNU GPL v3 license)

Next generation

Over the years, the academic and practitioner communities have realized that all current DHT designs suffer from a security weakness, known as the Sybil attack. Petar Maymounkov [19], one of the original authors of Kademlia, has proposed a way to circumvent this weakness by incorporating social trust relationships into the system design. See [20], page 20 for social-network based approaches.

The new system, codenamed Tonika [21] or also known by its domain name as 5ttt, is based on an algorithm design known as Electric routing [22] and co-authored with the mathematician Jonathan Kelner [23]. Maymounkov has now undertaken a comprehensive implementation effort of this new system, which is entirely based on the Go programming language. However, research into effective defences against Sybil attacks is generally considered an open question, and wide variety of potential defences are proposed every year in top security research conferences.

References

[1] * Kademlia: A Peer-to-peer information system based on the XOR Metric (http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf)
[2] http://www.scs.stanford.edu/~dm/home/papers/
[3] Stefan Saroiu, P. Krishna Gummadi and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. Technical Report UW-CSE-01-06-02, University of Washington, Department of Computer Science and Engineering, July 2001.
[4] Daniel Stutzbach and Reza Rejaie. Understanding Churn in Peer-to-Peer Networks (http://www.barsoom.org/papers/imc-2006-churn.pdf) Section 5.5 Uptime Predictability, Internet Measurement Conference, Rio de Janeiro, October, 2006.
[5] http://kadc.sourceforge.net/
[6] Mojito and LimeWire (http://www.slyck.com/story1235.html)
[7] Mojito Wiki (archive.org) (http://web.archive.org/web/20090217070609/http://wiki.limewire.org/index.php?title=Mojito)
[8] Gtk-gnutella changelog (https://gtk-gnutella.svn.sourceforge.net/svnroot/gtk-gnutella/trunk/gtk-gnutella/ChangeLog) Retrieved January, 2010.
[9] http://khashmir.sourceforge.net/
[10] http://entangled.sourceforge.net/
[11] http://www.limewire.org/nightly/modules/mojito/api/org/limewire/mojito/class-use/MojitoDHT.html
[12] http://code.google.com/p/maidsafe-dht/
[13] http://www.thomas.ambus.dk/plan-x/routing/
[14] http://daylight.sourceforge.net/
[15] http://bitdht.sourceforge.net/
[16] https://github.com/rkapsi/ardverk-dht
[17] http://tomp2p.net/
[18] http://code.google.com/p/openkad/
[19] http://pdos.csail.mit.edu/~petar/
[20] http://pdos.csail.mit.edu/papers/sybil-dht-socialnets08.pdf
[21] http://pdos.csail.mit.edu/~petar/5ttt.org/
[22] http://arxiv.org/abs/0909.2859
[23] http://math.mit.edu/~kelner/

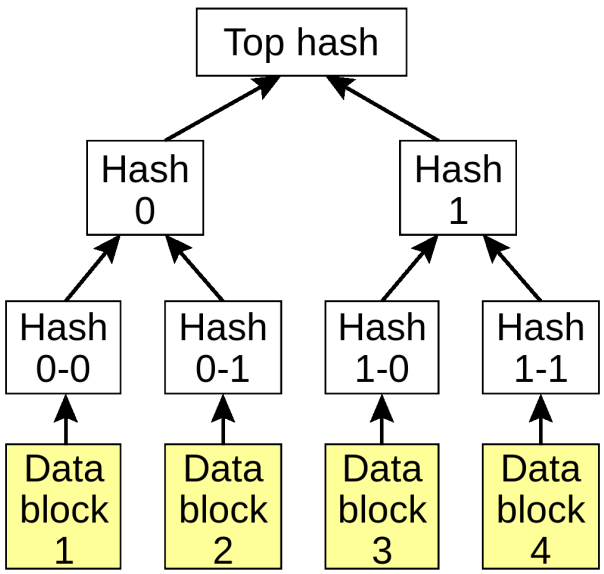
External links

- Academic Home Page (http://pdos.csail.mit.edu/~petar/) of Petar Maymounkov, co-designer of Kademlia.
- Xlattice projects (http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html) Kademlia Specification and definitions.
- Yi Qiao (http://www.cs.northwestern.edu/~yqiao) and Fabian E. Bustamante (http://www.cs.northwestern.edu/~fabianb) USENIX 2006 paper that characterizes Overnet and Gnutella (http://www.aqualab.cs.northwestern.edu/publications/YQiao06SUO.pdf)
- Stutzbach, Daniel (2006). "Improving Lookup Performance over a Widely-Deployed DHT" (http://www.barsoom.org/~agthorr/papers/infocom-2006-kad.pdf). University of Oregon.
- Kaleidoscope: Adding Colors to Kademlia (http://www.cs.technion.ac.il/~gilga/P2P2013_49.pdf) IEEE P2P 2013 (cache friendly lookup for kademlia)
- Mainline DHT Measurement (http://www.cs.helsinki.fi/u/jakangas/MLDHT/) at Department of Computer Science, University of Helsinki, Finland.

Merkle tree

In cryptography and computer science a **hash tree** or **Merkle tree** is a tree in which every non-leaf node is labelled with the hash of the labels of its children nodes. Hash trees are useful because they allow efficient and secure verification of the contents of larger data structures. Hash trees are a generalization of hash lists and hash chains.

Demonstrating that a leaf node is a part of the given hash tree requires processing an amount of data proportional to the logarithm of the number of nodes of the tree; this contrasts with hash lists, where the amount is proportional to the number of nodes. The concept is named after Ralph Merkle.



A binary hash tree

Uses

Hash trees can be used to verify any kind of data stored, handled and transferred in and between computers. Currently the main use of hash trees is to make sure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks. Suggestions have been made to use hash trees in trusted computing systems.[1] Hash Trees are used in the ZFS file system,[2] BitTorrent protocol, Apache Wave protocol,[3] Git distributed revision control system, the Tahoe-LAFS backup system, the Bitcoin peer-to-peer network, and a number of NoSQL systems like Apache Cassandra and Riak.[4]

Hash trees were patented in 1979 by Ralph Merkle. The original purpose was to make it possible to efficiently handle many Lamport one-time signatures. Lamport signatures are believed to still be secure in the event that quantum computers become reality; see post-quantum cryptography. Unfortunately each Lamport key can only be used to sign a single message. But combined with hash trees they can be used for many messages and then become a fairly efficient digital signature scheme.

Overview

A hash tree is a tree of hashes in which the leaves are hashes of data blocks in, for instance, a file or set of files. Nodes further up in the tree are the hashes of their respective children. For example, in the picture *hash 0* is the result of hashing the result of concatenating *hash 0-0* and *hash 0-1*. That is, *hash 0 = hash(hash 0-0 + hash 0-1)* where + denotes concatenation.

Most hash tree implementations are binary (two child nodes under each node) but they can just as well use many more child nodes under each node.

Usually, a cryptographic hash function such as SHA-2 or SHA-3 is used for the hashing. If the hash tree only needs to protect against unintentional damage, much less secure checksums such as CRCs can be used.

In the top of a hash tree there is a *top hash* (or *root hash* or *master hash*). Before downloading a file on a p2p network, in most cases the top hash is acquired from a trusted source, for instance a friend or a web site that is known to have good recommendations of files to download. When the top hash is available, the hash tree can be

received from any non-trusted source, like any peer in the p2p network. Then, the received hash tree is checked against the trusted top hash, and if the hash tree is damaged or fake, another hash tree from another source will be tried until the program finds one that matches the top hash.

The main difference from a hash list is that one branch of the hash tree can be downloaded at a time and the integrity of each branch can be checked immediately, even though the whole tree is not available yet. For example, in the picture the integrity of *data block 2* can be verified immediately if the tree already contains *hash 0-0* and *hash 1* by hashing the data block and iteratively combining the result with *hash 0-0* and then *hash 1* and finally comparing the result with the *top hash*. Similarly, the integrity of *data block 3* can be verified if the tree already has *hash 1-1* and *hash 0*. This can be an advantage since it is efficient to split files up in very small data blocks so that only small blocks have to be re-downloaded if they get damaged. If the hashed file is very big, such a hash tree or hash list becomes fairly big. But if it is a tree, one small branch can be downloaded quickly, the integrity of the branch can be checked, and then the downloading of data blocks can start.

There are several additional tricks, benefits and details regarding hash trees. See the references and external links below for more in-depth information.

Tiger tree hash

The Tiger tree hash is a widely used form of hash tree. It uses a binary hash tree (two child nodes under each node), usually has a data block size of 1024-bytes and uses the cryptographically secure Tiger hash.

Tiger tree hashes are used in Gnutella, Gnutella2, and Direct Connect P2P file sharing protocols and in file sharing applications such as Phex, BearShare, LimeWire, Shareaza, DC++[5] and Valknut.Wikipedia:Citation needed

References

[1] J. Kilian. Improved efficient arguments (preliminary version). In CRYPTO, 1995.

[2] Jeff Bonwick's Blog *ZFS End-to-End Data Integrity* (https://blogs.oracle.com/bonwick/entry/zfs_end_to_end_data)

[3] Google Wave Federation Protocol *Wave Protocol Verification Paper* (http://www.waveprotocol.org/protocol/whitepapers/wave-protocol-verification)

[4] "When a replica is down for an extended period of time, or the machine storing hinted handoffs for an unavailable replica goes down as well, replicas must synchronize from one another. In this case, Cassandra and Riak implement a Dynamo-inspired process called anti-entropy. In anti-entropy, replicas exchange Merkle trees to identify parts of their replicated key ranges which are out of sync. A Merkle tree is a hierarchical hash verification: if the hash over the entire keyspace is not the same between two replicas, they will exchange hashes of smaller and smaller portions of the replicated keyspace until the out-of-sync keys are identified. This approach reduces unnecessary data transfer between replicas which contain mostly similar data." http://www.aosabook.org/en/nosql.html

[5] "DC++'s feature list" (http://dcplusplus.sourceforge.net/features.html)

Further reading

- Merkle tree patent 4,309,569 (http://www.google.com/patents/US4309569) – Explains both the hash tree structure and the use of it to handle many one-time signatures.
- Tree Hash EXchange format (THEX) (http://web.archive.org/web/20080316033726/http://www.open-content.net/specs/draft-jchapweske-thex-02.html) – A detailed description of Tiger trees.
- Efficient Use of Merkle Trees (http://www.emc.com/emc-plus/rsa-labs/historical/recent-improvements-efficient-use-merkle-trees.htm) – RSA labs explanation of the original purpose of Merkle trees: To handle many Lamport one-time signatures.

External links

- http://www.codeproject.com/cs/algorithms/thexcs.asp – Tiger Tree Hash (TTH) source code in C# – by Gil Schmidt
- http://sourceforge.net/projects/tigertree/ – Tiger Tree Hash (TTH) implementations in C and Java
- RHash (http://rhash.sourceforge.net/), an open source command-line tool, which can calculate TTH and magnet links with TTH.
- "Series of mini-lectures about cryptographic hash functions" (http://www.guardtime.com/educational-series-on-hashes/); includes application in time-stamping and provable security; by A. Buldas, 2011.

Peer-to-peer

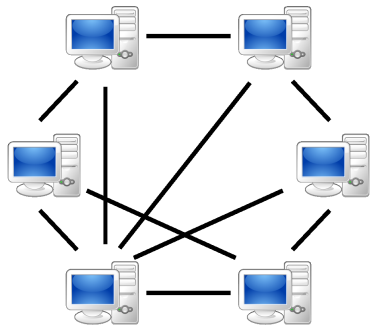
Not to be confused with Point-to-point (telecommunications).

This article is about peer-to-peer computer networks. For other uses, see Peer-to-peer (disambiguation).

A **peer-to-peer (P2P) network** is a type of decentralized and distributed network architecture in which individual nodes in the network (called *"peers"*) act as both suppliers and consumers of resources, in contrast to centralized client–server model where client nodes request access to resources provided by central servers.

Networks in which all computers have equal status are called peer-to-peer or P2P networks.

In a peer-to-peer network, tasks (such as searching for files or streaming audio/video) are shared amongst multiple interconnected peers who each make a portion of their resources (such as processing power, disk storage or network bandwidth) directly available to other network participants, without the need for centralized coordination by servers.[1]



A **peer-to-peer (P2P) network** in which interconnected nodes ("peers") share resources amongst each other without the use of a centralized administrative system

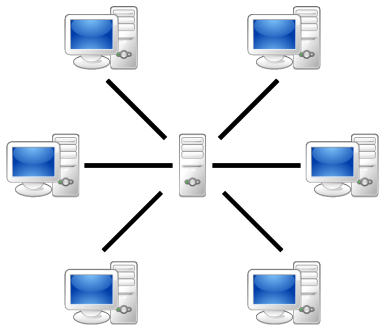
Historical development

While P2P systems had previously been used in many application domains,[2] the concept was popularized by file sharing systems such as the music-sharing application Napster (originally released in 1999).Wikipedia:Citation needed The peer-to-peer movement allowed millions of Internet users to connect "directly, forming groups and collaborating to become user-created search engines, virtual supercomputers, and filesystems." [3] The basic concept of peer-to-peer computing was envisioned in earlier software systems and networking discussions, reaching back to principles stated in the first Request for Comments, RFC 1.[4]

Tim Berners-Lee's vision for the World Wide Web was close to a P2P network in that it assumed each user of the web would be an active editor and contributor, creating and linking content to form

an interlinked "web" of links. The early Internet was more open than present day, where two machines connected to the Internet could send packets to each other without firewalls and other security measures. This contrasts to the broadcasting-like structure of the web as it has developed over the years.

The idea of peer-to-peer began in the 1960s, when ARPANET was created as a network to share files between US research facilities and every host was seen as equal, none more important than the others.[5] As a precursor to the Internet, ARPANET was a successful client-server network where "every participating node could request and serve content." However, ARPANET was not self-organized and it lacked the ability to "provide any means for context or content based routing beyond 'simple' addressed based routing."



A network based on the **client-server model**, where individual *clients* request services and resources from centralized servers

Therefore, a distributed messaging system that is often likened as an early peer-to-peer architecture was established: USENET. USENET was developed in 1979 and is a system that enforces a decentralized model of control. The basic model is a client-server model from the user or client perspective that offers a self-organizing approach to newsgroup servers. However, news servers communicate with one another as peers to propagate Usenet news articles over the entire group of network servers. The same consideration applies to SMTP email in the sense that the core email relaying network of Mail transfer agents has a peer-to-peer character, while the periphery of e-mail clients and their direct connections is strictly a client–server relationship.Wikipedia:Citation needed

In May 1999, with millions more people on the Internet, Shawn Fanning introduced the music and file-sharing application called Napster. Napster was the beginning of peer-to-peer networks, as we know them today, where "participating users establish a virtual network, entirely independent from the physical network, without having to obey any administrative authorities or restrictions."

Current applications

Peer-to-peer networks underlie numerous applications. The most commonly known application is file sharing, which popularized the technology.

Communications

- Instant messaging systems and online chat networks.

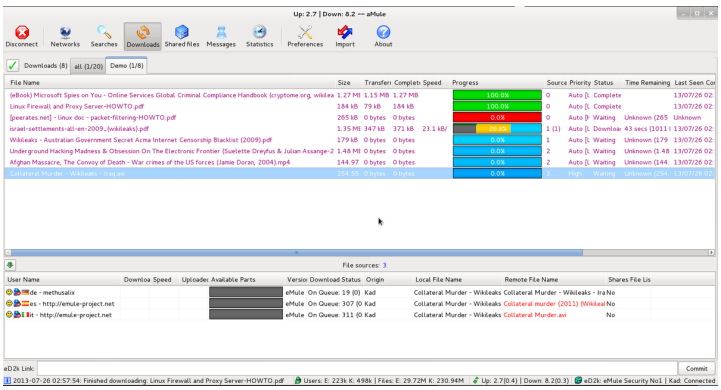
Other P2P applications

- Bitcoin and alternatives such as Peercoin are peer-to-peer-based digital currencies.
- The U.S. Department of Defense is conducting research on P2P networks as part of its modern network warfare strategy. In May, 2003, Anthony Tether, then director of DARPA, testified that the U.S. military uses P2P networks.
- Wireless community network, Netsukuku
- Dalesa a peer-to-peer web cache for LANs (based on IP multicasting).
- Open Garden, connection sharing application that shares Internet access with other devices using Wi-Fi or Bluetooth.

- Research like the Chord project, the PAST storage utility, the P-Grid, and the CoopNet content distribution system.
- JXTA
- CurrencyFair is a peer-to-peer foreign currency exchange marketplace.

Content delivery

In P2P networks, clients provide resources as well as using them. This means that unlike client-server systems, the content serving capacity of peer-to-peer networks can actually *increase* as more users begin to access the content (especially with protocols such as Bittorrent that require users to share). This property is one of the major advantages of using P2P networks because it makes the setup and running costs very small for the original content distributor.



The use of peer-to-peer file-sharing software such as aMule (pictured here) is responsible for the bulk of P2P internet traffic

File-sharing networks

See also: Peer-to-peer file sharing

Many file peer-to-peer file sharing networks, such as Gnutella, G2, and the eDonkey network popularized peer-to-peer technologies. From 2004 on, such networks form the largest contributor of network traffic on the Internet.Wikipedia:Citation needed

- Peer-to-peer content delivery networks. See: Kontiki, Ignite, RedSwoosh.
- Peer-to-peer content services, e.g. caches for improved performance such as Correli Caches[6]
- Software publication and distribution (Linux distribution, several games); via file sharing networks.

Streaming media

- Streaming media. P2PTV and PDTP. Applications include TVUPlayer, Joost, CoolStreaming, Cybersky-TV, PPLive, LiveStation, Giraffic and Didiom.
- Spotify uses a peer-to-peer network along with streaming servers to stream music to its desktop music player.
- Peercasting for multicasting streams. See PeerCast, IceShare, FreeCast, Rawflow
- Pennsylvania State University, MIT and Simon Fraser University are carrying on a project called LionShare designed for facilitating file sharing among educational institutions globally.
- Osiris (Serverless Portal System) allows its users to create anonymous and autonomous web portals distributed via P2P network.

Intellectual property law and illegal sharing

Peer-to-peer networking involves data transfer from one user to another without using an intermediate server. Companies developing P2P applications have been involved in numerous legal cases, primarily in the United States, primarily over issues surrounding copyright law. Two major cases are *Grokster vs RIAA* and *MGM Studios, Inc. v. Grokster, Ltd.*.[7] In both of the cases the file sharing technology was ruled to be legal as long as the developers had no ability to prevent the sharing of the copyrighted material.Wikipedia:Citation needed

Architecture

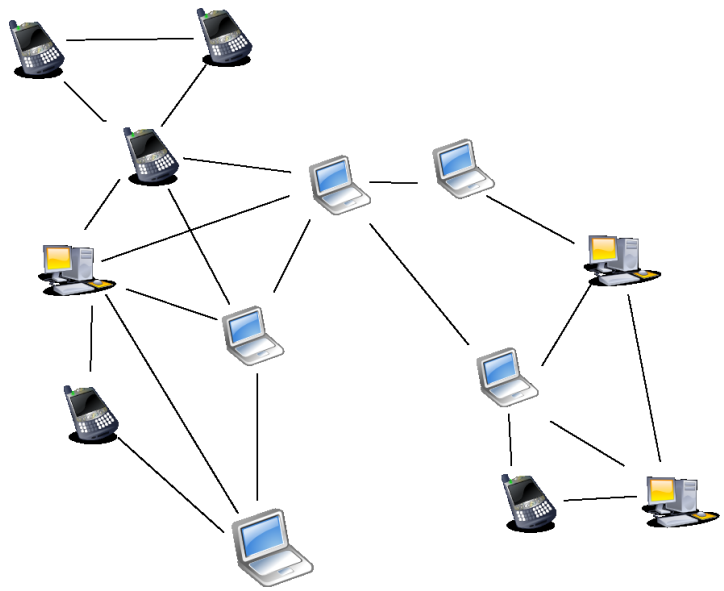
A peer-to-peer network is designed around the notion of equal *peer* nodes simultaneously functioning as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client–server model where communication is usually to and from a central server. A typical example of a file transfer that uses the client-server model is the File Transfer Protocol (FTP) service in which the client and server programs are distinct: the clients initiate the transfer, and the servers satisfy these requests.

Routing and resource discovery

Peer-to-peer networks generally implement some form of virtual overlay network on top of the physical network topology, where the nodes in the overlay form a subset of the nodes in the physical network. Data is still exchanged directly over the underlying TCP/IP network, but at the application layer peers are able to communicate with each other directly, via the logical overlay links (each of which corresponds to a path through the underlying physical network). Overlays are used for indexing and peer discovery, and make the P2P system independent from the physical network topology. Based on how the nodes are linked to each other within the overlay network, and how resources are indexed and located, we can classify networks as *unstructured* or *structured* (or as a hybrid between the two).

Unstructured networks

Unstructured peer-to-peer networks do not impose a particular structure on the overlay network by design, but rather are formed by nodes that randomly form connections to each other. (Gnutella, Gossip, and Kazaa are examples of unstructured P2P protocols.)



Overlay network diagram for an **unstructured P2P network**, illustrating the ad hoc nature of the connections between nodes

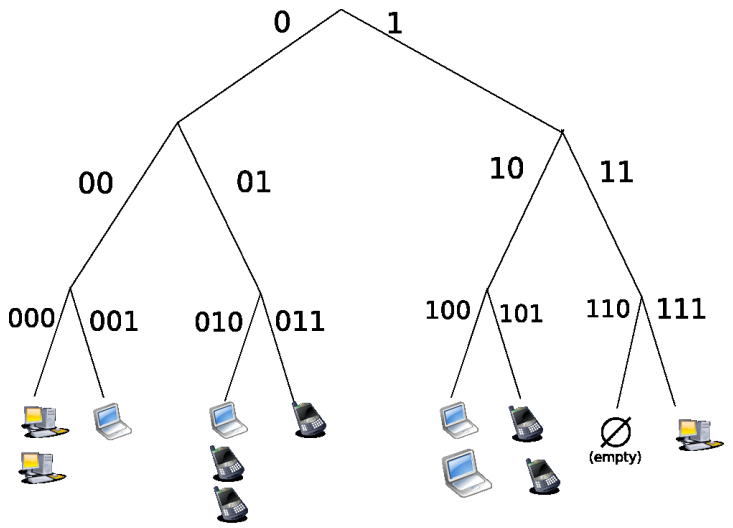
Because there is no structure globally imposed upon them, unstructured networks are easy to build and allow for localized optimizations to different regions of the overlay. Also, because the role of all peers in the network is the same, unstructured networks are highly robust in the face of high rates of "churn"—that is, when large numbers of peers are frequently joining and leaving the network.

However the primary limitations of unstructured networks also arise from this lack of structure. In particular, when a peer wants to find a desired piece of data in the network, the search query must be flooded through the network to find as many peers as possible that share the data. Flooding causes a very high amount of signaling traffic in the network, uses more CPU/memory (by requiring every peer to process all search queries), and does not ensure that search queries will always be resolved. Furthermore, since there is no correlation between a peer and the content managed by it, there is no guarantee that flooding will find a peer that has the desired data. Popular content is likely to be available at several peers and any peer searching for it is likely to find the same thing. But if a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful.

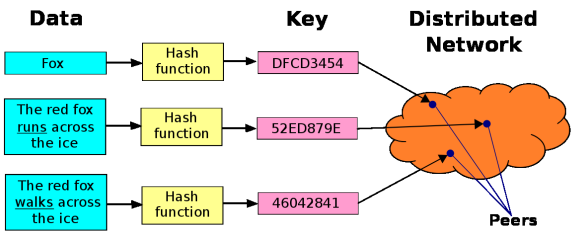
Structured networks

In *structured peer-to-peer networks* the overlay is organized into a specific topology, and the protocol ensures that any node can efficiently[8] search the network for a file/resource, even if the resource is extremely rare.

The most common type of structured P2P networks implement a distributed hash table (DHT),[9][10] in which a variant of consistent hashing is used to assign ownership of each file to a particular peer. This enables peers to search for resources on the network using a hash table: that is, (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key.[11][12]



Overlay network diagram for a **structured P2P network**, using a distributed hash table (DHT) to identify and locate nodes/resources



Distributed hash tables

However, in order to route traffic efficiently through the network, nodes in a structured overlay must maintain lists of neighbors that satisfy specific criteria. This makes them less robust in networks with a high rate of *churn* (i.e. with large numbers of nodes frequently joining and leaving the network). More recent evaluation of P2P resource discovery solutions under real workloads have pointed out several issues in DHT-based solutions such as high cost of advertising/discovering resources and static and dynamic load imbalance.

Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the Kad network, the Storm botnet, YaCy, and the Coral Content Distribution Network. Some prominent research projects include the Chord project, Kademlia, PAST storage utility, P-Grid, a self-organized and emerging overlay network, and CoopNet content distribution system.Wikipedia:Citation needed DHT-based networks have also been widely utilized for accomplishing efficient resource discovery for grid computing systems, as it aids in resource management and scheduling of applications.

Hybrid models

Hybrid models are a combination of peer-to-peer and client-server models. A common hybrid model is to have a central server that helps peers find each other. Spotify is an example of a hybrid model. There are a variety of hybrid models, all of which make trade-offs between the centralized functionality provided by a structured server/client network and the node equality afforded by the pure peer-to-peer unstructured networks. Currently, hybrid models have better performance than either pure unstructured networks or pure structured networks because certain functions, such as searching, do require a centralized functionality but benefit from the decentralized aggregation of nodes provided by unstructured networks.

Security and trust

Peer-to-peer systems pose unique challenges from a computer security perspective.

Like any other form of software, P2P applications can contain vulnerabilities. What makes this particularly dangerous for P2P software, however, is that peer-to-peer applications act as servers as well as clients, meaning that they can be more vulnerable to remote exploits.

Routing attacks

Also, since each node plays a role in routing traffic through the network, malicious users can perform a variety of "routing attacks", or denial of service attacks. Examples of common routing attacks include "incorrect lookup routing" whereby malicious nodes deliberately forward requests incorrectly or return false results, "incorrect routing updates" where malicious nodes corrupt the routing tables of neighboring nodes by sending them false information, and "incorrect routing network partition" where when new nodes are joining they bootstrap via a malicious node, which places the new node in a partition of the network that is populated by other malicious nodes.

Corrupted data and malware

See also: Data validation and Malware

The prevalence of malware varies between different peer-to-peer protocols. Studies analyzing the spread of malware on P2P networks found, for example, that 63% of the answered download requests on the Limewire network contained some form of malware, whereas only 3% of the content on OpenFT contained malware. In both cases, the top three most common types of malware accounted for the large majority of cases (99% in Limewire, and 65% in OpenFT). Another study analyzing traffic on the Kazaa network found that 15% of the 500,000 file sample taken were infected by one or more of the 365 different computer viruses that were tested for.

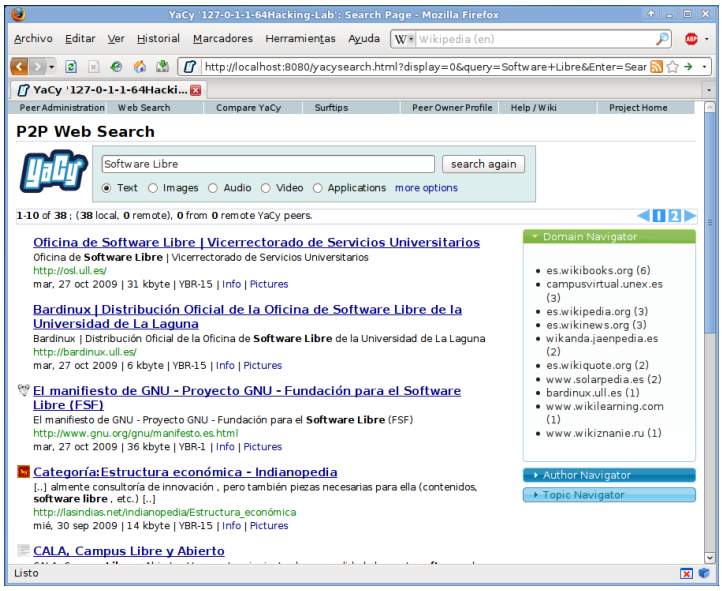
Corrupted data can also be distributed on P2P networks by modifying files that are already being shared on the network. For example, on the FastTrack network, the RIAA managed to introduce faked chunks into downloads and downloaded files (mostly MP3 files). Files infected with the RIAA virus were unusable afterwards and contained malicious code. The RIAA is also known to have uploaded fake music and movies to P2P networks in order to deter illegal file sharing. Consequently, the P2P networks of today have seen an enormous increase of their security and file verification mechanisms. Modern hashing, chunk verification and different encryption methods have made most networks resistant to almost any type of attack, even when major parts of the respective network have been replaced by faked or nonfunctional hosts.

Creating more resilient and scalable computer networks

See also: Wireless mesh network and Distributed computing

The decentralized nature of P2P networks increases robustness because it removes the single point of failure that can be inherent in a client-server based system. As nodes arrive and demand on the system increases, the total capacity of the system also increases, and the likelihood of failure decreases. If one peer on the network fails to function properly, the whole network is not compromised or damaged. In contrast, in a typical client–server architecture, clients share only their demands with the system, but not their resources. In this case, as more clients join the system, fewer resources are available to serve each client, and if the central server fails, the entire network is taken down.

Distributed storage and search



Search results for the query "software libre", using YaCy a free distributed search engine that runs on a peer-to-peer network instead making requests to centralized index servers (like Google, Yahoo, and other corporate search engines)

There are both advantages and disadvantages in P2P networks related to the topic of data backup, recovery, and availability. In a centralized network, the system administrators are the only forces controlling the availability of files being shared. If the administrators decide to no longer distribute a file, they simply have to remove it from their servers, and it will no longer be available to users. Along with leaving the users powerless in deciding what is distributed throughout the community, this makes the entire system vulnerable to threats and requests from the government and other large forces. For example, YouTube has been pressured by the RIAA, MPAA, and entertainment industry to filter out copyrighted content. Although server-client networks are able to monitor and manage content availability, they can have more stability in the availability of the content they choose to host. A client should not have trouble accessing obscure content that is being shared on a stable centralized network. P2P networks, however, are more unreliable in sharing unpopular files because sharing files in a P2P network requires that at least one node in the network has the requested data, and that node must be able to connect to the node requesting the data. This requirement is occasionally hard to meet because users may delete or stop sharing data at any point.

In this sense, the community of users in a P2P network is completely responsible for deciding what content is available. Unpopular files will eventually disappear and become unavailable as more people stop sharing them. Popular files, however, will be highly and easily distributed. Popular files on a P2P network actually have more stability and availability than files on central networks. In a centralized network a simple loss of connection between the server and clients is enough to cause a failure, but in P2P networks the connections between every node must be lost in order to cause a data sharing failure. In a centralized system, the administrators are responsible for all data recovery and backups, while in P2P systems, each node requires its own backup system. Because of the lack of central authority in P2P networks, forces such as the recording industry, RIAA, MPAA, and the government are unable to delete or stop the sharing of content on P2P systems.

In addition to data storage, P2P systems also provide the ability to provide distributed search engines to locate resources shared on a network. Examples include:

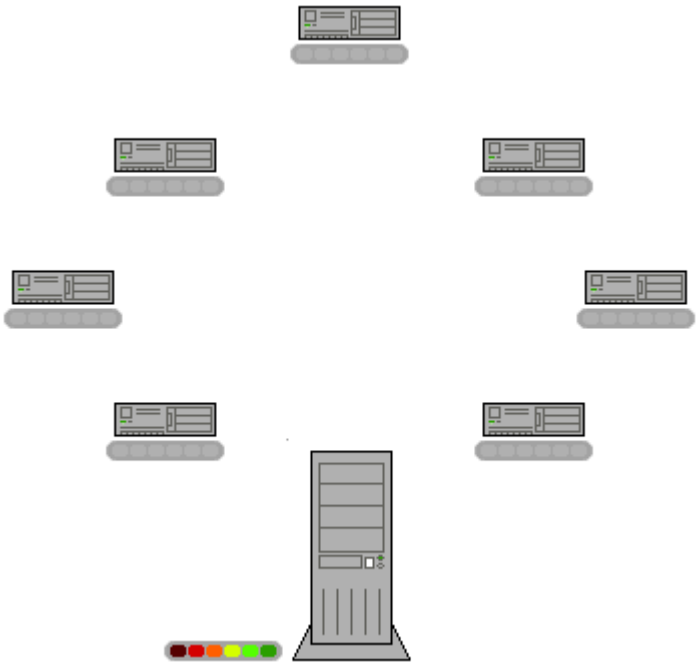
- YaCy, a free distributed search engine, built on principles of peer-to-peer networks.
- FAROO, another distributed search engine.
- The sciencenet P2P search engine.

Social implications

See also: Social peer-to-peer processes

Incentivizing resource sharing and cooperation

Cooperation among a community of participants is key to the continued success of P2P systems aimed at casual human users; these reach their full potential only when large numbers of nodes contribute resources. But in current practice P2P networks often contain large numbers of users who utilize resources shared by other nodes, but who do not share anything themselves (often referred to as the "freeloader problem"). Freeloading can have a profound impact on the network and in some cases can cause the community to collapse.[13] In these types of networks "users have natural disincentives to cooperate because cooperation consumes their own resources and may degrade their own performance." [14] Studying the social attributes of P2P networks is challenging due to large populations of turnover, asymmetry of interest and zero-cost identity. A variety of incentive mechanisms have been implemented to encourage or even force nodes to contribute resources.



The BitTorrent protocol: In this animation, the colored bars beneath all of the 7 clients in the upper region above represent the file being shared, with each color representing an individual piece of the file. After the initial pieces transfer from the seed (large system at the bottom), the pieces are individually transferred from client to client. The original seeder only needs to send out one copy of the file for all the clients to receive a copy.

Some researchers have explored the benefits of enabling virtual communities to self-organize and introduce incentives for resource sharing and cooperation, arguing that the social aspect missing from today's P2P systems should be seen both as a goal and a means for self-organized virtual communities to be built and fostered.[15] Ongoing research efforts for designing effective incentive mechanisms in P2P systems, based on principles from game theory, are beginning to take on a more psychological and information-processing direction.

Privacy/anonymity

Some peer-to-peer networks (e.g. Freenet) place a heavy emphasis on privacy and anonymity—that is, ensuring that the contents of communications are hidden from eavesdroppers, and that the identities/locations of the participants are concealed. Public key cryptography can be used to provide encryption, data validation, authorization, and authentication for data/messages. Onion routing and other mix network protocols (e.g. Tarzan) can be used to provide anonymity.

Political implications

Intellectual property law and illegal sharing

Although peer-to-peer networks can be used for legitimate purposes, rights holders have targeted peer-to-peer over the involvement with sharing copyrighted material. Peer-to-peer networking involves data transfer from one user to another without using an intermediate server. Companies developing P2P applications have been involved in numerous legal cases, primarily in the United States, primarily over issues surrounding copyright law. Two major cases are *Grokster vs RIAA* and *MGM Studios, Inc. v. Grokster, Ltd.*. In both of the cases the file sharing technology was ruled to be legal as long as the developers had no ability to prevent the sharing of the copyrighted material. To establish criminal liability for the copyright infringement on peer-to-peer systems, the government must prove that the defendant infringed a copyright for the purpose of a personal financial gain or advantage.[16] Fair use exceptions allow limited use of copyrighted material to be downloaded without acquiring permission from the rights holders. These documents are usually news reporting or under the lines of research and scholarly work. Controversies have developed over the concern of illegitimate use of peer-to-peer networks regarding public safety and national security. When a file is downloaded through a peer-to-peer network, it is impossible to know who created the file or what users are connected to the network at a given time. Trustworthiness of sources is a potential security threat that can be seen with peer-to-peer systems.[17]

Network neutrality

Peer-to-peer applications present one of the core issues in the network neutrality controversy. Internet service providers (ISPs) have been known to throttle P2P file-sharing traffic due to its high-bandwidth usage.[18] Compared to Web browsing, e-mail or many other uses of the internet, where data is only transferred in short intervals and relative small quantities, P2P file-sharing often consists of relatively heavy bandwidth usage due to ongoing file transfers and swarm/network coordination packets. In October 2007, Comcast, one of the largest broadband Internet providers in the USA, started blocking P2P applications such as BitTorrent. Their rationale was that P2P is mostly used to share illegal content, and their infrastructure is not designed for continuous, high-bandwidth traffic. Critics point out that P2P networking has legitimate legal uses, and that this is another way that large providers are trying to control use and content on the Internet, and direct people towards a client-server-based application architecture. The client-server model provides financial barriers-to-entry to small publishers and individuals, and can be less efficient for sharing large files. As a reaction to this bandwidth throttling, several P2P applications started implementing protocol obfuscation, such as the BitTorrent protocol encryption. Techniques for achieving "protocol obfuscation" involves removing otherwise easily identifiable properties of protocols, such as deterministic byte sequences and packet sizes, by making the data look as if it were random. The ISP's solution to the high bandwidth is P2P caching, where an ISP stores the part of files most accessed by P2P clients in order to save access to the Internet.

Current research

Researchers are currently using computer simulations to aid in understanding and evaluating the complex behaviors of individuals within the network. "Networking research often relies on simulation in order to test and evaluate new ideas. An important requirement of this process is that results must be reproducible so that other researchers can replicate, validate, and extend existing work." [19] If the research cannot be reproduced, then the opportunity for further research is hindered. "Even though new simulators continue to be released, the research community tends towards only a handful of opensource simulators. The demand for features in simulators, as shown by our criteria and survey, is high. Therefore, the community should work together to get these features in open-source software. This would reduce the need for custom simulators, and hence increase repeatability and reputability of experiments."

References

[1] Rüdiger Schollmeier, *A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications*, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002).

[2] D. Barkai, *Peer-to-Peer Computing*, Intel Press, 2002.

[3] Oram, A. (Ed.). (2001). Peer-to-peer: Harnessing the Benefits of a Disruptive Technologies. O'Reilly Media, Inc.

[4] RFC 1, *Host Software*, S. Crocker, IETF Working Group (April 7, 1969)

[5] Steinmetz, R., & Wehrle, K. (2005). 2. What Is This "Peer-to-Peer" About? (pp. 9-16). Springer Berlin Heidelberg.

[6] Gareth Tyson, Andreas Mauthe, Sebastian Kaune, Mu Mu and Thomas Plagemann. Corelli: A Dynamic Replication Service for Supporting Latency-Dependent Content in Community Networks. In Proc. 16th ACM/SPIE Multimedia Computing and Networking Conference (MMCN), San Jose, CA (2009). (http://www.dcs.kcl.ac.uk/staff/tysong/files/MMCN09.pdf)

[7] John Borland, Judge: File-Swapping Tools are Legal , http://news.cnet.com/Judge-File-swapping-tools-are-legal/2100-1027_3-998363.html/

[8] Typically approximating O(log N), where N is the number of nodes in the P2P system

[9] Other design choices include overlay rings and d-Torus. See for example

[10] R. Ranjan, A. Harwood, and R. Buyya, "Peer-to-peer based resource discovery in global grids: a tutorial," *IEEE Commun. Surv.*, vol. 10, no. 2. and P. Trunfio, "Peer-to-Peer resource discovery in Grids: Models and systems," *Future Generation Computer Systems* archive, vol. 23, no. 7, Aug. 2007.

[11] Moni Naor and Udi Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach (http://www.wisdom.weizmann.ac.il/~naor/PAPERS/dh.pdf). Proc. SPAA, 2003.

[12] Gurmeet Singh Manku. Dipsea: A Modular Distributed Hash Table (http://www-db.stanford.edu/~manku/phd/index.html). Ph. D. Thesis (Stanford University), August 2004.

[13] Krishnan, R., Smith, M. D., Tang, Z., & Telang, R. (2004, January). The impact of free-riding on peer-to-peer networks. In System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on (pp. 10-pp). IEEE.

[14] Feldman, M., Lai, K., Stoica, I., & Chuang, J. (2004, May). Robust incentive techniques for peer-to-peer networks. In Proceedings of the 5th ACM conference on Electronic commerce (pp. 102-111). ACM.

[15] P. Antoniadis and B. Le Grand, "Incentives for resource sharing in self-organized communities: From economics to social psychology," Digital Information Management (ICDIM '07), 2007

[16] Majoras, D. B. (2005). Peer-to-peer file-sharing technology consumer protection and competition issues. Federal Trade Commission, Retrieved from http://www.ftc.gov/reports/p2p05/050623p2prpt.pdf

[17] The Government of the Hong Kong Special Administrative Region, (2008). Peer-to-peer network. Retrieved from website: http://www.infosec.gov.hk/english/technical/files/peer.pdf

[18] Janko Roettgers, 5 Ways to Test Whether your ISP throttles P2P, http://newteevee.com/2008/04/02/5-ways-to-test-if-your-isp-throttles-p2p/

[19] Basu, A., Fleming, S., Stanier, J., Naicken, S., Wakeman, I., & Gurbani, V. K. (2013). The state of peer-to-peer network simulators. ACM Computing Surveys (CSUR), 45(4), 46.

External links

- Glossary (http://www.p2pna.com/glossary.html) of P2P terminology
- Foundation of Peer-to-Peer Computing (http://www.sciencedirect.com/science/issue/5624-2008-999689997-678759), Special Issue, Elsevier Journal of Computer Communication, (Ed) Javed I. Khan and Adam Wierzbicki, Volume 31, Issue 2, February 2008
- Ross J. Anderson. The eternity service (http://www.cl.cam.ac.uk/users/rja14/eternity/eternity.html). In *Pragocrypt 1996*, 1996.
- Marling Engle & J. I. Khan. Vulnerabilities of P2P systems and a critical look at their solutions (http://www.medianet.kent.edu/techreports/TR2006-11-01-p2pvuln-EK.pdf), May 2006
- Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies (http://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.html). ACM Computing Surveys, 36(4):335–371, December 2004.
- Biddle, Peter, Paul England, Marcus Peinado, and Bryan Willman, The Darknet and the Future of Content Distribution (http://crypto.stanford.edu/DRM2002/darknet5.doc). In *2002 ACM Workshop on Digital Rights Management*, November 2002.
- John F. Buford, Heather Yu, Eng Keong Lua P2P Networking and Applications (http://www.p2pna.com). ISBN 0123742145, Morgan Kaufmann, December 2008
- Djamal-Eddine Meddour, Mubashar Mushtaq, and Toufik Ahmed, " Open Issues in P2P Multimedia Streaming (http://multicomm.polito.it/proc_multicomm06_8.pdf)", in the proceedings of the 1st Multimedia Communications Workshop MULTICOMM 2006 held in conjunction with IEEE ICC 2006 pp 43–48, June 2006, Istanbul, Turkey.
- Detlef Schoder and Kai Fischbach, " Core Concepts in Peer-to-Peer (P2P) Networking (http://www.econbiz.de/archiv1/2008/42151_concepts_peer-to-peer_networking.pdf)". In: Subramanian, R.; Goodman, B. (eds.): *P2P Computing: The Evolution of a Disruptive Technology*, Idea Group Inc, Hershey. 2005
- Ramesh Subramanian and Brian Goodman (eds), *Peer-to-Peer Computing: Evolution of a Disruptive Technology* (http://www.igi-pub.com/books/details.asp?ID=4635), ISBN 1-59140-429-0, Idea Group Inc., Hershey, PA, USA, 2005.
- Shuman Ghosemajumder. Advanced Peer-Based Technology Business Models (http://dspace.mit.edu/handle/1721.1/8438). *MIT Sloan School of Management, 2002.*
- Silverthorne, Sean. *Music Downloads: Pirates- or Customers?* (http://hbswk.hbs.edu/item.jhtml?id=4206&t=innovation). Harvard Business School Working Knowledge, 2004.

Public key certificate

In cryptography, a **public key certificate** (also known as a **digital certificate** or **identity certificate**) is an electronic document used to prove ownership of a public key. The certificate includes information about the key, information about its owner's identity, and the digital signature of an entity that has verified the certificate's contents are correct. If the signature is valid, and the person examining the certificate trusts the signer, then they know they can use that key to communicate with its owner.

In a typical public-key infrastructure (PKI) scheme, the signer is a certificate authority (CA), usually a company such as VeriSign which charges customers to issue certificates for them. In a web of trust scheme, the signer is either the key's owner (a self-signed certificate) or other users ("endorsements") who the person examining the certificate might know and trust.

Certificates are an important component of Transport Layer Security (TLS, sometimes called by its older name SSL), where they prevent an attacker from impersonating a secure website or other server. They are also used in other important applications, such as email encryption and code signing.

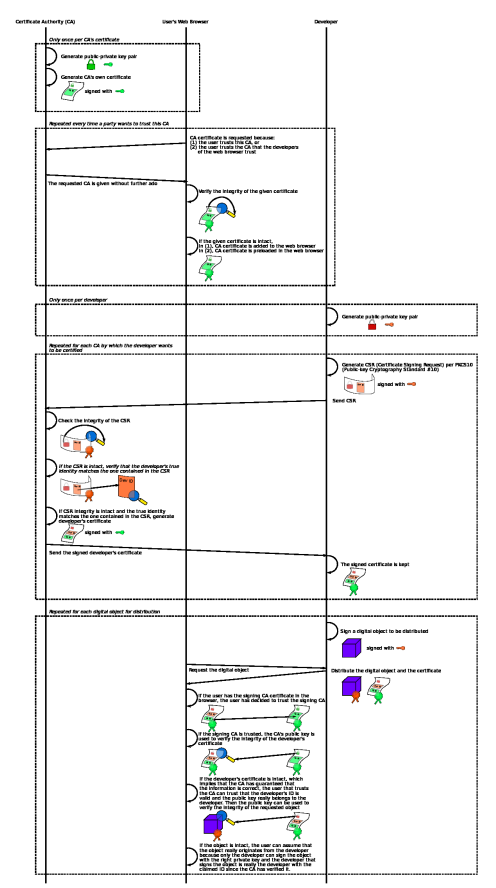


Diagram of an example usage of digital certificate

Operating systems

Certificates can be created for Unix-based servers with tools such as OpenSSL's `ca` [1] command,[2] or SuSE's `gensslcert`. These may be used to issue unmanaged certificates, certification authority (CA) certificates for managing other certificates, and user or computer certificate requests to be signed by the CA, as well as a number of other certificate related functions.

Similarly, Windows Server contains a CA as part of Certificate Services for the creation of digital certificates. In Windows Server 2008 and later the CA may be installed as part of Active Directory Certificate Services. The CA is used to manage and centrally issue certificates to users or computers. Microsoft also provides a number of different certificate utilities, such as SelfSSL.exe for creating unmanaged certificates, and Certreq.exe for creating and submitting certificate requests to be signed by the CA, and certutil.exe for a number of other certificate related functions.

Mac OS X comes with the Keychain Access program, which is able to perform various certificate-related services.

Contents of a typical digital certificate

See also: X.509 § Structure of a certificate

- **Serial Number**: Used to uniquely identify the certificate.
- **Subject**: The person, or entity identified.
- **Signature Algorithm**: The algorithm used to create the signature.
- **Signature**: The actual signature to verify that it came from the issuer.
- **Issuer**: The entity that verified the information and issued the certificate.
- **Valid-From**: The date the certificate is first valid from.
- **Valid-To**: The expiration date.
- **Key-Usage**: Purpose of the public key (e.g. encipherment, signature, certificate signing...).
- **Public Key**: The public key.
- **Thumbprint Algorithm**: The algorithm used to hash the public key certificate.
- **Thumbprint** (also known as fingerprint): The hash itself, used as an abbreviated form of the public key certificate.

Classification

Vendor defined classes

VeriSign uses the concept of classes for different types of digital certificates:[3]

- Class 1 for individuals, intended for email.
- Class 2 for organizations, for which proof of identity is required.
- Class 3 for servers and software signing, for which independent verification and checking of identity and authority is done by the issuing certificate authority.
- Class 4 for online business transactions between companies.
- Class 5 for private organizations or governmental security.

Other vendors may choose to use different classes or no classes at all as this is not specified in the PKI standards.

Usage in the European Union

The EU Directive 1999/93/EC on "a Community framework for electronic signatures" defines the term *qualified certificate* as a certificate which meets the requirements of Annex I and is provided by a certification service provider who fulfills the requirements of Annex II.

According to Annex I, qualified certificates must contain:

1. An indication that the certificate is issued as a qualified certificate
2. The identification of the certification service provider and the state in which it is established
3. The name of the signatory or a pseudonym, which shall be identified as such
4. Provision for a specific attribute of the signatory to be included if relevant, depending on the purpose for which the certificate is intended
5. Signature verification data which correspond to signature creation data under the control of the signatory
6. An indication of the beginning and end of the period of validity of the certificate
7. The identity code of the certificate
8. The advanced electronic signature of the certification service provider issuing it
9. Limitations on the scope of use of the certificate, if applicable
10. Limits on the value of transactions for which the certificate can be used, if applicable

Annex II requires certification service providers to:

1. Demonstrate the reliability necessary for providing certification services

2. Ensure the operation of a prompt and secure directory and a secure and immediate revocation service
3. Ensure that the date and time when a certificate is issued or revoked can be determined precisely
4. Verify, by appropriate means in accordance with national law, the identity and, if applicable, any specific attributes of the person to which a qualified certificate is issued
5. Employ personnel who possess the expert knowledge, experience, and qualifications necessary for the services provided, in particular competence at managerial level, expertise in electronic signature technology and familiarity with proper security procedures. They must also apply administrative and management procedures which are adequate and correspond to recognized standards.
6. Use trustworthy systems and products which are protected against modification and ensure the technical and cryptographic security of the process supported by them
7. Take measures against forgery of certificates, and, in cases where the certification service provider generates signature creation data, guarantee confidentiality during the process of generating such data
8. Maintain sufficient financial resources to operate in conformity with the requirements laid down in the Directive, in particular to bear the risk of liability for damages, for example, by obtaining appropriate insurance
9. Record all relevant information concerning a qualified certificate for an appropriate period of time, in particular for the purpose of providing evidence of certification for the purposes of legal proceedings. Such recording may be done electronically.
10. Not store or copy signature creation data of the person to whom the certification service provider provided key management services
11. Before entering into a contractual relationship with a person seeking a certificate to support their electronic signature, inform that person by a durable means of communication of the precise terms and conditions regarding the use of the certificate, including any limitations on its use, the existence of a voluntary accreditation scheme and procedures for complaints and dispute settlement. Such information, which may be transmitted electronically, must be in writing and in readily understandable language. Relevant parts of this information must also be made available on request to third parties relying on the certificate.
12. Use trustworthy systems to store certificates in a verifiable form so that:
    - Only authorized persons can make entries and changes
    - Information can be checked for authenticity
    - Certificates are publicly available for retrieval in only those cases for which the certificate holder's consent has been obtained
    - Any technical changes compromising these security requirements are apparent to the operator

Certificates and web site security

The most common use of certificates is for HTTPS-based web sites. A web browser validates that a TLS (Transport Layer Security) web server is authentic, so that the user can feel secure that his/her interaction with the web site has no eavesdroppers and that the web site is who it claims to be. This security is important for electronic commerce. In practice, a web site operator obtains a certificate by applying to a certificate provider (a CA that presents as a commercial retailer of certificates) with a certificate signing request. The certificate request is an electronic document that contains the web site name, contact email address, company information and the public key (for security reasons the private key is not part of the request and is not sent to the certificate authority). The certificate provider signs the request, thus producing a public certificate. During web browsing, this public certificate is served to any web browser that connects to the web site and proves to the web browser that the provider believes it has issued a certificate to the owner of the web site.

Before issuing a certificate, the certificate provider will request the contact email address for the web site from a public domain name registrar, and check that published address against the email address supplied in the certificate request. Therefore, an https web site is only secure to the extent that the end user can be sure that the web site is

operated by someone in contact with the person who registered the domain name.

As an example, when a user connects to `https://www.example.com/` with their browser, if the browser does not give any certificate warning message, then the user can be theoretically sure that interacting with `https://www.example.com/` is equivalent to interacting with the entity in contact with the email address listed in the public registrar under "example.com", even though that email address may not be displayed anywhere on the web site. No other surety of any kind is implied. Further, the relationship between the purchaser of the certificate, the operator of the web site, and the generator of the web site content may be tenuous and is not guaranteed. At best, the certificate guarantees uniqueness of the web site, provided that the web site itself has not been compromised (hacked) or the certificate issuing process subverted.

A certificate provider can opt to issue three types of certificates, each requiring its own degree of vetting rigor. In order of increasing rigor (and naturally, cost) they are: Domain Validation, Organization Validation and Extended Validation. These rigors are loosely agreed upon by voluntary participants in the CA/Browser Forum.

Domain Validation

A certificate provider will issue a Validation (DV) class certificate to a purchaser if the purchaser can demonstrate one straightforward vetting criterion: the right to administratively manage the domain name in question. For example, a domain name registrar might sell (or more accurately, resell) DV certificates through their domain name management system.

Organization Validation

A certificate provider will issue an Organization Validation (OV) class certificate to a purchaser if the purchaser can meet two criteria: the right to administratively manage the domain name in question, and perhaps, the organization's actual existence as a legal entity. A certificate provider publishes its OV vetting criteria through its Certificate Policy.

Extended Validation

To acquire an Extended Validation (EV) certificate, the purchaser must persuade the certificate provider of its legitimacy by surviving a battery complex vetting criteria, the majority of which are manually performed. As with OV certificates, a certificate provider publishes its EV vetting criteria through its Certificate Policy.

Browsers will generally offer users a special visual indication when a site presents an EV certificate. For example, it might change the background color of the URL bar from neutral to green. In this way, the user can decide whether or not to more readily trust the site as being legitimate.

Weaknesses

A web browser will give no warning to the user if a web site suddenly presents a different certificate, even if that certificate has a lower number of key bits, even if it has a different provider, and even if the previous certificate had an expiry date far into the future.Wikipedia:Citation needed However a change from an EV certificate to a non-EV certificate will be apparent as the green bar will no longer be displayed. Where certificate providers are under the jurisdiction of governments, those governments may have the freedom to order the provider to generate any certificate, such as for the purposes of law enforcement. Subsidiary wholesale certificate providers also have the freedom to generate any certificate.

All web browsers come with an extensive built-in list of trusted root certificates, many of which are controlled by organizations that may be unfamiliar to the user. Each of these organizations is free to issue any certificate for any web site and have the guarantee that web browsers that include its root certificates will accept it as genuine. In this instance, end users must rely on the developer of the browser software to manage its built-in list of certificates and

on the certificate providers to behave correctly and to inform the browser developer of problematic certificates. While uncommon, there have been incidents, in which fraudulent certificates have been issued: in some cases, the browsers have detected the fraud; in others, some time passed before browser developers removed these certificates from their software.

The list of built-in certificates is also not limited to those provided by the browser developer: users (and to a degree applications) are free to extend the list for special purposes such as for company intranets. This means that if someone gains access to a machine and can install a new root certificate in the browser, that browser will recognize websites that use the inserted certificate as legitimate.

For provable security, this reliance on something external to the system has the consequence that any public key certification scheme has to rely on some special setup assumption, such as the existence of a certificate authority.[4]

Usefulness versus unsecured web sites

In spite of the limitations described above, certificate-authenticated SSL is considered mandatory by all security guidelines whenever a web site hosts confidential information or performs material transactions. This is because, in practice, in spite of the serious flaws described above, web sites secured by public key certificates are still more secure than unsecured http:// web sites.

References

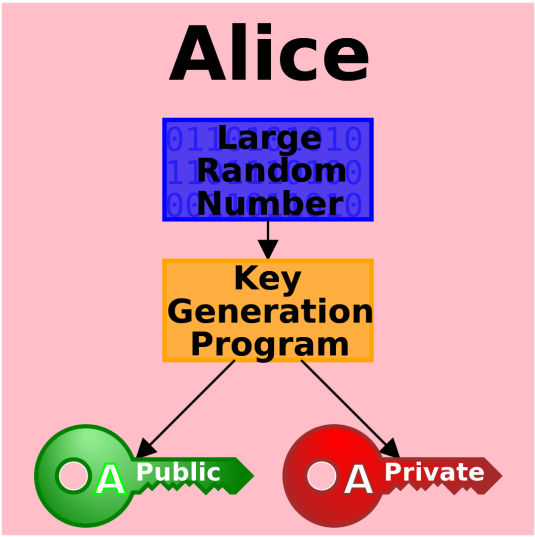
[1] http://www.openssl.org/docs/apps/ca.html
[2] OpenSSL: Documentation ca(1) (http://www.openssl.org/docs/apps/ca.html)
[3] VeriSign Class definitions (https://www.verisign.com/support/roots.html)
[4] Ran Canetti: Universally Composable Signature, Certification, and Authentication. CSFW 2004, http://eprint.iacr.org/2003/239

External links

- RFC 5280 (http://www.ietf.org/rfc/rfc5280.txt) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
- SSLTools Certificate Lookup (http://www.ssltools.com/certificate_lookup/www.wikipedia.org) Wikipedia.org Public key certificate details

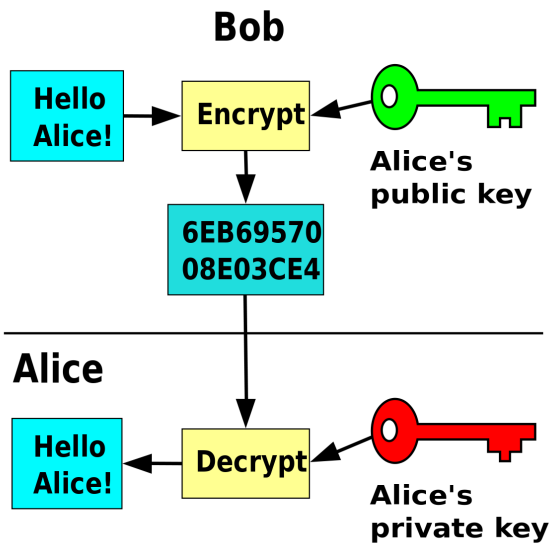
Public-key cryptography

Public-key cryptography, also known as **asymmetric cryptography**, is a class of cryptographic algorithms which require two separate keys, one of which is *secret* (or *private*) and one of which is *public*. Although different, the two parts of this key pair are mathematically linked. The public key is used to encrypt plaintext or to verify a digital signature; whereas the private key is used to decrypt ciphertext or to create a digital signature. The term "asymmetric" stems from the use of different keys to perform these opposite functions, each the inverse of the other – as contrasted with conventional ("symmetric") cryptography which relies on the same key to perform both.



An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

Public-key algorithms are based on mathematical problems which currently admit no efficient solution that are inherent in certain integer factorization, discrete logarithm, and elliptic curve relationships. It is computationally easy for a user to generate their own public and private key-pair and to use them for encryption and decryption. The strength lies in the fact that it is "impossible" (computationally infeasible) for a properly generated private key to be determined from its corresponding public key. Thus the public key may be published without compromising security, whereas the private key must not be revealed to anyone not authorized to read messages or perform digital signatures. Public key algorithms, unlike symmetric key algorithms, do *not* require a secure initial exchange of one (or more) secret keys between the parties.

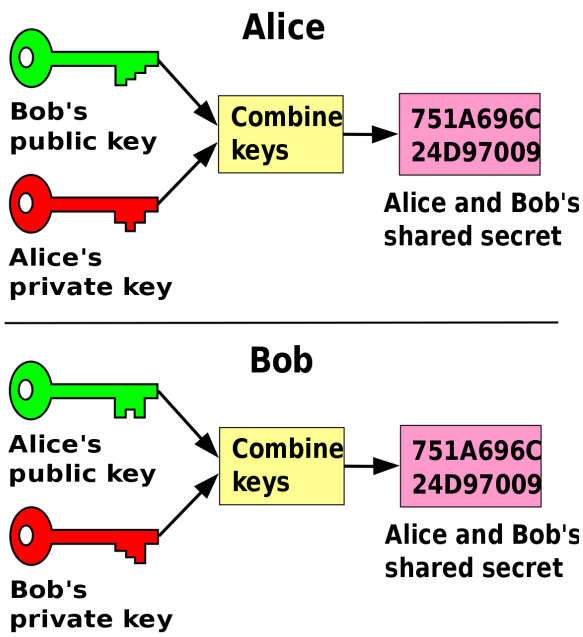


In an asymmetric key encryption scheme, anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt. Security depends on the secrecy of the private key.

Message authentication involves processing a message with a private key to produce a digital signature. Thereafter anyone can verify this signature by processing the signature value with the signer's corresponding public key and comparing that result with the message. Success confirms the message is unmodified since it was signed, and – presuming the signer's private key has remained secret to the signer – that the signer, and no one else, intentionally performed the signature operation. In practice, typically only a hash or digest of the message, and not the message itself, is encrypted as the signature.

Public-key algorithms are fundamental security ingredients in cryptosystems, applications and protocols. They underpin such Internet standards as Transport Layer Security (TLS), PGP, and GPG. Some public key algorithms provide key distribution and secrecy (e.g., Diffie–Hellman key exchange), some provide digital signatures (e.g., Digital Signature Algorithm), and some provide both (e.g., RSA).

Public-key cryptography finds application in, amongst others, the IT security discipline information security. Information security (IS) is concerned with all aspects of protecting electronic information assets against security threats. Public-key cryptography is used as a method of assuring the confidentiality, authenticity and non-repudiability of electronic communications and data storage.



In the Diffie–Hellman key exchange scheme, each party generates a public/private key pair and distributes the public key. After obtaining an authentic copy of each other's public keys, Alice and Bob can compute a shared secret offline. The shared secret can be used, for instance, as the key for a symmetric cipher.

Understanding

Public-key cryptography is often used to secure electronic communication over an open networked environment such as the internet. Open networked environments are susceptible to a variety of communication security problems such as man-in-the-middle attacks and other security threats. Sending a secure communication means that the communication being sent must not be readable during transit (preserving confidentiality), the communication must not be modified during transit (preserving the integrity of the communication) and to enforce non-repudiation or non-denial of the sending of the communication. Combining public-key cryptography with an Enveloped Public Key Encryption (EPKE) method, allows for the secure sending of a communication over an open networked environment.

The distinguishing technique used in public-key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same as the key used to decrypt it. Each user has a pair of cryptographic keys – a **public encryption key** and a **private decryption key**. Similarly, a key pair used for digital signatures consists of a **private signing key** and a **public verification key**. The public key is widely distributed, while the private key is known only to its proprietor. The keys are related mathematically, but the parameters are chosen so that calculating the private key from the public key is either impossible or prohibitively expensive.

In contrast, symmetric-key algorithms – variations of which have been used for thousands of years – use a *single* secret key, which must be shared and kept private by both the sender and the receiver, for both encryption and decryption. To use a symmetric encryption scheme, the sender and receiver must securely share a key in advance.

Because symmetric key algorithms are nearly always much less computationally intensive than asymmetric ones, it is common to exchange a key using a key-exchange algorithm, then transmit data using that key and a symmetric key algorithm. PGP and the SSL/TLS family of schemes use this procedure, and are thus called *hybrid cryptosystems*.

Description

There are two main uses for public-key cryptography:

- *Public-key encryption*, in which a message is encrypted with a recipient's public key. The message cannot be decrypted by anyone who does not possess the matching private key, who is thus presumed to be the owner of that key and the person associated with the public key. This is used in an attempt to ensure confidentiality.
- *Digital signatures*, in which a message is signed with the sender's private key and can be verified by anyone who has access to the sender's public key. This verification proves that the sender had access to the private key, and therefore is likely to be the person associated with the public key. This also ensures that the message has not been tampered, as any manipulation of the message will result in changes to the encoded message digest, which otherwise remains unchanged between the sender and receiver.

An analogy to public-key encryption is that of a locked mail box with a mail slot. The mail slot is exposed and accessible to the public – its location (the street address) is, in essence, the public key. Anyone knowing the street address can go to the door and drop a written message through the slot. However, only the person who possesses the key can open the mailbox and read the message.

An analogy for digital signatures is the sealing of an envelope with a personal wax seal. The message can be opened by anyone, but the presence of the unique seal authenticates the sender.

A central problem with the use of public-key cryptography is confidence/proof that a particular public key is authentic, in that it is correct and belongs to the person or entity claimed, and has not been tampered with or replaced by a malicious third party. The usual approach to this problem is to use a public-key infrastructure (PKI), in which one or more third parties – known as certificate authorities – certify ownership of key pairs. PGP, in addition to being a certificate authority structure, has used a scheme generally called the "web of trust", which decentralizes such authentication of public keys by a central mechanism, and substitutes individual endorsements of the link between user and public key. To date, no fully satisfactory solution to the "public key authentication problem" has been found.Wikipedia:Citation needed

History

During the early history of cryptography, two parties would rely upon a key that they would exchange between themselves by means of a secure, but non-cryptographic, method. For example, a face-to-face meeting or an exchange, via a trusted courier, could be used. This key, which both parties kept absolutely secret, could then be used to exchange encrypted messages. A number of significant practical difficulties arise with this approach to distributing keys. Public-key cryptography addresses these drawbacks so that users can communicate securely over a public channel without having to agree upon a shared key beforehand.

In 1874, a book by William Stanley Jevons[1] described the relationship of one-way functions to cryptography, and went on to discuss specifically the factorization problem used to create the trapdoor function in the RSA system. In July 1996, one observer commented on the Jevons book in this way:

> In his book *The Principles of Science: A Treatise on Logic and Scientific Method*, published in 1874, Jevons observed there are many situations where the "direct" operation is relatively easy, but the "inverse" operation is significantly more difficult. One example mentioned briefly is that enciphering (encryption) is easy while deciphering (decryption) is not. In the same section of Chapter 7: Introduction titled "Induction an Inverse Operation", much more attention is devoted to the principle that multiplication of integers is easy, but finding the (prime) factors of the product is much harder. Thus, Jevons anticipated a key feature of the RSA Algorithm for public key cryptography, although he certainly did not invent the concept of public key cryptography.

In 1997, it was publicly disclosed that asymmetric key algorithms were secretly developed by James H. Ellis, Clifford Cocks, and Malcolm Williamson at the Government Communications Headquarters (GCHQ) in the UK in

1973. In the public disclosure it was claimed that these researchers had independently developed Diffie–Hellman key exchange, and a special case of RSA. The GCHQ cryptographers referred to the technique as "non-secret encryption". This work was named an IEEE Milestone in 2010.

An asymmetric-key cryptosystem was published in 1976 by Whitfield Diffie and Martin Hellman who, influenced by Ralph Merkle's work on public-key distribution, disclosed a method of public-key agreement. This method of key exchange, which uses exponentiation in a finite field, came to be known as Diffie–Hellman key exchange. This was the first published practical method for establishing a shared secret-key over an authenticated (but not private) communications channel without using a prior shared secret. Merkle's "public-key-agreement technique" became known as Merkle's Puzzles, and was invented in 1974 and published in 1978.

A generalization of Cocks's scheme was independently invented in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman, all then at MIT. The latter authors published their work in 1978, and the algorithm appropriately came to be known as RSA. RSA uses exponentiation modulo a product of two very large primes, to encrypt and decrypt, performing both public key encryption and public key digital signature. Its security is connected to the (presumed) extreme difficulty of factoring large integers, a problem for which there is no known efficient (i.e. practicably fast) general technique. In 1979, Michael O. Rabin published a related cryptosystem that is probably secure, at least as long as the factorization of the public key remains difficult – it remains an assumption that RSA also enjoys this security.

Since the 1970s, a large number and variety of encryption, digital signature, key agreement, and other techniques have been developed in the field of public-key cryptography. The ElGamal cryptosystem, invented by Taher ElGamal relies on the similar and related high level of difficulty of the discrete logarithm problem, as does the closely related DSA, which was developed at the US National Security Agency (NSA) and published by NIST as a proposed standard. The introduction of elliptic curve cryptography by Neal Koblitz and Victor Miller, independently and simultaneously in the mid-1980s, has yielded new public-key algorithms based on the discrete logarithm problem. Although mathematically more complex, elliptic curves provide smaller key sizes and faster operations for approximately equivalent estimated security.

Security

Some encryption schemes can be proven secure on the basis of the presumed difficulty of a mathematical problem, such as factoring the product of two large primes or computing discrete logarithms. Note that "secure" here has a precise mathematical meaning, and there are multiple different (meaningful) definitions of what it means for an encryption scheme to be "secure". The "right" definition depends on the context in which the scheme will be deployed.

The most obvious application of a public key encryption system is confidentiality – a message that a sender encrypts using the recipient's public key can be decrypted only by the recipient's paired private key. This assumes, of course, that no flaw is discovered in the basic algorithm used.

Another type of application in public-key cryptography is that of digital signature schemes. Digital signature schemes can be used for sender authentication and non-repudiation. In such a scheme, a user who wants to send a message computes a digital signature for this message, and then sends this digital signature (together with the message) to the intended receiver. Digital signature schemes have the property that signatures can be computed only with the knowledge of the correct private key. To verify that a message has been signed by a user and has not been modified, the receiver needs to know only the corresponding public key. In some cases (e.g. RSA), a single algorithm can be used to both encrypt and create digital signatures. In other cases (e.g. DSA), each algorithm can only be used for one specific purpose.

To achieve both authentication and confidentiality, the sender should include the recipient's name in the message, sign it using his private key, and then encrypt both the message and the signature using the recipient's public key.

These characteristics can be used to construct many other (sometimes surprising) cryptographic protocols and applications, such as digital cash, password-authenticated key agreement, multi-party key agreement, time-stamping services, non-repudiation protocols, etc.

Practical considerations

Enveloped Public Key Encryption

Enveloped Public Key Encryption (EPKE) is the method of applying public-key cryptography and ensuring that an electronic communication is transmitted confidentially, has the contents of the communication protected against being modified (communication integrity) and cannot be denied from having been sent (non-repudiation). This is often the method used when securing communication on an open networked environment such by making use of the Transport Layer Security (TLS) or Secure Sockets Layer (SSL) protocols.

EPKE consists of a two stage process that includes both Forward Public Key Encryption (FPKE) also known as symmetric encryption and Inverse Public Key Encryption (IPKE). Both Forward Public Key Encryption and Inverse Public Key encryption make up the foundation of Enveloped Public Key Encryption (these two processes are described in full in their own sections).

For EKPE (asymmetric encryption) to work effectively, it is required that:

- Every participant in the communication has their own unique pair of keys. The first key that is required is a public key and the second key that is required is a private key.
- Each person's own private and public keys must be mathematically related where the private key is used to decrypt a communication sent using a public key and vice versa. The RSA algorithm is a well known asymmetric encryption algorithm.
- The private key must be kept absolutely private by the owner though the public key can be published in a public directory such as with a certification authority.

To send a message using EPKE, the sender of the message first encrypts the message using their own private key, this ensures non-repudiation of the message (Inverse Public Key Encryption). The sender then encrypts their digitally signed message using the receivers public key (Forward Public Key Encryption) thus applying a digital envelope to the message. This step ensures confidentiality during the transmission of the message. The receiver of the message then uses their private key to decrypt the message thus removing the digital envelope and then uses the senders public key to decrypt the senders digital signature. At this point, if the message has been unaltered during transmission, the message will be clear to the receiver.

Due to the computationally complex nature of the RSA asymmetric encryption algorithm, the time taken to encrypt a large documents or files to be transmitted can take an increased amount of time to complete. To speed up the process of transmission, instead of applying the senders digital signature to the large documents or files, the sender can rather hash the documents or files and then encrypt the generated hash key with their digital signature therefore enforcing non-repudiation. Hashing is a much faster computation to complete as opposed to using the RSA encryption algorithm alone. The sender would then encrypt the newly generated hash key and encrypt the original documents or files with the receiver's public key. The transmission would then take place securely and with confidentiality and non-repudiation still intact. The receiver would then decrypt both the encrypted hash key and the encrypted documents or files with their private key. Finally, to ensure that the message being sent was in fact sent by the appropriate sender, the receiver would then use the senders public key to decrypt the hash key and then the receiver would hash the documents or files using the same hashing algorithm as the sender and see if the hash key that is generated matches the hash key generated by the sender. If both hash keys match, the documents or files are the same as the sender's and non-repudiation has been preserved.

Note: The sender and receiver do not usually carry out the process mentioned above manually though rather rely on sophisticated software to automatically complete the EPKE process.

Forward Public Key Encryption

The goal of Forward Public Key Encryption (FPKE) encryption is to ensure that the communication being sent is kept confidential during transit.

To send a message using FPKE, the sender of the message uses the public key of the receiver to encrypt the contents of the message. The encrypted message is then transmitted electronically to the receiver and the receiver can then use their own matching private key to decrypt the message.

The encryption process of using the receivers public key is useful for preserving the confidentiality of the message as only the receiver has the matching private key to decrypt the message. Therefore, the sender of the message cannot decrypt the message once it has been encrypted using the receivers public key. However, FKPE does not address the problem of non-repudiation, as the message could have been sent by anyone that has access to the receivers public key.

Inverse Public Key Encryption

The goal of Inverse Public Key Encryption (IPKE) is to ensure that the sender of the communication that is being sent is known to the receiver and that the sender of the message cannot refute that the message that they have sent was not sent by them. Therefore, the purpose of IPKE is to ensure the non-repudiation of the message being sent. This is useful in a practical setting where a sender wishes to make an electronic purchase of shares and the receiver wants to confirm that it was indeed the actual sender requesting the purchase and not someone else. IPKE is also known as a digital signature.

To send a message using IPKE, the message is encrypted using the senders private key which serves as the senders digital signature. The digitally "signed" and encrypted message is then sent to the receiver who can then use the senders public key to decrypt the message and reveal the original contents.

IPKE is useful for applying ones digital signature to a message thus enforcing non-repudiation however, when the message is transmitted it is possible for the message to be intercepted by an unauthorized person and the unauthorized person could then use the senders public key to decrypt the contents of the message thus not enforcing the confidentiality of the message being sent.

Certification Authority

In order for Enveloped Public Key Encryption to be as secure as possible, there needs to be a "gatekeeper" of public and private keys, or else anyone could publish their public key and masquerade as the intended sender of a communication. This digital key "gatekeeper" is known as a certification authority. A certification authority is a trusted third party that can issue public and private keys thus certifying public keys.

A postal analogy

An analogy that can be used to understand the advantages of an asymmetric system is to imagine two people, Alice and Bob, who are sending a secret message through the public mail. In this example, Alice wants to send a secret message to Bob, and expects a secret reply from Bob.

With a symmetric key system, Alice first puts the secret message in a box, and locks the box using a padlock to which she has a key. She then sends the box to Bob through regular mail. When Bob receives the box, he uses an identical copy of Alice's key (which he has somehow obtained previously, maybe by a face-to-face meeting) to open the box, and reads the message. Bob can then use the same padlock to send his secret reply.

In an asymmetric key system, Bob and Alice have separate padlocks. First, Alice asks Bob to send his open padlock to her through regular mail, keeping his key to himself. When Alice receives it she uses it to lock a box containing her message, and sends the locked box to Bob. Bob can then unlock the box with his key and read the message from Alice. To reply, Bob must similarly get Alice's open padlock to lock the box before sending it back to her.

The critical advantage in an asymmetric key system is that Bob and Alice never need to send a copy of their keys to each other. This prevents a third party – perhaps, in this example, a corrupt postal worker that will open unlocked boxes – from copying a key while it is in transit, allowing the third party to spy on all future messages sent between Alice and Bob. So, in the public key scenario, Alice and Bob need not trust the postal service as much. In addition, if Bob were careless and allowed someone else to copy *his* key, Alice's messages to Bob would be compromised, but Alice's messages to other people would remain secret, since the other people would be providing different padlocks for Alice to use.

Another kind of asymmetric key system, called a three-pass protocol, requires neither party to even touch the other party's padlock (or key); Bob and Alice have separate padlocks. First, Alice puts the secret message in a box, and locks the box using a padlock to which only she has a key. She then sends the box to Bob through regular mail. When Bob receives the box, he adds his own padlock to the box, and sends it back to Alice. When Alice receives the box with the two padlocks, she removes her padlock and sends it back to Bob. When Bob receives the box with only his padlock on it, Bob can then unlock the box with his key and read the message from Alice. Note that, in this scheme, the order of decryption is NOT the same as the order of encryption – this is only possible if commutative ciphers are used. A commutative cipher is one in which the order of encryption and decryption is interchangeable, just as the order of multiplication is interchangeable (i.e. `A*B*C = A*C*B = C*B*A`). This method is secure for certain choices of commutative ciphers, but insecure for others (e.g. a simple `XOR`). For example, let $E_1()$ and $E_2()$ be two encryption functions, and let "`M`" be the message so that if Alice encrypts it using $E_1()$ and sends $E_1(M)$ to Bob. Bob then again encrypts the message as $E_2(E_1(M))$ and sends it to Alice. Now, Alice decrypts $E_2(E_1(M))$ using $E_1()$. Alice will now get $E_2(M)$, meaning when she sends this again to Bob, he will be able to decrypt the message using $E_2()$ and get "`M`". Although none of the keys were ever exchanged, the message "`M`" may well be a key (e.g. Alice's Public key). This three-pass protocol is typically used during key exchange.

Actual algorithms: two linked keys

Not all asymmetric key algorithms operate in precisely this fashion. The most common ones have the property that Alice and Bob each own *two* keys, one for encryption and one for decryption. In a secure asymmetric key encryption scheme, the private key should not be deducible from the public key. This is known as public-key encryption, since an encryption key can be published without compromising the security of messages encrypted with that key.

In the analogy above, Bob might publish instructions on how to make a lock ("public key"). However, the workings of the lock are such that it is impossible (so far as is known) to deduce from the instructions given just exactly how to make a key that will open that lock (e.g. a "private key"). Those wishing to send messages to Bob must use the public key to encrypt the message, then Bob can use his private key to decrypt it.

Another example has Alice and Bob each choosing a key at random, and then contacting each other to compare the depth of each notch on their keys. Having determined the difference, a locked box is built with a special lock that has each pin inside divided into 2 pins, matching the numbers of their keys. Now the box will be able to be opened with either key, and Alice and Bob can exchange messages inside the box in a secure fashion.

Weaknesses

Of course, there is a possibility that someone could "pick" Bob's or Alice's lock. Among symmetric key encryption algorithms, only the one-time pad can be proven to be secure against any adversary – no matter how much computing power is available. However, there is no public-key scheme with this property, since all public-key schemes are susceptible to a "brute-force key search attack". Such attacks are impractical if the amount of computation needed to succeed – termed the "work factor" by Claude Shannon – is out of reach of all potential attackers. In many cases, the work factor can be increased by simply choosing a longer key. But other algorithms may have much lower work factors, making resistance to a brute-force attack irrelevant. Some special and specific algorithms have been developed to aid in attacking some public key encryption algorithms – both RSA and ElGamal

encryption have known attacks that are much faster than the brute-force approach. These factors have changed dramatically in recent decades, both with the decreasing cost of computing power and with new mathematical discoveries.

Aside from the resistance to attack of a particular key pair, the security of the certification hierarchy must be considered when deploying public key systems. Some certificate authority − usually a purpose-built program running on a server computer − vouches for the identities assigned to specific private keys by producing a digital certificate. Public key digital certificates are typically valid for several years at a time, so the associated private keys must be held securely over that time. When a private key used for certificate creation higher in the PKI server hierarchy is compromised, or accidentally disclosed, then a "man-in-the-middle attack" is possible, making any subordinate certificate wholly insecure.

Major weaknesses have been found for several formerly promising asymmetric key algorithms. The 'knapsack packing' algorithm was found to be insecure after the development of a new attack. Recently, some attacks based on careful measurements of the exact amount of time it takes known hardware to encrypt plain text have been used to simplify the search for likely decryption keys (see "side channel attack"). Thus, mere use of asymmetric key algorithms does not ensure security. A great deal of active research is currently underway to both discover, and to protect against, new attack algorithms.

Another potential security vulnerability in using asymmetric keys is the possibility of a "man-in-the-middle" attack, in which the communication of public keys is intercepted by a third party (the "man in the middle") and then modified to provide different public keys instead. Encrypted messages and responses must also be intercepted, decrypted, and re-encrypted by the attacker using the correct public keys for different communication segments, in all instances, so as to avoid suspicion. This attack may seem to be difficult to implement in practice, but it is not impossible when using insecure media (e.g. public networks, such as the Internet or wireless forms of communications) − for example, a malicious staff member at Alice or Bob's Internet Service Provider (ISP) might find it quite easy to carry out. In the earlier postal analogy, Alice would have to have a way to make sure that the lock on the returned packet really belongs to Bob before she removes her lock and sends the packet back. Otherwise, the lock could have been put on the packet by a corrupt postal worker pretending to be Bob, so as to fool Alice.

One approach to prevent such attacks involves the use of a certificate authority, a trusted third party responsible for verifying the identity of a user of the system. This authority issues a tamper-resistant, non-spoofable digital certificate for the participants. Such certificates are signed data blocks stating that this public key belongs to that person, company, or other entity. This approach also has its weaknesses − for example, the certificate authority issuing the certificate must be trusted to have properly checked the identity of the key-holder, must ensure the correctness of the public key when it issues a certificate, and must have made arrangements with all participants to check all their certificates before protected communications can begin. Web browsers, for instance, are supplied with a long list of "self-signed identity certificates" from PKI providers − these are used to check the *bona fides* of the certificate authority and then, in a second step, the certificates of potential communicators. An attacker who could subvert any single one of those certificate authorities into issuing a certificate for a bogus public key could then mount a "man-in-the-middle" attack as easily as if the certificate scheme were not used at all. Despite its theoretical and potential problems, this approach is widely used. Examples include SSL and its successor, TLS, which are commonly used to provide security for web browsers, for example, so that they might be used to securely send credit card details to an online store.

Computational cost

The public key algorithms known thus far are relatively computationally costly compared with most symmetric key algorithms of apparently equivalent security. The difference factor is the use of typically quite large keys. This has important implications for their practical use. Most are used in hybrid cryptosystems for reasons of efficiency – in such a cryptosystem, a shared secret key ("session key") is generated by one party, and this much briefer session key is then encrypted by each recipient's public key. Each recipient then uses the corresponding private key to decrypt the session key. Once all parties have obtained the session key, they can use a much faster symmetric algorithm to encrypt and decrypt messages. In many of these schemes, the session key is unique to each message exchange, being pseudo-randomly chosen for each message.

Associating public keys with identities

The binding between a public key and its "owner" must be correct, or else the algorithm may function perfectly and yet be entirely insecure in practice. As with most cryptography applications, the protocols used to establish and verify this binding are critically important. Associating a public key with its owner is typically done by protocols implementing a public key infrastructure – these allow the validity of the association to be formally verified by reference to a trusted third party in the form of either a hierarchical certificate authority (e.g., X.509), a local trust model (e.g. SPKI), or a web of trust scheme, like that originally built into PGP and GPG, and still to some extent usable with them. Whatever the cryptographic assurance of the protocols themselves, the association between a public key and its owner is ultimately a matter of subjective judgment on the part of the trusted third party, since the key is a mathematical entity, while the owner – and the connection between owner and key – are not. For this reason, the formalism of a public key infrastructure must provide for explicit statements of the policy followed when making this judgment. For example, the complex and never fully implemented X.509 standard allows a certificate authority to identify its policy by means of an object identifier, which functions as an index into a catalog of registered policies. Policies may exist for many different purposes, ranging from anonymity to military classifications.

Relation to real world events

A public key will be known to a large and, in practice, unknown set of users. All events requiring revocation or replacement of a public key can take a long time to take full effect with all who must be informed (i.e. all those users who possess that key). For this reason, systems that must react to events in real time (e.g., safety-critical systems or national security systems) should not use public-key encryption without taking great care. There are four issues of interest:

Privilege of key revocation

A malicious (or erroneous) revocation of some (or all) of the keys in the system is likely, or in the second case, certain, to cause a complete failure of the system. If public keys can be revoked individually, this is a possibility. However, there are design approaches that can reduce the practical chance of this occurring. For example, by means of certificates, we can create what is called a "compound principal" – one such principal could be "Alice and Bob have Revoke Authority". Now, only Alice and Bob (in concert) can revoke a key, and neither Alice nor Bob can revoke keys alone. However, revoking a key now requires both Alice *and* Bob to be available, and this creates a problem of reliability. In concrete terms, from a security point of view, there is now a "single point of failure" in the public key revocation system. A successful Denial of Service attack against either Alice or Bob (or both) will block a required revocation. In fact, any partition of authority between Alice and Bob will have this effect, regardless of how it comes about.

Because the principle allowing revocation authority for keys is very powerful, the mechanisms used to control it should involve **both** as many participants as possible (to guard against malicious attacks of this type), while at the

same time as few as possible (to ensure that a key can be revoked without dangerous delay). Public key certificates that include an expiration date are unsatisfactory in that the expiration date may not correspond with a real-world revocation need – but at least such certificates need not all be tracked down system-wide, nor must all users be in constant contact with the system at all times.

Distribution of a new key

After a key has been revoked, or when a new user is added to a system, a new key must be distributed in some predetermined manner. Assume that Carol's key has been **revoked** (e.g. by exceeding its expiration date, or because of a compromise of Carol's matching private key). Until a new key has been distributed, Carol is effectively "out of contact". No one will be able to send her messages without violating system protocols (i.e. without a valid public key, no one can encrypt messages to her), and messages from her cannot be signed, for the same reason. Or, in other words, the "part of the system" controlled by Carol is, in essence, unavailable. Security requirements have been ranked higher than system availability in such designs.

One could leave the power to create (and certify) keys (as well as to revoke them) in the hands of each user – the original PGP design did so – but this raises problems of user understanding and operation. For security reasons, this approach has considerable difficulties – if nothing else, some users will be forgetful, or inattentive, or confused. On the one hand, a message revoking a public key certificate should be spread as fast as possible, while on the other hand, parts of the system might be rendered inoperable *before* a new key can be installed. The time window can be reduced to zero by always issuing the new key together with the certificate that revokes the old one, but this requires co-location of authority to both revoke keys and generate new keys.

It is most likely a system-wide failure if the (possibly combined) principal that issues new keys fails by issuing keys improperly. This is an instance of a "common mutual exclusion" – a design can make the reliability of a system high, but only at the cost of system availability (and *vice versa*).

Spreading the revocation

Notification of a key certificate revocation must be spread to all those who might potentially hold it, and as rapidly as possible.

There are but two means of spreading information (i.e. a key revocation) in a distributed system: either the information is "pushed" to users from a central point (or points), or else it is "pulled" from a central point(or points) by the end users.

Pushing the information is the simplest solution, in that a message is sent to all participants. However, there is no way of knowing whether all participants will actually *receive* the message. If the number of participants is large, and some of their physical or network distances are great, then the probability of complete success (which is, in ideal circumstances, required for system security) will be rather low. In a partly updated state, the system is particularly vulnerable to "denial of service" attacks as security has been breached, and a vulnerability window will continue to exist as long as some users have not "gotten the word". Put another way, pushing certificate revocation messages is neither easy to secure, nor very reliable.

The alternative to pushing is pulling. In the extreme, all certificates contain all the keys needed to verify that the public key of interest (i.e. the one belonging to the user to whom one wishes to send a message, or whose signature is to be checked) is still valid. In this case, at least some use of the system will be blocked if a user cannot reach the verification service (i.e. one of the systems that can establish the current validity of another user's key). Again, such a system design can be made as reliable as one wishes, at the cost of lowering security – the more servers to check for the possibility of a key revocation, the longer the window of vulnerability.

Another trade-off is to use a somewhat less reliable, but more secure, verification service, but to include an expiration date for each of the verification sources. How long this "timeout" should be is a decision that requires a trade-off between availability and security that will have to be decided in advance, at the time of system design.

Recovery from a leaked key

Assume that the principal authorized to revoke a key has decided that a certain key must be revoked. In most cases, this happens after the fact – for instance, it becomes known that at some time in the past an event occurred that endangered a private key. Let us denote the time at which it is decided that the compromise occurred as *T*.

Such a compromise has two implications. First, messages encrypted with the matching public key (now or in the past) can no longer be assumed to be secret. One solution to avoid this problem is to use a protocol that has perfect forward secrecy. Second, signatures made with the *no longer trusted to be actually private key* after time *T* can no longer be assumed to be authentic without additional information (i.e. who, where, when, etc.) about the events leading up to the digital signature. These will not always be available, and so all such digital signatures will be less than credible. A solution to reduce the impact of leaking a private key of a signature scheme is to use timestamps.

Loss of secrecy and/or authenticity, even for a single user, has system-wide security implications, and a strategy for recovery must thus be established. Such a strategy will determine who has authority to, and under what conditions one must, revoke a public key certificate. One must also decide how to spread the revocation, and ideally, how to deal with all messages signed with the key since time *T* (which will rarely be known precisely). Messages sent to that user (which require the proper – now compromised – private key to decrypt) must be considered compromised as well, no matter when they were sent.

Examples

Examples of well-regarded asymmetric key techniques for varied purposes include:

- Diffie–Hellman key exchange protocol
- DSS (Digital Signature Standard), which incorporates the Digital Signature Algorithm
- ElGamal
- Various elliptic curve techniques
- Various password-authenticated key agreement techniques
- Paillier cryptosystem
- RSA encryption algorithm (PKCS#1)
- Cramer–Shoup cryptosystem
- YAK authenticated key agreement protocol

Examples of asymmetric key algorithms not widely adopted include:

- NTRUEncrypt cryptosystem
- McEliece cryptosystem

Examples of notable – yet insecure – asymmetric key algorithms include:

- Merkle–Hellman knapsack cryptosystem

Examples of protocols using asymmetric key algorithms include:

- GPG, an implementation of OpenPGP
- Internet Key Exchange
- PGP
- ZRTP, a secure VoIP protocol
- Secure Socket Layer, now codified as the IETF standard Transport Layer Security (TLS)
- SILC
- SSH
- Bitcoin
- Off-the-Record Messaging

Notes

[1] Jevons, William Stanley, *The Principles of Science: A Treatise on Logic and Scientific Method* (http://www.archive.org/stream/principlesofscie00jevorich#page/n166/mode/1up) p. 141, Macmillan & Co., London, 1874, 2nd ed. 1877, 3rd ed. 1879. Reprinted with a foreword by Ernst Nagel, Dover Publications, New York, NY, 1958.

References

- Frederick J. Hirsch (http://home.comcast.net/~fjhirsch/Papers/wwwj/). "SSL/TLS Strong Encryption: An Introduction" (http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html#cryptographictech). *Apache HTTP Server*. Retrieved 2013-04-17.. The first two sections contain a very good introduction to public-key cryptography.
- N. Ferguson; B. Schneier (2003). *Practical Cryptography*. Wiley. ISBN 0-471-22357-3.
- J. Katz; Y. Lindell (2007). *Introduction to Modern Cryptography*. CRC Press. ISBN 1-58488-551-3.
- A. J. Menezes; P. C. van Oorschot; S. A. Vanstone (1997). *Handbook of Applied Cryptography* (http://www.cacr.math.uwaterloo.ca/hac/). ISBN 0-8493-8523-7.
- IEEE 1363: Standard Specifications for Public-Key Cryptography (http://grouper.ieee.org/groups/1363)
- Christof Paar, Jan Pelzl, "Introduction to Public-Key Cryptography" (http://wiki.crypto.rub.de/Buch/movies.php), Chapter 6 of "Understanding Cryptography, A Textbook for Students and Practitioners". (companion web site contains online cryptography course that covers public-key cryptography), Springer, 2009.

External links

- Oral history interview with Martin Hellman (http://purl.umn.edu/107353), Charles Babbage Institute, University of Minnesota. Leading cryptography scholar Martin Hellman discusses the circumstances and fundamental insights of his invention of public key cryptography with collaborators Whitfield Diffie and Ralph Merkle at Stanford University in the mid-1970s.
- An account of how GCHQ kept their invention of PKE secret until 1997 (http://www.ladlass.com/intel/archives/010256.html)Wikipedia:Link rot

Route poisoning

Route poisoning is a method to prevent a router from sending packets through a route that has become invalid within computer networks. Distance-vector routing protocols in computer networks use route poisoning to indicate to other routers that a route is no longer reachable and should not be considered from their routing tables. Unlike the *split horizon with poison reverse*, route poisoning provides for sending updates with unreachable hop counts immediately to all the nodes in the network.

When the protocol detects an invalid route, all of the routers in the network are informed that the bad route has an infinite (∞) route metric. This makes all nodes on the invalid route seem infinitely distant, preventing any of the routers from sending packets over the invalid route.

Some distance-vector routing protocols, such as RIP, use a maximum hop count to determine how many routers the traffic must go through to reach the destination. Each route has a hop count number assigned to it which is incremented as the routing information is passed from router to router. A route is considered unreachable if the hop count exceeds the maximum allowed. Route poisoning is a method of quickly forgetting outdated routing information from other router's routing tables by changing its hop count to be unreachable (higher than the maximum number of hops allowed) and sending a routing update. In the case of RIP, the maximum hop count is 15, so to perform route poisoning on a route its hop count is changed to 16, deeming it unreachable, and a routing update is sent.

If these updates are lost, some nodes in the network would not be informed that a route is invalid, so they could attempt to send packets over the bad route and cause a problem known as a routing loop. Therefore route poisoning is used in conjunction with holddowns to keep update messages from falsely reinstating the validity of a bad route. This prevents routing loops, improving the overall efficiency of the network.

References

- The TCP-IP Guide, RIP Special Features For Resolving RIP Algorithm Problems, by Charles M. Kozierok [1]
- RFC 1058: Routing Information Protocol, by C. Hedrick, Rutgers University (June 1988) [2]

[1] http://www.tcpipguide.com/free/t_RIPSpecialFeaturesForResolvingRIPAlgorithmProblems-2.htm
[2] http://www.ietf.org/rfc/rfc1058.txt

Sybil attack

The **Sybil attack** in computer security is an attack wherein a reputation system is subverted by forging identities in peer-to-peer networks. It is named after the subject of the book *Sybil*, a case study of a woman diagnosed with dissociative identity disorder.[1] The name was suggested in or before 2002 by Brian Zill at Microsoft Research. The term "pseudospoofing" had previously been coined by L. Detweiler on the Cypherpunks mailing list and used in the literature on peer-to-peer systems for the same class of attacks prior to 2002, but this term did not gain as much influence as "Sybil attack".

Description

In a Sybil attack the attacker subverts the reputation system of a peer-to-peer network by creating a large number of pseudonymous identities, using them to gain a disproportionately large influence. A reputation system's vulnerability to a Sybil attack depends on how cheaply identities can be generated, the degree to which the reputation system accepts inputs from entities that do not have a chain of trust linking them to a trusted entity, and whether the reputation system treats all entities identically. Evidence shows large-scale Sybil attack can be carried out in a very cheap and efficient way in the realistic system like BitTorrent Mainline DHT.

An *entity* on a peer-to-peer network is a piece of software which has access to local resources. An entity advertises itself on the peer-to-peer network by presenting an *identity*. More than one identity can correspond to a single entity. In other words the mapping of identities to entities is many to one. Entities in peer-to-peer networks use multiple identities for purposes of redundancy, resource sharing, reliability and integrity. In peer-to-peer networks the identity is used as an abstraction so that a remote entity can be aware of identities without necessarily knowing the correspondence of identities to local entities. By default, each distinct identity is usually assumed to correspond to a distinct local entity. In reality many identities may correspond to the same local entity.

A faulty node or an adversary may present multiple identities to a peer-to-peer network in order to appear and function as multiple distinct nodes. After becoming part of the peer-to-peer network, the adversary may then overhear communications or act maliciously. By masquerading and presenting multiple identities, the adversary can control the network substantially.

In the context of (human) online communities, such multiple identities are known as sockpuppets.

Prevention

Validation techniques can be used to prevent Sybil attacks and dismiss masquerading hostile entities. A local entity may accept a remote identity based on a central authority which ensures a one-to-one correspondence between an identity and an entity and may even provide a reverse lookup. An identity may be validated either directly or indirectly. In direct validation the local entity queries the central authority to validate the remote identities. In indirect validation the local entity relies on already accepted identities which in turn vouch for the validity of the remote identity in question.

Identity-based validation techniques generally provide accountability at the expense of anonymity, which can be an undesirable tradeoff especially in online forums that wish to permit censorship-free information exchange and open discussion of sensitive topics. A validation authority can attempt to preserve users' anonymity by refusing to perform reverse lookups, but this approach makes the validation authority a prime target for attack. Alternatively, the authority can use some mechanism other than knowledge of a user's real identity - such as verification of an *unidentified* person's physical presence at a particular place and time - to enforce a one-to-one correspondence between online identities and real-world users.

Sybil prevention techniques based on the connectivity characteristics of social graphs can also limit the extent of damage that can be caused by a given sybil attacker while preserving anonymity, though these techniques cannot

prevent sybil attacks entirely, and may be vulnerable to widespread small-scale sybil attacks. Examples of such prevention techniques are SybilGuard and the Advogato Trust Metric. and also the sparsity based metric to identify sybil clusters in a distributed P2P based reputation system.

References

[1] http://www.npr.org/2011/10/20/141514464/real-sybil-admits-multiple-personalities-were-fake

- Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT (http://www.cs.helsinki.fi/u/lxwang/publications/P2P2013_13.pdf) by Liang Wang and Jussi Kangasharju. IEEE P2P 2013.
- Real-world sybil attacks in BitTorrent mainline DHT (http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6503215&url=http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6503215) by Liang Wang and Jussi Kangasharju. IEEE GLOBECOM 2012.
- Sybil Attacks Against Mobile Users: Friends and Foes to the Rescue (http://eprints.ucl.ac.uk/18812/1/18812.pdf) by Daniele Quercia and Stephen Hailes. IEEE INFOCOM 2010.
- On the establishment of distinct identities in overlay networks (http://thrackle.eas.asu.edu/users/goran/papers/dc06.pdf) by Rida A. Bazzi and Goran Konjevod
- An Offline Foundation for Online Accountable Pseudonyms (http://pdos.csail.mit.edu/papers/accountable-pseudonyms-socialnets08.pdf) by Bryan Ford and Jacob Strauss
- A Sybil-proof one-hop DHT (http://pdos.csail.mit.edu/papers/sybil-dht-socialnets08.pdf) by Chris Lesniewski-Laas
- The Sybil Attack in Sensor Networks: Analysis & Defenses (http://www.ece.cmu.edu/~adrian/projects/sybil.pdf)
- A Survey of Solutions to the Sybil Attack (https://gnunet.org/sites/default/files/Tech Report - A Survey of Solutions to the Sybil Attack.pdf)
- SybilGuard: Defending Against Sybil Attacks via Social Networks (http://www.csd.uoc.gr/~hy558/papers/sybilguard.pdf)
- SybilGuard: defending against sybil attacks via social networks (http://portal.acm.org/citation.cfm?id=1159945) by Haifeng Yu (requires an ACM account)
- On Network formation: Sybil attacks and Reputation systems (http://dimacs.rutgers.edu/Workshops/InformationSecurity/slides/gamesandreputation.pdf)
- Trust Transfer: Encouraging Self-Recommendations without Sybil Attack (https://www.cs.tcd.ie/publications/tech-reports/reports.05/TCD-CS-2005-47.pdf)
- A Survey of DHT Security Techniques (http://www.globule.org/publi/SDST_acmcs2009.html) by Guido Urdaneta, Guillaume Pierre and Maarten van Steen. ACM Computing surveys, 2009.
- An experiment on the weakness of reputation algorithms used in professional social networks: the case of Naymz (http://www.unibg.it/lazzari/doc/marco_lazzari_reputation_algorithms_professional_social_networks_naymz.htm) by Marco Lazzari. Proceedings of the IADIS International Conference e-Society 2010.

Topological space

In topology and related branches of mathematics, a **topological space** is a set of points, along with a set of neighbourhoods for each point, that satisfy a set of axioms relating points and neighbourhoods. The definition of a topological space relies only upon set theory and is the most general notion of a mathematical space that allows for the definition of concepts such as continuity, connectedness, and convergence. Other spaces, such as manifolds and metric spaces, are specializations of topological spaces with extra structures or constraints. Being so general, topological spaces are a central unifying notion and appear in virtually every branch of modern mathematics. The branch of mathematics that studies topological spaces in their own right is called point-set topology or general topology.

Definition

Main article: Characterizations of the category of topological spaces

The utility of the notion of a topology is shown by the fact that there are several equivalent definitions of this structure. Thus one chooses the axiomatisation suited for the application. The most commonly used, and the most elegant, is that in terms of *open sets*, but the most intuitive is that in terms of *neighbourhoods* and so we give this first. Note: A variety of more axiomatisations of topological spaces are listed in the Exercises of the book by Vaidyanathaswamy.

Neighbourhoods definition

This axiomatization is due to Felix Hausdorff. Let X be a set; the elements of X are usually called *points*, though they can be any mathematical object. We allow X to be empty. Let $\mathbf{N}$ be a function assigning to each x (point) in X a non-empty collection $\mathbf{N}(x)$ of subsets of X. The elements of $\mathbf{N}(x)$ will be called *neighbourhoods* of x with respect to $\mathbf{N}$ (or, simply, *neighbourhoods of* x). The function $\mathbf{N}$ is called a neighbourhood topology if the axioms below[1] are satisfied; and then X with $\mathbf{N}$ is called a **topological space**. A topological space in which the *points* are functions is called a function space.

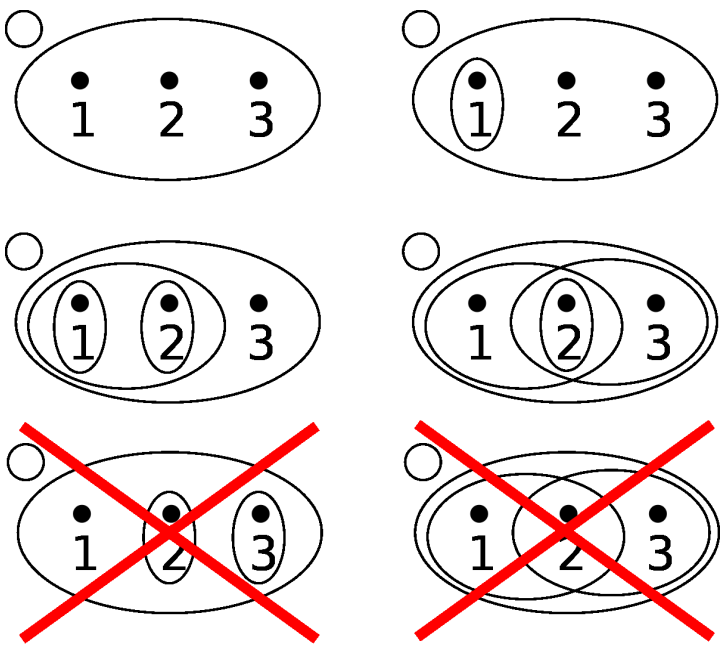
1. If N is a neighbourhood of x (i.e., $N \in \mathbf{N}(x)$), then $x \in N$. In other words, each point belongs to every one of its neighbourhoods.
2. If N is a subset of X and contains a neighbourhood of x, then N is a neighbourhood of x. I.e., every superset of a neighbourhood of a point x in X is again a neighbourhood of x.
3. The intersection of two neighbourhoods of x is a neighbourhood of x.
4. Any neighbourhood N of x contains a neighbourhood M of x such that N is a neighbourhood of each point of M.

The first three axioms for neighbourhoods have a clear meaning. The fourth axiom has a very important use in the structure of the theory, that of linking together the neighbourhoods of different points of X.

A standard example of such a system of neighbourhoods is for the real line $\mathbf{R}$, where a subset N of $\mathbf{R}$ is defined to be a *neighbourhood* of a real number x if there is an open interval containing x and contained in N.

Open sets definition

Given such a structure, we can define a subset U of X to be **open** if U is a neighbourhood of all points in U. It is a remarkable fact that the open sets then satisfy the elegant axioms given below, and that, given these axioms, we can recover the neighbourhoods satisfying the above axioms by defining N to be a neighbourhood of x if N contains an open set U such that $x \in U$.[2]



Four examples and two non-examples of topologies on the three-point set {1,2,3}. The bottom-left example is not a topology because the union of {2} and {3} [i.e. {2,3}] is missing; the bottom-right example is not a topology because the intersection of {1,2} and {2,3} [i.e. {2}], is missing.

A *topological space* is then a set X together with a collection of subsets of X, called **open sets** and satisfying the following axioms:[3]

1. The empty set and X itself are open.
2. Any union of open sets is open.
3. The intersection of any finite number of open sets is open.

The collection τ of open sets is then also called a **topology** on X, or, if more precision is needed, an **open set topology**. The sets in τ are called the open sets, and their complements in X are called closed sets. A subset of X may be neither closed nor open, either closed or open, or both. A set that is both closed and open is called a clopen set.

Examples

1. $X = \{1, 2, 3, 4\}$ and collection $\tau = \{\{\}, \{1, 2, 3, 4\}\}$ of only the two subsets of X required by the axioms form a topology, the trivial topology (indiscrete topology).
2. $X = \{1, 2, 3, 4\}$ and collection $\tau = \{\{\}, \{2\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}, \{1, 2, 3, 4\}\}$ of six subsets of X form another topology.
3. $X = \{1, 2, 3, 4\}$ and collection $\tau = P(X)$ (the power set of X) form a third topology, the discrete topology.
4. $X = \mathbf{Z}$, the set of integers, and collection τ equal to all finite subsets of the integers plus $\mathbf{Z}$ itself is *not* a topology, because (for example) the union of all finite sets not containing zero is infinite but is not all of $\mathbf{Z}$, and so is not in τ .

Closed sets definition

Using de Morgan's laws, the above axioms defining open sets become axioms defining closed sets:

1. The empty set and X are closed.
2. The intersection of any collection of closed sets is also closed.
3. The union of any pair of closed sets is also closed.

Using these axioms, another way to define a topological space is as a set X together with a collection τ of closed subsets of X. Thus the sets in the topology τ are the closed sets, and their complements in X are the open sets.

Other definitions

There are many other equivalent ways to define a topological space: in other words, the concepts of neighbourhood or of open respectively closed set can be reconstructed from other starting points and satisfy the correct axioms.

Another way to define a topological space is by using the Kuratowski closure axioms, which define the closed sets as the fixed points of an operator on the power set of X.

A net is a generalisation of the concept of sequence. A topology is completely determined if for every net in X the set of its accumulation points is specified.

Comparison of topologies

Main article: Comparison of topologies

A variety of topologies can be placed on a set to form a topological space. When every set in a topology τ_1 is also in a topology τ_2 and τ_1 is a subset of τ_2, we say that τ_2 is *finer* than τ_1, and τ_1 is *coarser* than τ_2. A proof that relies only on the existence of certain open sets will also hold for any finer topology, and similarly a proof that relies only on certain sets not being open applies to any coarser topology. The terms *larger* and *smaller* are sometimes used in place of finer and coarser, respectively. The terms *stronger* and *weaker* are also used in the literature, but with little agreement on the meaning, so one should always be sure of an author's convention when reading.

The collection of all topologies on a given fixed set X forms a complete lattice: if $F = \{\tau_\alpha \mid \alpha \text{ in } A\}$ is a collection of topologies on X, then the meet of F is the intersection of F, and the join of F is the meet of the collection of all topologies on X that contain every member of F.

Continuous functions

A function $f : X \to Y$ between topological spaces is called **continuous** if for all $x \in X$ and all neighbourhoods N of $f(x)$ there is a neighbourhood M of x such that $f(M) \subseteq N$. This relates easily to the usual definition in analysis. Equivalently, f is continuous if the inverse image of every open set is open.[4] This is an attempt to capture the intuition that there are no "jumps" or "separations" in the function. A homeomorphism is a bijection that is continuous and whose inverse is also continuous. Two spaces are called *homeomorphic* if there exists a homeomorphism between them. From the standpoint of topology, homeomorphic spaces are essentially identical.

In category theory, **Top**, the category of topological spaces with topological spaces as objects and continuous functions as morphisms is one of the fundamental categories in category theory. The attempt to classify the objects of this category (up to homeomorphism) by invariants has motivated areas of research, such as homotopy theory, homology theory, and K-theory etc.

Examples of topological spaces

A given set may have many different topologies. If a set is given a different topology, it is viewed as a different topological space. Any set can be given the discrete topology in which every subset is open. The only convergent sequences or nets in this topology are those that are eventually constant. Also, any set can be given the trivial topology (also called the indiscrete topology), in which only the empty set and the whole space are open. Every sequence and net in this topology converges to every point of the space. This example shows that in general topological spaces, limits of sequences need not be unique. However, often topological spaces must be Hausdorff spaces where limit points are unique.

There are many ways of defining a topology on **R**, the set of real numbers. The standard topology on **R** is generated by the open intervals. The set of all open intervals forms a base or basis for the topology, meaning that every open set is a union of some collection of sets from the base. In particular, this means that a set is open if there exists an open interval of non zero radius about every point in the set. More generally, the Euclidean spaces $\mathbf{R}^n$ can be given a

topology. In the **usual topology** on $\mathbf{R}^n$ the basic open sets are the open balls. Similarly, **C**, the set of complex numbers, and $\mathbf{C}^n$ have a standard topology in which the basic open sets are open balls.

Every metric space can be given a metric topology, in which the basic open sets are open balls defined by the metric. This is the standard topology on any normed vector space. On a finite-dimensional vector space this topology is the same for all norms.

Many sets of linear operators in functional analysis are endowed with topologies that are defined by specifying when a particular sequence of functions converges to the zero function.

Any local field has a topology native to it, and this can be extended to vector spaces over that field.

Every manifold has a natural topology since it is locally Euclidean. Similarly, every simplex and every simplicial complex inherits a natural topology from $\mathbf{R}^n$.

The Zariski topology is defined algebraically on the spectrum of a ring or an algebraic variety. On $\mathbf{R}^n$ or $\mathbf{C}^n$, the closed sets of the Zariski topology are the solution sets of systems of polynomial equations.

A linear graph has a natural topology that generalises many of the geometric aspects of graphs with vertices and edges.

The Sierpiński space is the simplest non-discrete topological space. It has important relations to the theory of computation and semantics.

There exist numerous topologies on any given finite set. Such spaces are called finite topological spaces. Finite spaces are sometimes used to provide examples or counterexamples to conjectures about topological spaces in general.

Any set can be given the cofinite topology in which the open sets are the empty set and the sets whose complement is finite. This is the smallest T_1 topology on any infinite set.

Any set can be given the cocountable topology, in which a set is defined as open if it is either empty or its complement is countable. When the set is uncountable, this topology serves as a counterexample in many situations.

The real line can also be given the lower limit topology. Here, the basic open sets are the half open intervals $[a, b)$. This topology on **R** is strictly finer than the Euclidean topology defined above; a sequence converges to a point in this topology if and only if it converges from above in the Euclidean topology. This example shows that a set may have many distinct topologies defined on it.

If Γ is an ordinal number, then the set $\Gamma = [0, \Gamma)$ may be endowed with the order topology generated by the intervals (a, b), $[0, b)$ and (a, Γ) where a and b are elements of Γ.

Topological constructions

Every subset of a topological space can be given the subspace topology in which the open sets are the intersections of the open sets of the larger space with the subset. For any indexed family of topological spaces, the product can be given the product topology, which is generated by the inverse images of open sets of the factors under the projection mappings. For example, in finite products, a basis for the product topology consists of all products of open sets. For infinite products, there is the additional requirement that in a basic open set, all but finitely many of its projections are the entire space.

A quotient space is defined as follows: if X is a topological space and Y is a set, and if $f : X \to Y$ is a surjective function, then the quotient topology on Y is the collection of subsets of Y that have open inverse images under f. In other words, the quotient topology is the finest topology on Y for which f is continuous. A common example of a quotient topology is when an equivalence relation is defined on the topological space X. The map f is then the natural projection onto the set of equivalence classes.

The **Vietoris topology** on the set of all non-empty subsets of a topological space X, named for Leopold Vietoris, is generated by the following basis: for every n-tuple $U_1, ..., U_n$ of open sets in X, we construct a basis set consisting of

all subsets of the union of the U_i that have non-empty intersections with each U_i.

Classification of topological spaces

Topological spaces can be broadly classified, up to homeomorphism, by their topological properties. A topological property is a property of spaces that is invariant under homeomorphisms. To prove that two spaces are not homeomorphic it is sufficient to find a topological property not shared by them. Examples of such properties include connectedness, compactness, and various separation axioms.

See the article on *topological properties* for more details and examples.

Topological spaces with algebraic structure

For any algebraic objects we can introduce the discrete topology, under which the algebraic operations are continuous functions. For any such structure that is not finite, we often have a natural topology compatible with the algebraic operations, in the sense that the algebraic operations are still continuous. This leads to concepts such as topological groups, topological vector spaces, topological rings and local fields.

Topological spaces with order structure

- **Spectral**. A space is spectral if and only if it is the prime spectrum of a ring (Hochster theorem).
- **Specialization preorder**. In a space the **specialization** (or **canonical**) **preorder** is defined by $x \leq y$ if and only if $\mathrm{cl}\{x\} \subseteq \mathrm{cl}\{y\}$.

Specializations and generalizations

The following spaces and algebras are either more specialized or more general than the topological spaces discussed above.

- Proximity spaces provide a notion of closeness of two sets.
- Metric spaces embody a metric, a precise notion of distance between points.
- Uniform spaces axiomatize ordering the distance between distinct points.
- Cauchy spaces axiomatize the ability to test whether a net is Cauchy. Cauchy spaces provide a general setting for studying completions.
- Convergence spaces capture some of the features of convergence of filters.
- Grothendieck sites are categories with additional data axiomatizing whether a family of arrows covers an object. Sites are a general setting for defining sheaves.

Notes

[1] Brown 2006, section 2.1.
[2] Brown 2006, section 2.2.
[3] Armstrong 1983, definition 2.1.
[4] Armstrong 1983, theorem 2.6.

References

- Armstrong, M. A. (1983) [1979]. *Basic Topology*. Undergraduate texts in mathematics. Springer. ISBN 0-387-90839-0.
- Bredon, Glen E., *Topology and Geometry* (Graduate Texts in Mathematics), Springer; 1st edition (October 17, 1997). ISBN 0-387-97926-3.
- Bourbaki, Nicolas; *Elements of Mathematics: General Topology*, Addison-Wesley (1966).

- Brown, Ronald, *Topology and groupoids*, Booksurge (2006) ISBN 1-4196-2722-8 (3rd edition of differently titled books) (order from amazon.com).
- Čech, Eduard; *Point Sets*, Academic Press (1969).
- Fulton, William, *Algebraic Topology*, (Graduate Texts in Mathematics), Springer; 1st edition (September 5, 1997). ISBN 0-387-94327-7.
- Lipschutz, Seymour; *Schaum's Outline of General Topology*, McGraw-Hill; 1st edition (June 1, 1968). ISBN 0-07-037988-2.
- Munkres, James; *Topology*, Prentice Hall; 2nd edition (December 28, 1999). ISBN 0-13-181629-2.
- Runde, Volker; *A Taste of Topology (Universitext)*, Springer; 1st edition (July 6, 2005). ISBN 0-387-25790-X.
- Steen, Lynn A. and Seebach, J. Arthur Jr.; *Counterexamples in Topology*, Holt, Rinehart and Winston (1970). ISBN 0-03-079485-4.
- Vaidyanathaswamy, R. (1960). *Set Topology*. Chelsea Publishing Co. ISBN 0486404560.
- Willard, Stephen (2004). *General Topology*. Dover Publications. ISBN 0-486-43479-6.

External links

- Hazewinkel, Michiel, ed. (2001), "Topological space" (http://www.encyclopediaofmath.org/index.php?title=p/t093130), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Topological space (http://planetmath.org/?op=getobj&from=objects&id=380) at PlanetMath.org.

Web of trust

For the internet security website, see WOT Services.

In cryptography, a **web of trust** is a concept used in PGP, GnuPG, and other OpenPGP-compatible systems to establish the authenticity of the binding between a public key and its owner. Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such). As with computer networks, there are many independent webs of trust, and any user (through their identity certificate) can be a part of, and a link between, multiple webs.

The web of trust concept was first put forth by PGP creator Phil Zimmermann in 1992 in the manual for PGP version 2.0:

> As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

Operation of a web of trust

All OpenPGP-compliant implementations include a certificate vetting scheme to assist with this; its operation has been termed a web of trust. OpenPGP identity certificates (which include public key(s) and owner information) can be digitally signed by other users who, by that act, endorse the association of that public key with the person or entity listed in the certificate. This is commonly done at key signing parties.

OpenPGP-compliant implementations also include a vote counting scheme which can be used to determine which public key – owner association a user will trust while using PGP. For instance, if three partially trusted endorsers have vouched for a certificate (and so its included public key – owner binding), OR if one fully trusted endorser has done so, the association between owner and public key in that certificate will be trusted to be correct. The parameters are user-adjustable (e.g., no partials at all, or perhaps 6 partials) and can be completely bypassed if desired.

The scheme is flexible, unlike most public key infrastructure designs, and leaves trust decision(s) in the hands of individual users. It is not perfect and requires both caution and intelligent supervision by users. Essentially all PKI designs are less flexible and require users to follow the trust endorsement of the PKI generated, certificate authority (CA)-signed, certificates. Intelligent action is normally neither required nor allowed. These arrangements are not perfect either, and require both caution and care by users.

In simpler terms, you have 2 keys: a public key that you let the people you trust know; and a private key that only you know. Your private key will decrypt any information encrypted with your public key. In the web of trust you have a key ring with a group of people's public keys.

You encrypt your information with the recipients' public key, and only their private key will decrypt it. You then digitally sign the information with your private key, so when they verify it with your public key, they can confirm that it is you. Doing this will ensure that the information came from you and has not been tampered with, and only the person you are sending it to can read the information (because only they know their private key).

Contrast with typical PKI

In contrast, a typical X.509 PKI permits each certificate to be signed only by a single party: a certificate authority (CA). The CA's certificate may itself be signed by a different CA, all the way up to a 'self-signed' root certificate. Root certificates must be available to those who use a lower level CA certificate and so are typically distributed widely. They are for instance, distributed with such applications as browsers and email clients. In this way SSL/TLS-protected Web pages, email messages, etc. can be authenticated without requiring users to manually install root certificates. Applications commonly include over one hundred root certificates from dozens of PKIs, thus by default bestowing trust throughout the hierarchy of certificates which lead back to them.

Web of trust problems

The OpenPGP web of trust is essentially unaffected by such things as company failures, and has continued to function with little change. However, a related problem does occur. Users, whether individuals or organizations, who lose track of a private key can no longer decrypt messages sent to them produced using the matching public key found in an OpenPGP certificate. Early PGP certificates did not include expiry dates, and those certificates had unlimited lives. Users had to prepare a signed cancellation certificate against the time when the matching private key was lost or compromised. One very prominent cryptographer is still getting messages encrypted using a public key for which he long ago lost track of the private key. He can't do much with those messages except discard them after notifying the sender that they were unreadable and requesting resending with a public key for which he still has the matching private key. Later PGP, and all OpenPGP compliant certificates include expiry dates which automatically preclude such troubles (eventually) when used sensibly. This problem can also be easily avoided by the use of "designated revokers", which were introduced in the early 1990s. A key owner may designate a third party that has permission to revoke the key owner's key (in case the key owner loses his own private key and thus loses the ability

to revoke his own public key).

A non-technical, social difficulty with a Web of Trust like the one built into PGP/OpenPGP type systems is that every web of trust without a central controller (e.g., a CA) depends on other users for trust. Those with new certificates (i.e., produced in the process of generating a new key pair) will not likely be readily trusted by other users' systems, that is by those they have not personally met, until they find enough endorsements for the new certificate. This is because many other Web of Trust users will have their certificate vetting set to require one or more fully trusted endorsers of an otherwise unknown certificate (or perhaps several partial endorsers) before using the public key in that certificate to prepare messages, believe signatures, etc.

Despite the wide use of OpenPGP compliant systems and easy availability of on-line multiple key servers, it is possible in practice to be unable to readily find someone (or several people) to endorse a new certificate (e.g., by comparing physical identification to key owner information and then digitally signing the new certificate). Users in remote areas or undeveloped ones, for instance, may find other users scarce. And, if the other's certificate is also new (and with no or few endorsements from others), then its signature on any new certificate can offer only marginal benefit toward becoming trusted by still other parties' systems and so able to securely exchange messages with them. Key signing parties are a relatively popular mechanism to resolve this problem of finding other users who can install one's certificate in existing webs of trust by endorsing it. Websites also exist to facilitate the location of other OpenPGP users to arrange keysignings. The Gossamer Spider Web of Trust [1] also makes key verification easier by linking OpenPGP users via a hierarchical style web of trust where end users can benefit by coincidental or determined trust of someone who is endorsed as an introducer, or by explicitly trusting GSWoT's top-level key minimally as a level 2 introducer (the top-level key endorses level 1 introducers).

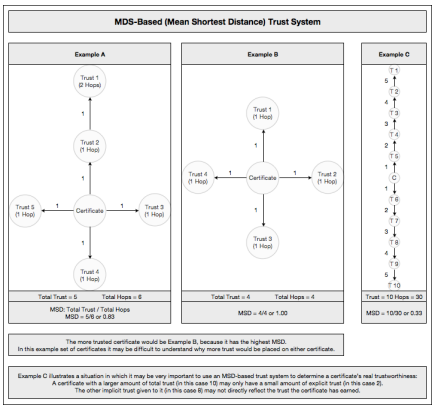
The possibility of finding chains of certificates is often justified by the "small world phenomenon": given two individuals, it is often possible to find a short chain of people between them such that each person in the chain knows the preceding and following links. However, such a chain is not necessarily useful: the person encrypting an email or verifying a signature not only has to find a chain of signatures from his private key to his correspondent's, but also to trust each person of the chain to be honest and competent about signing keys (that is, he has to judge whether these people are likely to honestly follow the guidelines about verifying the identity of people before signing keys). This is a much stronger constraint.

Doing the math

The work of Audun Jøsang is an algebraic treatment of trust measurements and methods and calculation that attempts to capture how trust varies as it "traverses" a web of trust. The other sites (below) contains a link to his publications. A good one to start with might be "An Algebra for Assessing Trust in Certification Chains [2]". Also see Subjective logic and Trust metric.

Mean shortest distance

In statistical analysis of the PGP/GnuPG/OpenPGP Web of trust the **mean shortest distance (MSD)** is one measurement of how "trusted" a given PGP key is within the "strongly connected" set of PGP keys that make up the Web of trust.



MSD-Based Trust Explanation

Drew Streib wrote the following in his explanation of keyring analysis:

> There are a variety of metrics one could apply to this set, but I've chosen initially to measure the "mean shortest distance" (MSD) to each key. Since every key is reachable from every other in the strong set, it is possible to find out the shortest distance (number of hops) to any given key from any other key. Averaging these distances gives the MSD to that key from every other key in the strong set.
>
> It is desirable to have as short as possible an MSD to your key, as that means that on average, people can reach your key quickly through signatures, and thus your key is relatively more trusted than a key with a higher MSD.
>
> NOTE: This does not mean that you should universally trust keys with a low MSD. This is merely a relative measurement for statistical purposes.
>
> The MSD has the property of being no more than 1 higher than your lowest signature. In the worst case, every key in the strong set could reach you by getting to that key, plus 1 hop to get to you. It also encourages the joining of keys that are separated by great distances in the graph, as it will make you a highway of sorts for shortest paths between keys in those groups. In the end, it encourages an overall tightening of the world graph, shortening distances between key owners.

MSD has become a common metric for analysis of sets of PGP keys. Very often you will see the MSD being calculated for a given subset of keys and compared with the *global MSD* which generally refers to the keys ranking within one of the larger key analyses of the global Web of trust.

References

[1] http://www.gswot.org
[2] http://www.unik.no/people/josang/papers/Jos1999-NDSS.pdf

Further reading

Ferguson, Niels; Bruce Schneier (2003). *Practical Cryptography*. John Wiley & Sons. ISBN 0-471-22357-3.

External links

- An explanation of the PGP Web of Trust (http://www.rubin.ch/pgp/weboftrust.en.html)
- analysis of the *strong set* in the PGP web of trust (http://pgp.cs.uu.nl/plot/) - regularly updated

X.509

In cryptography, **X.509** is an ITU-T standard for a public key infrastructure (PKI) and Privilege Management Infrastructure (PMI). X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.

History and usage

X.509 was initially issued on July 3, 1988 and was begun in association with the X.500 standard. It assumes a strict hierarchical system of certificate authorities (CAs) for issuing the certificates. This contrasts with web of trust models, like PGP, where anyone (not just special CAs) may sign and thus attest to the validity of others' key certificates. Version 3 of X.509 includes the flexibility to support other topologies like bridges and meshes.[1] It can be used in a peer-to-peer, OpenPGP-like web of trustWikipedia:Citation needed, but was rarely used that way as of 2004. The X.500 system has only ever been implemented by sovereign nations for state identity information sharing treaty fulfillment purposes, and the IETF's Public-Key Infrastructure (X.509), or **PKIX**, working group has adapted the standard to the more flexible organization of the Internet. In fact, the term *X.509 certificate* usually refers to the IETF's PKIX Certificate and CRL Profile of the X.509 v3 certificate standard, as specified in RFC 5280, commonly referred to as PKIX for *Public Key Infrastructure (X.509)*.Wikipedia:Citation needed

Certificates

In the X.509 system, a certification authority issues a certificate binding a public key to a particular distinguished name in the X.500 tradition, or to an *alternative name* such as an e-mail address or a DNS entry.Wikipedia:Citation needed

An organization's trusted root certificates can be distributed to all employees so that they can use the company PKI system. Browsers such as Internet Explorer, Firefox, Opera, Safari and Chrome come with a predetermined set of root certificates pre-installed, so SSL certificates from larger vendors will work instantly; in effect the browsers' developers determine which CAs are trusted third parties for the browsers' users.Wikipedia:Citation needed

X.509 also includes standards for certificate revocation list (CRL) implementations, an often neglected aspect of PKI systems. The IETF-approved way of checking a certificate's validity is the Online Certificate Status Protocol (OCSP). Firefox 3 enables OCSP checking by default along with versions of Windows including Vista and later.Wikipedia:Citation needed

Structure of a certificate

The structure foreseen by the standards is expressed in a formal language, namely Abstract Syntax Notation One.

The structure of an X.509 v3 digital certificate is as follows:

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

Each extension has its own id, expressed as Object identifier, which is a set of values, together with either a critical or non-critical indication. A certificate-using system MUST reject the certificate if it encounters a critical extension that it does not recognize, or a critical extension that contains information that it cannot process. A non-critical extension MAY be ignored if it is not recognized, but MUST be processed if it is recognized.[2]

The structure of Version 1 is given in RFC 1422.[3]

ITU-T introduced issuer and subject unique identifiers in version 2 to permit the reuse of issuer or subject name after some time. An example of reuse will be when a CA goes bankrupt and its name is deleted from the country's public list. After some time another CA with the same name may register itself, even though it is unrelated to the first one. However, IETF recommends that no issuer and subject names be reused. Therefore, version 2 is not widely deployed in the Internet.Wikipedia:Citation needed

Extensions were introduced in version 3. A CA can use extensions to issue a certificate only for a specific purpose (e.g. only for signing digital object). Each extension can be critical or non-critical. If an extension is critical and the system processing the certificate does not recognize the extension or cannot process it, the system MUST reject the entire certificate. A non-critical extension, on the other hand, can be ignored while the system processes the rest of the certificate.Wikipedia:Citation needed

In all versions, the serial number MUST be unique for each certificate issued by a specific CA (as mentioned in RFC 2459).

Extensions informing a specific usage of a certificate

RFC 5280 (and its predecessors) defines a number of certificate extensions which indicate how the certificate should be used. Most of them are arcs from the `joint-iso-ccitt(2) ds(5) id-ce(29)` OID. Some of the most common, defined in section 4.2.1, are:

- Basic Constraints, `{ id-ce 19 }`, are used to indicate whether the certificate belongs to a CA.
- Key Usage, `{ id-ce 15 }`, provides a bitmap specifying the cryptographic operations which may be performed using the public key contained in the certificate; for example, it could indicate that the key should be used for signatures but not for encipherment.
- Extended Key Usage, `{ id-ce 37 }`, is used, typically on a leaf certificate, to indicate the purpose of the public key contained in the certificate. It contains a list of OIDs, each of which indicates an allowed use. For example, `{ id-pkix 3 1 }` indicates that the key may be used on the server end of a TLS or SSL connection; `{ id-pkix 3 4 }` indicates that the key may be used to secure email.

In general, if a certificate has several extensions restricting its use, all restrictions must be satisfied for a given use to be appropriate. RFC 5280 gives the specific example of a certificate containing both keyUsage and extendedKeyUsage: in this case, both must be processed and the certificate can only be used if both extensions are coherent in specifying the usage of a certificate. For example, NSS uses both extensions to specify certificate usage.[4]

Certificate filename extensions

Common filename extensions for X.509 certificates are:Wikipedia:Citation needed

- `.pem` – (Privacy-enhanced Electronic Mail) Base64 encoded DER certificate, enclosed between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----"
- `.cer`, `.crt`, `.der` – usually in binary DER form, but Base64-encoded certificates are common too (see `.pem` above)
- `.p7b`, `.p7c` – PKCS#7 SignedData structure without data, just certificate(s) or CRL(s)
- `.p12` – PKCS#12, may contain certificate(s) (public) and private keys (password protected)
- `.pfx` – PFX, predecessor of PKCS#12 (usually contains data in PKCS#12 format, e.g., with PFX files generated in IIS)

PKCS#7 is a standard for signing or encrypting (officially called "enveloping") data. Since the certificate is needed to verify signed data, it is possible to include them in the SignedData structure. A `.P7C` file is a degenerated SignedData structure, without any data to sign.Wikipedia:Citation needed

PKCS#12 evolved from the *personal information exchange* (PFX) standard and is used to exchange public and private objects in a single file.Wikipedia:Citation needed

Sample X.509 certificates

This is an example of a decoded X.509 certificate for `www.freesoft.org`, generated with OpenSSL—the actual certificate is about 1 kB in size. It was issued by Thawte (since acquired by VeriSign and now owned by Symantec), as stated in the Issuer field. Its subject contains many personal details, but the most important part is usually the common name (CN), as this is the part that must match the host being authenticated. Also included is an RSA public key (modulus and public exponent), followed by the signature, computed by taking a MD5 hash of the first part of the certificate and signing it (applying the encryption operation) using Thawte's RSA private key.

```
$ openssl x509 -in freesoft-certificate.pem -noout -text
Certificate:
   Data:
       Version: 1 (0x0)
```

```
        Serial Number: 7829 (0x1e95)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/emailAddress=server-certs@thawte.com
        Validity
            Not Before: Jul  9 16:04:02 1998 GMT
            Not After : Jul  9 16:04:02 1999 GMT
        Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                 OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                    33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                    66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                    70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                    16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                    c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                    8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                    d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                    e8:35:1c:9e:27:52:7e:41:8f
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
        92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
        ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
        d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
        0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
        5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
        8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
        68:9f
```

To validate this certificate, one needs a second certificate that matches the Issuer (Thawte Server CA) of the first certificate. First, one verifies that the second certificate is of a CA kind; that is, that it can be used to issue other certificates. This is done by inspecting a value of the *CA* attribute in the *X509v3 extension* section. Then the RSA public key from the CA certificate is used to decode the signature on the first certificate to obtain a MD5 hash, which must match an actual MD5 hash computed over the rest of the certificate. An example CA certificate follows:

```
$ openssl x509 -in thawte-ca-certificate.pem -noout -text
Certificate:
   Data:
       Version: 3 (0x2)
       Serial Number: 1 (0x1)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
```

```
            CN=Thawte Server CA/emailAddress=server-certs@thawte.com
    Validity
        Not Before: Aug  1 00:00:00 1996 GMT
        Not After : Dec 31 23:59:59 2020 GMT
    Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
             OU=Certification Services Division,
             CN=Thawte Server CA/emailAddress=server-certs@thawte.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
                00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
                85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
                6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
                6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
                29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
                6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
                5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
                3a:c2:b5:66:22:12:d6:87:0d
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:TRUE
Signature Algorithm: md5WithRSAEncryption
    07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
    a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
    3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
    4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
    8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
    e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
    b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
    70:47
```

This is an example of a self-signed certificate, as the issuer and subject are the same. There's no way to verify this certificate except by checking it against itself; instead, these top-level certificates are manually stored by web browsers. Thawte is one of the root certificate authorities recognized by both Microsoft and Netscape. This certificate comes with the web browser and is trusted by default. As a long-lived, globally trusted certificate that can sign anything (as there are no constraints in the *X509v3 Basic Constraints* section), its matching private key has to be closely guarded.

Security

There are a number of publications about PKI problems by Bruce Schneier, Peter Gutmann and other security experts.

Certificate complexity

A majority of Internet users, either business or social, currently lack the basic ability, knowledge and willingness to effectively use cryptographic applications in a way that can successfully deter imminent threats. The complexity of this task is one of the weaknesses of public key cryptography. A lack of user friendliness and overall usability thus affects solution efficiency.[5] To deal with such issues, major software companies have included a bundle of root certificates, which have been audited for security purposes, into user browsers and operating systems. For the sake of user friendliness and interoperability, all web browsers and operating systems currently contain this audited *Trusted Root Store* of certificate issuing authorities. Certificates issued by these organizations, or their subordinate authorities, are transparently trusted by relying entities. These certificates are automatically deemed as secure and trustworthy, as opposed to those issued by “unknown” issuers, which a relying party is warned not to trust. This interprets into certificates published by all authorities that have not been included in the root store. This approach attempts to make the provision of system security automatic and transparent, and essentially removes from the end user the decision making process about the trustworthiness of web entities.

The X.509 standard was primarily designed to support the X.500 structure, but today's use cases center around the web. Many features are of little or no relevance today. The X.509 specification suffersWikipedia:Please clarify from being over-functional and underspecified and the normative information is spread across many documents from different standardization bodies. Several profiles were developed to solve this, but these introduce interoperability issues and did not fix the problem.

Architectural weaknesses

- Use of blacklisting invalid certificates (using CRLs and OCSP) instead of whitelisting,
- CRLs are notably a poor choice because of large sizes and convoluted distribution patterns,
- Ambiguous OCSP semantics and lack of historical revocation status,
- Revocation of root certificates is not addressed,
- Aggregation problem: Identity claims (authenticate with an identifier), attribute claims (submit a bag of vetted attributes), and policy claims are combined in a single container. This raises privacy, policy mapping, and maintenance issues,
- Delegation problem: CAs cannot technically restrict subordinate CAs from issuing certificates outside a limited namespaces or attribute set; this feature of X.509 is not in use. Therefore a large number of CAs exist on the Internet, and classifying them and their policies is an insurmountable task. Delegation of authority within an organization cannot be handled at all, as in common business practice.
- Federation problem: Certificate chains that are the result of subordinate CAs, bridge CAs, and cross-signing make validation complex and expensive in terms of processing time. Path validation semantics may be ambiguous. The hierarchy with a third-party trusted party is the only model. This is inconvenient when a bilateral trust relationship is already in place.

Problems with certificate authorities

- The subject, not the relying party, purchases certificates. The subject will often utilize the cheapest issuer, so quality is not being paid for in the competing market. This is partly addressed by Extended Validation certificates.
- Certification authorities deny almost all warranties to the user (including subject or even relying parties).
- The expiration date should be used to limit the time the key strength is deemed sufficient. This parameter is abused by certification authorities to charge the client an extension fee. This places an unnecessary burden on the user with key roll-over.
- "Users use an undefined certification request protocol to obtain a certificate which is published in an unclear location in a nonexistent directory with no real means to revoke it."
- Like all businesses, CAs are subject to the legal jurisdiction(s) of their site(s) of operation, and may be legally compelled to compromise the interests of their customers and their users. Intelligence agencies have also made use of false certificates issued through extralegal compromise of CAs, such as DigiNotar, to carry out Man-In-The-Middle attacks.Wikipedia:Citation needed

Implementation issues

Implementations suffer from design flaws, bugs, different interpretations of standards and lack of interoperability of different standards. Some problems are:Wikipedia:Citation needed

- Many implementations turn off revocation check:
  - Seen as obstacle, policies are not enforced
  - If it was turned on in all browsers by default, including code signing, it would probably crash the infrastructure.
- DNs are complex and little understood (lack of canonicalization, internationalization problems, ..)
- rfc822Name has two notations
- Name and policy constraints hardly supported
- Key usage ignored, first certificate in a list being used
- Enforcement of custom OIDs is difficult
- Attributes should not be made critical because it makes clients crash.
- Unspecified length of attributes lead to product-specific limits

Exploits

- MD2-based certificates were used for a long time and were vulnerable to preimage attacks. Since the root certificate already had a self-signature, attackers could use this signature and use it for an intermediate certificate.
- In 2005, Arjen Lenstra and Benne de Weger demonstrated "how to use hash collisions to construct two X.509 certificates that contain identical signatures and that differ only in the public keys", achieved using a collision attack on the MD5 hash function.
- In 2008, Alexander Sotirov and Marc Stevens presented at the Chaos Communication Congress a practical attack that allowed them to create a rogue Certificate Authority, accepted by all common browsers, by exploiting the fact that RapidSSL was still issuing X.509 certificates based on MD5.
- X.509 certificates based on SHA-1 had been deemed to be secure up until very recent times. In April 2009 at the Eurocrypt Conference,[6] Australian Researchers of Macquarie University presented "Automatic Differential Path Searching for SHA-1".[7] The researchers were able to deduce a method which increases the likelihood of a collision by several orders of magnitude.[8]
- EV-certificates are of very limited help, because Browsers do not have policies that disallow EV-certificates.[9]
- There are implementation errors with X.509 that allow e.g. falsified subject names using null-terminated strings[10] or code injections attacks in certificates.

- By using illegal[11] 0x80 padded subidentifiers of Object Identifiers, wrong implementations or by using integer-overflows of the client's browsers, an attacker can include an unknown attribute in the CSR, which the CA will sign, which the client wrongly interprets as "CN" (OID=2.5.4.3). Dan Kaminsky at the 26th Chaos Communication Congress "Black OPs of PKI"

PKI standards for X.509

- PKCS7 (Cryptographic Message Syntax Standard - public keys with proof of identity for signed and/or encrypted message for PKI)Wikipedia:Citation needed
- Secure Sockets Layer (SSL) - cryptographic protocols for Internet secure communicationsWikipedia:Citation needed
- Online Certificate Status Protocol (OCSP) / Certificate Revocation List (CRL) - this is for validating proof of identityWikipedia:Citation needed
- PKCS12 (Personal Information Exchange Syntax Standard) - used to store a private key with the appropriate public key certificateWikipedia:Citation needed

Certification authority

Main article: Certificate authority

A certification authority (CA) is an entity which issues digital certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many public key infrastructure (PKI) schemes.Wikipedia:Citation needed

There are many commercial CAs that charge for their services. Institutions and governments may have their own CAs, and there are free CAs.Wikipedia:Citation needed

Public-Key Infrastructure (X.509) Working Group

The The Public-Key Infrastructure (X.509) working group (PKIX) was a working group of the Internet Engineering Task Force dedicated to creating RFCs and other standard documentation on issues related to public key infrastructure based on X.509 certificates. PKIX was established in Autumn 1995 in conjunction with the National Institute of Standards and Technology. The Working Group was closed in November 2013.

Protocols and standards supporting X.509 certificates

- TLS/SSL
- S/MIME (Secure Multipurpose Internet Mail Extensions)
- IPsec
- SSH
- Smart card
- HTTPS
- EAP
- LDAP
- Trusted Computing Group (TNC TPM NGSCB)
- CableLabs (North American Cable Industry Technology Forum)
- WS-Security
- XMPP
- Microsoft Authenticode
- OPC UA

References

[1] RFC 4158
[2] RFC 5280 section 4.2, retrieved 12 February 2013 (http://tools.ietf.org/html/rfc5280#section-4.2,)
[3] RFC 1422 (http://www.ietf.org/rfc/rfc1422)
[4] All About Certificate Extensions (https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/nss_tech_notes/nss_tech_note3)
[5] Zissis, D. & Lekkas, D., "Trust coercion in the name of usable Public Key Infrastructure", Security And Communication Networks, John Wiley & Sons, (2013)
[6] Eurocrypt Conference (http://www.iacr.org/conferences/eurocrypt2009/)
[7] "Automatic Differential Path Searching for SHA-1" (http://eurocrypt2009rump.cr.yp.to/837a0a8086fa6ca714249409ddfae43d.pdf)
[8] SHA-1 Collision Attacks Now 2^{52} (http://www.secureworks.com/research/blog/index.php/2009/6/3/sha-1-collision-attacks-now-252/)
[9] Zusman and Sotirov Blackhat 2009 (http://www.blackhat.com/presentations/bh-usa-09/SOTIROV/BHUSA09-Sotirov-AttackExtSSL-PAPER.pdf)
[10] Marlinspike Blackhat 2009 (http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf)
[11] Rec. ITU-T X.690, clause 8.19.2

Additional reading

- ITU-T Recommendation X.509 (http://www.itu.int/rec/T-REC-X.509) (2005): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 08/05.
- C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure: Certificate Management Protocols", RFC 2510, March 1999
- Housley, R., W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 3280, April 2002. Obsoleted by RFC 5280, Obsoletes RFC 2459/ updated by RFC 4325, RFC 4630.
- Housley, R., W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 2459, January 1999. Obsoleted by RFC 3280.
- Arjen Lenstra, Xiaoyun Wang and Benne de Weger, On the possibility of constructing meaningful hash collisions for public keys, full version, with an appendix on colliding X.509 certificates, 2005 (http://www.win.tue.nl/~bdeweger/CollidingCertificates/) (see also (http://eprint.iacr.org/2005/067)).

External links

- ITU-T Recommendation X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks (http://www.itu.int/rec/T-REC-X.509/en)
- Peter Gutmann's articles, an overview of PKI (http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf), X.509 implementation notes X.509 Style Guide (http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt)
- PKIX website (http://www.ietf.org/html.charters/pkix-charter.html)
- Enterprise Trust Integration and Web Services Security standards and demos (http://www.trustedwebservices.org)
- FAQ from RSA Labs (http://www.rsasecurity.com/rsalabs/node.asp?id=2155)
- Sun Inc. - Secure code guidelines (http://java.sun.com/security/seccodeguide.html)
- RFC 4158 - Internet X.509 Public Key Infrastructure: Certification Path Building
- CSR Decoder and Certificate Decoder (http://certlogik.com/decoder) - can be used to decode and examine an encoded CSR or certificate.
- SSL Checker (http://certlogik.com/sslchecker/) - can be used to test a certificate and that it has been installed correctly
- phpseclib: X.509 Decoder (http://phpseclib.sourceforge.net/x509/decoder.php) - decodes to an associative array whose keys correspond to X.509's ASN.1 description

- Certificate Lookup (http://www.ssltools.com/certificate_lookup) - ssl tool that can be used to verify and troubleshoot X.509 certificate installations or examine their contents
- SSLTools Manager for Windows (http://www.ssltools.com/manager) - Windows utility to create X.509 certificates for IIS or Exchange.
- Web site with a free, graphical X509 Builder (http://sectools.free.fr/EN/) tool for Windows.

Article Sources and Contributors

Bitcoin *Source*: https://en.wikipedia.org/w/index.php?oldid=614924761 *Contributors*: 020543m, 0Core0, 0x0F, 10pippe, 123sage321, 12uihy, 13Goldem, 4th-otaku, 51coin, 7partparadigm, 84user, 99of9, A. di M., APL, ASOTMKX, Aalegado, Abckhxkcvrtyu, Aceat64, Acmilan15, Activist, AdjustablePliers, Advorak, Aesir.le, AgadaUrbanit, Agyle, Ainderbyquernhow, Ajfweb, Akh81, Alarics, Ale85, Alex.ryazantsev, Alfy32, Ali78v, AliveFreeHappy, Alphazeta33, Andonic, Andrewman327, Andyhowlett, Anna Roy, AnonMoos, Anonymous68th, Another John S, Anotherusername1, Antiquax, Antonio92, Aoidh, Aoxfordca, Araignee, Are you ready for IPv6?, Arichnad, Arjayay, Arpabone, Art LaPella, Arthur Rubin, Astrohacker, Asukite, Athaba, Atheros1, Atomicthumbs, Aussiejohn, AustinZ, BD2412, Back ache, BackDraft9387, BalanceUT, Barek, BarnstarFactory, Barry McGuiness, Batouzo, Bazuz, Beach drifter, BeachComber1972, Beachy, Ben Ben, BenM, Bender235, Benlisquare, BerkeleyLaw1979, BernardZ, Bgwhite, Biblbroks, Bidofthis, Billymac00, BitBus, Bitbain, BitcoinTomWilliams, BitcoinWiki, Bitcoineer, BitcoinrealityCheck, Bitenthusiast, Bitslots, Bizso, Black Falcon, Black Rainbow 999, Blue Matt, Blue oracl2495, Bluefist, Blueyed, Bob Re-born, Bobby223322, Bobrayner, Bodnotbod, Bojo1498, Bonadea, Bosmon, Bposert, Brianhe, Bulwersator, Burpen, BurritoBazooka, Bussings, Byelf2007, Byset, Byung do jung, C S, C.Fred, CJDuhaime, CLSwiki, Cagliost, Cahk, Calabe1992, Callanecc, Calmer Waters, Cambrasa, CameraWallet, Canton, Casascius, Cathy Richards, Cavirtex, Celosia, Cesiumfrog, Cgtdk, Checkingfax, ChocTinFoil, Chosenken, ChrNPal, Chris the speller, ChrisGualtieri, ChrisPDX, Chrisarnesen, Chronosilence, Chronulator, Chuckwolber, Chunchi8, Ciphergoth, Citation Needed, Civic Cat, Cleanelephant, Clearish, Cleduc, Cliff12345, Cloudswrest, Cmskog, Codename Lisa, CogitoErgoSum14, Cojovo, ColdWind, ComfyKem, Conchisness, ContinueWithCaution, Cooperh, Coubs514, Courcelles, Craig Mayhew, Craigrottman, Crazyanimal, Crazycomputers, Cretog8, Cricobr, Criticcon, Crookesmoor, CrunchySkies, Currency cobano, Cuzkatzimhut, Cyborg4, Cyde, Cyferx, D-Notice, DMNT, DPRoberts534, Da5id403, Dabomb87, Dairhead, Damian Yerrick, Dandv, Daniel Brockman, Danmaz74, Danny Sprinkle, Danrok, Danuthaiduc, DataSurfer, DataWraith, DaveInAustin, Davemcarlson, David Gerard, David Hedlund, David Levy, David Woodward, DavidHarkness, DavidWBrooks, Davidhorman, Daytona2, Dbabbitt, Dbrisinda, Dchestnykh, Dcoetzee, Ddas, Ddcorkum, Delldot, DemocraticLuntz, Denispir, Der Golem, Dereckson, Derek R Bullamore, Destynova, Dewritech, Diablo-D3, Dickdock, Dismas, DissidentAggressor, Dl2000, Dmarquard, Dmh, Dmoores55, Dmurashchik, Do3cc, Doc Merlin, Dom Kaos, DouglasCalvert, Dpacmittal, Dr.K., DrCroco, Dratman, Drbreznjev, Dree12, Drernie, Dreth, Drmies, Dscotese, Dthomsen8, Dusanson, Dvdlevy120, Dvtimes99, Dzhim, E123, Ecureuil espagnol, Ecurrencies, Ed Brey, EdJohnston, Edoe, Edward, Ehrenkater, Ekki01, Ekspiulo, Elassint, Elatanatari, Electric Wombat, Electricmuffin11, Eliteness, Eliteware, Ellomate, Elockid, Eloquence, Elucches, Emeitner, Emj, Encrypto1, Epicgenius, Equilibrium007, EricLarson80, Ericsheldon, Esrogs, Ethridgela, EverGreg, Excirial, Exercisephys, Exposo, Extramaster, EyeKnows, Ezzayakoo, FML, FOXIBOX, Factsearch, Falkirks, FatalError, Favonian, Fblan001, Feezo, Fences and windows, Firenu, FisherQueen, Fiskbullar, Flat Out, Fleetham, Flyer22, FormerNukeSubmariner, Fortheloveofbacon, Fraggle81, FrankAndProust, Frap, Fred Bauder, Frettsy, Frietjes, Froid, FuFoFuEd, Fudoreaper, Furrykef, G S Palmer, GKFX, GSK, Gagarine, Galzigler, Gamaliel, Gary, Gatenosix, Gatopeich, Gautier lebon, GenQuest, Genjix, George strawberry, GeraldKaszuba, Geraldshields11, Gerritharkness, Gert7, Gidoca, Gil Dawson, Gilderien, Gilliam, Ginot, Giraffedata, Giszmo, Glavkos, Glenn, GliderMaven, Global Standard, Gmaxwell, Gnurkel, GoCubs88, Gogophergo, GoingBatty, Gowthamkare, Gracefool, Graingert, Greggydude, GregorB, Grondilu, Gronky, Gsarwa, Gts 2000, Guest2625, Guss82, Gwern, HLachman, HangingCurve, Hannasnow, Harksaw, Hats2543, Headinthedoor, HeinzzzderMannn, Helvetius, Henriwatson, Henryb2000, Hertz1888, Heyandy889, Hgilbert, Hisabness, Hobsonlane, Hontogaichiban, Hostfat, Hotelmason241, HowardStrong, Hto9950, Huangpo, Hulagutten, Hydrargyrum, Hyfen, I JethroBT, IGEL, Ianpurton, IceCreamEmpress, Ileresolu, Imperi, Ingolfson, Int21h, Interligator, Intgr, Intrepid-NY, Iolaka91823972, Ipsign, IronGargoyle, Isaac Rabinovitch, Iselilja, Isidore, Ivan.a.tikhonov, Ixfd64, Izmailov, Izyt, J.vayner, JLMadrigal, JUBALCAIN, JaGa, Jackzhp, JaconaFrere, JacquelineDerrida, Jadair10, Jake Wartenberg, Jakebed, Jakub Vrána, James Fryer, JamesBWatson, JanetteDoe, Jarble, JasonAQuest, JasonCooney, Jasper Deng, Jaumenuez, Jav wiki, JayBeeCool, Jbening, Jclemens, Jcmorin, Jcnetsys, Jdhowlett, JeanLucMargot, Jedonnelley, Jehorn, Jerebin, Jerome Charles Potts, JesseT77, Jesus Presley, JexsterB312, Jfmantis, Jgarzik, Jianhui67, Jibbsisme, Jim1138, JimNelin, Jimisdead, Jimmilu, Jimthing, Jmchugh, Joel7687, John, John Cline, John Holmes II, John Vandenberg, JohnCD, Johndrinkwater, Johnnyshocker, Jokes Free4Me, Jonathanriley, Jonpatterns, Josh Keen, JoshDieter, Josve05a, Jrn0074, Jswd, Jtjathomps, Juggernauts10, Julesd, Julia Eremina, Juneblender, Justintbassett, Kai445, Kaldari, Kamillas 1, Kap 7, Kaputa12, KarlFrei, Karlhendrikse, Kaskaad, Kbog, Keepinternetfree, Keithbob, KeithyIrwin, Kencf0618, Kenfyre, Kephir, Kevin143, Keyur mithawala, Kgrad, Khazar2, Kiko4564, Kinema, Kingturtle, Kirov Airship, Kitsios.a, Kitsunegami, Kittybrewster, Kjj31337, Kjrreid, Kmag, Kmcnamee, Koavf, Korrawit, Kosik, Kozuch, KrakatoaKatie, Krassotkin, Krautsk, Krisives, Kwamikagami, KyleLandas, Kylebk, L0ngpar1sh, L3lackEyedAngels, LaFayettePolitico, Ladislav Mecir, Lambda, Lambiam, LampWithALeaf, Larklight, Larmsterpoet, LarryLACa, Larrymcp, Laser brain, Lawsonstu, Laytonsmith14, Leandro.cesar, LeeParq, Leijurv, Lemonsdrops, Lenin and McCarthy, LeoFrank, Leonarbe, Leopard850, Leptus Froggi, Lesath, Leuko, Libertaar, Liftarn, LilHelpa, Lironah, Litecoinguy, Lmatt, LogX, Logictheo, Lone boatman, Loopback007, Lordgilman, LoseKabel, Louisstar, Lovibond, Luke-Jr, M.qrius, MARIODOESBREAKFAST, MER-C, MJustice, MS10EL, MSGJ, MSUGRA, Maccollie, Mack2, Maged123, Magioladitis, Mahir256, Majorbolz, Makomk, Makyen, Manderso, Mann jess, Manscher, Maproom, Marc Bago, Marcika, Marco Guzman, Jr, Mariagvozd, Mark Arsten, MarkMLl, Markewilliams, Martian, Mascarponette, Masharabinovich, Mate2code, Materialscientist, Matt06012011, MattGiuca, Matthieu Houriet, Mattl, Mattmorgan, Mattpalen, Mauls, Maurice Carbonaro, Mayosmith, Mbmexpress, Mchcopl, Mckinley99, Mdwh, Meco, Melksoft, Meni Rosenfeld, Merlinthe, Message From Xenu, Metallurgist, Mfwitten, Mgumn, Michael Hardy, Micru, Mike Hearn, Mike Linksvayer, Mike hayes, Mike the k, Mike30188, Mikesc86, Mikeyfaces, Mikhail Ryazanov, Milidepe, Miracle Pen, Miraje182, Misbach, MisterHungry, Mizerydearia, Mjbauer, Mleeds12, Mmeijeri, Modify, MohammedBinAbdullah, Mojargon, Mollyjohn1436, Momposi, Mononomic, MonteDaCunca, Mootros, Morgan Leigh, Mortense, Mr. Stradivarius, Mr. Strong Bad, Mr.choppers, MrAndreessen, MrBill3, MrScorch6200, MrTree, Mrcatzilla, Mrienstra, Mrmatiko, Mrschimpf, Mu Mind, Mutorq, Mwarren us, Myfare, N2e, Nagle, Ndeine, Nealmcb, Neilonidas, Neko-chan, Nelson, Neoconfederate, Netrapt, New worl, Nicboman, Niceguyedc, Nicolas1981, Nikkimaria, Nikos 1993, Ninly, NintendoFan, NittyG, No One of Consequence, Noformation, Nonnompow, Nonstopdrivel, Norm mit, NorthBySouthBaranof, NorthernKnightNo1, Nurg, Nuujinn, Nyashinski, Nyttend, Object404, Observer6, Ocaasi, Oddbodz, Officiallyover, OhanaUnited, OldCar, Oneiros, Oniscoid, Ontarioboy, OpenToppedBus, Opticbit, Orbixx, Oren0, OriumX, OsmanRF34, Osyed1, Ott2, OverQuantum, OverlordQ, Ozzke, P The D, Palmcluster, Palosirkka, PamD, Parjlarsson, ParkKimLim, Parsiferon, Patrick2409, PatrikR, Paul 012, Paulsnx2, Pdcook, Penbat, PeterEasthope, Peterowenwilson, Petrb, Petri Krohn, Phelix77, Philip Trueman, Philippe (WMF), Phlegat, Piedmont, Piet Delport, Pifactorial, Pifvyubjwm, Pigpie45, Pinethicket, Pinetreecrush, PinkAmpersand, PirdPirdPrid, Pjedicke, Pjf, Pjrm, Plaiche, Play Money for Dummies, Pmsyyz, Pol098, Polarandwet, Polargeo, Polargeo 3, Ponyo, Portillo, Pratyeka, Proudnewly1, Pyronite, Qrsdogg, QuantOfAsia, Quarl, QuentinUK, Quinn d, Quinxorin, Qwertym77, Qwertyus, R. fiend, R2D2!, RA0808, RCraig09, Randomblue, Raphaelbastide, Rathergood15, Rayman60, Rebroad, Recury, Red Slash, Redcert, Redrose64, Revent, Reywas92, Rezonansowy, Rfc1394, Rholme, Rhydic, Rich Farmbrough, Richardbondi, Rick Raubenheimer, Riley Huntley, Ripper234, Risoto2000, Riventree, Rivoton, Rjm at sleepers, Robert Horning, Roberticus, Robofish, Rodarmor, Rollins83, Rook944, Rooley555, Ross Fraser, RoyGoldsmith, Roybadami, Rsraleigh, Ruby Murray, Rubyuser, Rudd-O, Runtime, Rutebega, Ryan Vesey, Rōnin, S2000magician, SBaker43, SF007, SHCarter, SPACKlick, SPECIFICO, SYSS Mouse, Sainibindass, Samw, Samwalton9, Sandro kensan, Sanpitch, Sarath divakar, Satellizer, Sbjf, Schandi, Sckirklan, Scott Illini, Seancasey00, Sebleouf, Ses4j, Shadeofblue, Shadowzdx, Shadypalm88, Sharpseek, Shawnc, Shelbymoore3, Sheridan Zhoy, Shiftchange, Shizhao, ShoWPiece, Silbtsc, SimonTrew, Sipa1024, Sirius-n, Sithishade, Skakkle, Skmacksler, Skomorokh, Slushcz, Smallbones, SmartGuy Old, Smartse, Smite-Meister, Smyth, Snacks, Snarpel, Sneak, Socialmaven1, Socrates2008, Sokolesq, SolarStarSpire, Soler97, Solphusion, Someguy1221, Someone not using his real name, Sonba, Sonicsuns, Soundcomm, Spangineer, Speedplane, Sperxios, SpiltOctacle, Spobin, Springnuts, Sr.ganador, Srich32977, Stanjourdan, Startswithj, Statecraft, Stephenb, Steve Lux, Jr., Steve.alleny, Steve03Mills, Stfg, Strait, Suffusion of Yellow, SummerWithMorons, SuperHamster, Superdavywavy, Surfer43, Susurrus, Suumitgeek, SvGeloven, Svick, Swctg, Swenkman, Swikid, Swliv, SystemBuilder, T-Boy, TCMemoire, TJRC, Tabletop, Tabletrack53, TakuyaMurata, Tamfang, Tango303, Targaryen, Tarinth, Tbuckley89, Tcatm, Teaksmitty, Technical 13, Teggles, Teilolondon, Tekhnofiend, Tekn04, Tere, Terrycojones, Tesssla, Testem, Tgeller, The Anome, The Frenchie, The Interior, The editor1900, TheCuriousGnome, TheGoodBadWorst, TheLastNinja, Theamer, Thegreatgrabber, Theodore Kloba, Thepm, ThereNHere, ThisIsNotReal, Thkie, Thomas.W, Thomasee73, Thorwald, Throwaway85, Thumperward, ThunderPeel2001, Tigerchen, TimBray, Timotheus Canens, TippyGoomba, TiriPon, TitaniumCarbide, Titfditroyl, TitmanTrolled, Tkbx, Tobias Bergemann, TobyGoodwin, Toddy1, Tom Morris, Tomato expert1, Tommy, Tony Holkham, Tony Sidaway, Tony1, Topbanana, Torsch, Tow, Tpbradbury, Tradereddy, TrollDeBatalla, Trusilver, Truthflux, Tsgesq, Tuchomator, Tunacanoe, Tutelary, TwoEscarf, Twofivethreetwo, Tymothy, Tyraz, U4ealongan, Ulflund, Ultra two, UncleNinja, UneCanardNoire, Unforgiven24, Unknownlight, Urunak, Utopiah, V4vijayakumar, V8skittles, Vaelor, Vanzandtj, Vcwatcher, Verbaetlittera, Vermillion trade, Verpies, Victorsharpe, Vilalna, VinceSamios, Vinnie james, Vivaldi, Vivek.m1234, Voodoom, Voomoo, Vroo, W1k1p3d1a26, WakiMiko, WaldoJ, Wanders, Wavelength, Wcoenen, Werieth, Werowe, West Coast Gordo, WhereAmI, WhisperToMe, Wiki Wikardo, Wiki man 195, WikiDonn, WikiTryHardDieHard, Wikifanman33, Wikipelli, Wikiyakapoola, WinampLlama, Winston Chuen-Shih Yang, Wittylama, Wuerzele, WwATuu, X-Fi6, XP1, XTC99LLC, Xiphosurus, Xoviat, Xyzzyplugh, Yakushima, Yintan, Yiosie2356, Your Lord and Master, Ypnypn, Ysangkok, YuMaNuMa, YubbaDoo, ZAB, Znmeb, Zntrip, Zombiejesus, Zootboy, Zootreeves, Zulfikar ramzan, Zundark, Zythe, Zzedar, Érico Júnior Wouters, 1155 ,.הומר א anonymous edits

Bitcoin network *Source*: https://en.wikipedia.org/w/index.php?oldid=614359165 *Contributors*: Argumzio, Benlisquare, Bloukingfisher, Canton, Charlie7705, Checkingfax, Chris the speller, ChrisGualtieri, Chrisarnesen, Cliff12345, Closeapple, Coffeeandcode, Collabi, Dairhead, Dalukasio, Deepbit, Delldot, Divinity76, Dree12, Eliteware, Enderminh, Epicgenius, FOXIBOX, Falkirks, Giraffedata, Graingert, Hannasnow, JDLade, Jack Greenmaven, Jarble, Jprg1966, JustinClarkCasey, Kevin514, KyleLandas, Litecoinguy, MARIODOESBREAKFAST, Magioladitis, MeekMark, Mmeijeri, Mortense, Observer6, Oren0, Ozzke, Pburka, QuentinUK, Rezonansowy, Solarix, Suumitgeek, Targaryen, Tmfs10, VinceSamios, Werieth, Wiki.gcc, Wuerzele, Xhienne, Xiangfu, ZAB, 33 anonymous edits

Certificate authority *Source*: https://en.wikipedia.org/w/index.php?oldid=614512913 *Contributors*: A520, Aapuch, Abune, Acodring, Ahill.four, Ale2006, AlephGamma, Alistair1978, Annastacy, Anon lynx, Armandeh, ArnoldReinhold, Astonishment, Azure, BAlfson, Banano03, Bender235, Bgwhite, Bigjdady, Bmearns, Borgx, Breadtk, Brookehamilton, Btyner, Cameronpit, Camjuar, Codename Lisa, Comfortably Paranoid, Cryptoki, Damian Yerrick, Dav-FL-IN-AZ-id, Dawnseeker2000, Desummoner, Digi-cs, Doedoejohn, DouglasCalvert, Druckles, Edwulf, Eiger3970, Eiku, Espadrine, ExportRadical, Femto113, Fluff, FrankAndProust, Fredrik, GCW50, Gadiandi, Gerry Ashton, Ginsuloft, Globalsign, Gorgonzilla, Guido Arnold, Haakon, Halloko, Howardjp, Hu12, Iida-yosiaki, Intgr, Jackzhp, Jc3s5h, Jeremy Reeder, Jkl, Jleedev, Jnc, Jncraton, JohnHinsdale, JonHarder, Jrtayloriv, Kenimaru, Kenschry, Kevjudge, Kingturtle, LFaraone, Lakshmin, Ldardini, Leobold1, Leotohill, Lezek, Lijnema, LittleBenW, Lonaowna, Lotje, MER-C, Magioladitis, MajorVariola, Manawar4, Martin.komunide.com, MartinMichlmayr, Matt Crypto, Mcgyver5, Metalim, Meteor sandwich yum, Michael Hardy, Mikez302, Mnemo, Mortense, Mr. Sinister, MrWeeble, Mrbakerfield, Nabber00, Nczempin, Nealmcb, Nickdc, Nickyus, Noq, Nurg, OpinionPerson, Osamanaseem, Osndok, Pafciu, Pde, Peashy, PedanticSophist, Ppelleti, Pradeep itan, Primetomas, Rahul.kokcha, Ram Moskovitz, Rchandra, Reuben, Rich Farmbrough, Rmeneda, Robertrem, RoxyFlox, Ruud Koot, SEWilco, Saucepan, Seungbaeim, Sinafe, SiobhanHansa, Slowtalk, Slrawlins, Smyth, Someone not using his real name, Sroylance, Startcom, Superm401, Sweerek, Texaswebscout, The Thing That Should Not Be, Thenthornthing, Tide rolls, Tjdw, Toniher, TonyW, Vanished user 1234567890, Velle, Vlad, Webguynik, Widr, Wilfrednilsen, Ww, Xanga, Yadirh, Yaronf, Zazpot, Zundark, Île flottante, 204 anonymous edits

Chord (peer-to-peer) *Source*: https://en.wikipedia.org/w/index.php?oldid=599142316 *Contributors*: AL SAM, Ahoerstemeier, At2000, AxelBoldt, Beniz, BigChicken, Blaisorblade, Bpringlemeir, BrokenSegue, Chauttm, ChrisGualtieri, Cwolfsheep, CyberSkull, Dalstadt, Darkov, DavidCary, Dcoetzee, Dewritech, Duncan.Hull, Frank1634, Gilliam, GregorB, Karmastan, Kate, Kbrose, KnightRider, LittleDan, Lobner, M, M.B, Mcaramel, Mdd, MeltBanana, MementoVivere, Mgrinich, Miami33139, Mindmatrix, MithrandirMage, Miym, Mnulph, Neilc, Nethgirb, Nikai, Nkour, Quadell, Raptium, Rjloura, Sparky132, Stjones86, Sun Creator, Surachit, Sweet tea van, Tetrashima, That Guy, From That Show!, TittoAssini, Toufeeq, TripleF, Uhdeagle, Vectro, Wafulz, 90 anonymous edits

Complex network *Source*: https://en.wikipedia.org/w/index.php?oldid=607102835 *Contributors*: Abelfly, AdjustShift, AnAj, Andreas Kaufmann, Apocapoc, Apsimpson02, BD2412, BarroColorado, Bearcat, Bellagio99, Binksternet, Biocom, CesarB, Chicagoian, Ciphers, Davepape, Dggreen, DigitalTeleCom, Dreftymac, Edmanley, Eekerz, Fheyligh, Franciscofisc, Garfield Garfield, Gcalda, Gerel, Ggeorgie, Giftlite, Glich, Gonzalezdiazh, Greatal386, Hu12, Imersion, Inwind, JFromm, JamesBWatson, Jfmendes, JimR, Jshadias, Katharineamy, Ketiltrout, Kku, Kmote, Ldfcosta, Levineps, Loodog, Madcoverboy, Maltusnet, Mattg82, Mdanziger, Meekohi, Mentatseb, Michael Hardy, Michael Slone, Mihai Damian, Mikeblas, MrOllie, Myasuda, NYNetwork, Niazim1, Nog33, OMurgo, Oleg Alexandrov, Oshanker, Paintitblack ft, Panchobook, Pdecalculus, Phix7, Pledger166, Pseudomonas, Pucicu, R'n'B, Reyk, Rjwilmsi, Ronz, Sazaedo, ShelfSkewed, Skatalites, Skynet1, Smoke73, Snorri, Supernet, Suvi Murugan, Turgidson, VZakharov, Wht43, Wikiexp52, Wilsonthongwk, Wireader, Zengxiaohanzxh, 272 anonymous edits

Computer network *Source*: https://en.wikipedia.org/w/index.php?oldid=614326195 *Contributors*: *Kat*, 10metreh, 149AFK, 16@r, 1966batfan, 28421u2232nfenfcenc, 28bytes, 2D, 5 albert square, 802geek, A.amitkumar, A520, ACBest, AJCham, ARUNKUMAR P.R, Aajaja, Abraham, B.S., Acnetj, Adam1213, Adam850, Adambro, Addihockey10, AdjustShift, AeonicOmega, Aeonx, Aicchalmers, Aij, Aitias, Akb.pcb, Akendall, Alansohn, Aleenf1, AlexiusHoratius, Alicas, Allens, Alotsanguinedeity, Alpha 4615, Alphachimp, Altaïr, Altruism, Alucard 16, Amirrad.en, Amojobim, Anarcham, Anbu121, AndreasWittenstein, Andrew D White, Andrewcrawford, Andrewman327, Andy16666, Aneah, Angrysockhop, Anna Lincoln, Anonauthor, Anonymi, Antandrus, AnthonyJ Lock, Antiqueight, Anupam, Anupkeskar, Anurag trojan, Aoidh, Apau98, Apparition11, Apteva, Arctic Fox, ArglebargleIV, Arkgee14, Armchair info guy, ArmenSokhakyan, Arthena, Asrosen, Atw1996, Avinash0147, Avoided, Avril0412, AzaToth, BD2412, Banaticus, Barek, Baronnet, Bart133, Bassbonerocks, Beelaj, Bella Swan, Belovedfreak, Bender235, BevinBrett, Bhasrini, Bhny, Bjankuloski06en, Black Falcon, Blanchardb, Blaxthos, Blazerskj7315, BlueDevil, Bluerasberry, Bob f it, Bobby122, Bobo192, Bogey97, Bokunenjin, Bongwarrior, Brandon, Brian Crawford, Brookie, Btilm, BuddhaBubba, Buffbills7701, Bulldog73, Bumpusjames, Bushsf, ButOnMethItIs, Bwmitche, Bxn1358, C777, CBM, CWY2190, CWii, Cachiadavid, Caiaffa, Callanecc, Calltech, Caltas, Camw, Can You Prove That You're Human, Can't sleep, clown will eat me, CanadianLinuxUser, CapitalR, Capitalismojo, Capricorn42, Captain panda, Captain-tucker, Captainreiss, CardinalDan, Cbdorsett, CelticKing77, Cerebellum, CesarB, Cgmusselman, Chad44, Chakkalokesh, Chandlermbing, Chris the speller, ChrisGualtieri, Chrisch, Chriswiki, Chromaticity, Chzz, Clarince63, Clarkcj12, Closedmouth, Cnilep, Cnkids, Coeus559, CommonsDelinker, Corruptcopper, Courcelles, Cpiral, Cpl Syx, Cst17, Cstratacos, Cwilso, Cwoodskareska, Cxz111, Cybercobra, Cyclonenim, D, D. Recorder, D.c.camero, DAJF, DARTH SIDIOUS 2, DBhavsar709, DJBullfish, Dabomb87, Dac04, DakotaDAllen, DaltinWentsworth, DanMS, Daniel Procter, Danielbullis, Danlaycock, Dap263, DarkRetribution7, Darkkloud29, Darth Panda, Dasari12, Davosmith, Dayyanb, DeadEyeArrow, Deadbeef, Decltype, Demonslayer4000, Denisarona, DerHexer, Dewritech, Dgtsyb, Dgw, Diannaa, Dicklyon, Dillbie, Dinkytown, Dipankan001, Discospinster, Djg2006, Djmckee1, Djriley1993, Dlohcierekim, DonWW, Donner60, DoorsAjar, Dotancohen, DoubleBlue, Download, Dreadstar, Dwayne, DyingIce, Dylan620, EarthPerson, Easwarno1, Edward, Eeekster, Eggnogicecream, Egmontaz, Eitheladar, Eleos, Elinruby, Elphion, Emailtonaved, EncMstr, Enigmaman, Epbr123, Epicgenius, Er Komandante, Eraxx, Erdoğan said, Erianna, Eric-Wester, Escape Orbit, EvokeNZ, Excirial, Extransit, FJPB, Faithlessthewonderboy, Faizan, Falcon8765, Faradayplank, Favonian, FayssalF, Fazlurrahman95, Fgrose, Fieldday-sunday, Fiftwekid, Figma, Finnysgay, Fintanmurphy, Fireaxe888, Fireice, Firien, Flewis, Flyer22, Flyingcheese, Fozz79, Fraggle81, Frankie0607, Freedomlinux, Frmatt, Funnyfarmofdoom, Fyyer, Fæ, GLaDOS, GNMC, GRIFFnDOOR, Gail, GandalfDaGraay, GeorgeBarnick, Gholam, Giftlite, Gilliam, Ginsuloft, Glane23, Glenn, GliderMaven, Gnowor, Gogo Dodo, GoingBatty, Gonchibolso12, GoneAwayNowAndRetired, Grafen, Greedo8, Greenrd, GroveGuy, Guggfe, Gurch, Gurchzilla, Gv250, Gyan pokhara, Hammersoft, Happysailor, HarisM, Harryzilber, Harshrateria, Haseo9999, Hcberkowitz, Hchrm, Hdt83, Headbomb, Hellokittylover12, Hemachandra18, Herbythyme, HiDrNick, Hpcanswers, Hut 8.5, Hz.tiang, INkubusse, Ianmillner, Igbe obinna, Ignatzmice, Igoldste, Imroy, Incompetence, Indon, Inseeisyou, Intgr, Ipatrol, Iridescent, IronGargoyle, It4it-wiki, Itusg15q4user, J. Martin Wills, J.delanoy, JBazuzi, JEBrown87544, JLRedperson, JLaTondre, Jackelfive, Jackfork, Jackol, JaconaFrere, Jake Wartenberg, James086, Jan1nad, Japheth the Warlock, Jarble, Jared Preston, Jasper Deng, Jauerback, Jay, Jayakrishnan0804, Jclemens, Jeepday, Jeff G., Jeffrd10, Jer71, Jerzy, Jesant13, Jflabourdette, Jfmantis, Jheyahr, Jianhui67, Jifdsjgfudufdgrgfuhfdhdushdfuahafu, Jim1138, Jimmi Hugh, Jjasi, Jjensen347, Jjron, Jni, Joantorres, Joaquin008, Joelhanley, John Cline, John Stumbles, Johnuniq, JonHarder, Jordanfeldman, Joseph Solis in Australia, Josh3580, Joshua Gyamfi, Joshua Scott, Josve05a, Joy, Jpbowen, Jprg1966, Jx1180, Jyoti.mickey, K.anonymous, K6ka, KGasso, Katieh5584, Kazochi, Kbrose, Keilana, KelleyCook, Kgfleischmann, Kieferb12, Kilopi, King of Hearts, Kingpin13, Kirachinmoku, Kispvtltd, Kms, KnowBuddy, Knowz, Koffieyahoo, Koolabsol, Kotapali Mahesh, Kozuch, Kranix, Krishnavedala, Krj373, Kudret abi, Kureishi, Kushalbiswas777, Kvng, L Kensington, L314t, L33th4x0rguy, LFaraone, Landon1980, Leafyplant, LeaveSleaves, LeinaD natipaC, Levineps, Leye1, Lightmouse, Lights, LilHelpa, Lilac Soul, Lillightning, Limideen, LindsayH, Ling.Nut, Loile0801, Lotje, Lsmll, Lsteinb, Lucky arien2001, Ludovic.ferre, Lugia2453, Luk, Lukeritchie, MER-C, Macbookmiller, Macy, Madcoverboy, Madhero88, Magioladitis, Maismuhanad, Mandarax, Maneendra, Manifestation, Manjax76, Marco94, Marek69, Mark Arsten, MarkBolton, Markdude09, MarkmacVSS, Mascharanas, Masterjamie, Matdrodes, Materialscientist, Mattgirling, MatthewIreland, Mattl2001, Maura Driscoll, Maxwell bernard, McGeddon, McSly, Mdd, Meh1234567, MelbourneStar, Mentifisto, Metricopolus, Michael Angelkovich, Michael93555, Micke-sv, Mikaey, Mike Rosoft, Mike.lifeguard, MikhailVS, Milind m2255, Mindmatrix, Minimac, Minimac's Clone, Misortie, MisterCharlie, Miym, Mlewis000, Mlouns, Mmernex, Mmmeg, Modamoda, Moe Epsilon, Montchav, Montrevux, MorrisRob, Mottsauce, Mr. Wheely Guy, MrSmook, Mrnatural, Mrt3366, MuZemike, Muhammadbabarzaman, Muhandes, MusikAnimal, Mynameiswill, N5iln, NFD9001, NYKevin, Nanzilla, Nathanid, NawlinWiki, Nazi 2007, Neillucas, NeoJustin, Nepenthes, Nethgirb, Netito777, Networkingguy, NewEnglandYankee, Newyorkadam, Ngriffeth, NicatronTg, Nihiltres, Nopetro, Northamerica1000, NorwegianBlue, NoticeBored, NottNott, Novalis, Nrm123, Nsda, Nshunter, Nubiatech, Nurg, Nurlan926, Ojasweesharma, Oliver202, OllieFury, Onure, Opelio, Orange Suede Sofa, Orbst, Oroso, Otolemur crassicaudatus, OverlordQ, Oxymoron83, PAntoni, Padillah, Pajz, Pakakj.20june, Pascal.Tesson, PatrikR, Paul August, Pchov, Pdcook, Pedro, Person1928, Peterwhy, PhJ, Pharaoh of the Wizards, Phatom87, PhilKnight, Philip Trueman, PhilipJS, Phoe6, Piano non troppo, Pigman, Pinethicket, Pingveno, Plaga701, PleaseStand, Polluxian, Porkrind, Porterjoh, Possum, Poweroid, Prari, Prashanthns, Pratyya Ghosh, Promethean, Public Menace, Puffin, Pwarrior, Pxma, Pyrospirit, Quaeler, Quantpole, Qwertyus, Qwyrxian, Qxz, R'n'B, RA0808, RJaguar3, Racerx11, Radar33, RadioFan, Radon210, Raed abu farha, Rahul440, RainbowOfLight, Rajasekharareddykolli, Rananera74, RandomAct, Rangek, Rangoon11, Raven21421, Razarajpoot, Razertek, Rctay, RealityApologist, Reaper Eternal, Red Thrush, Reliableforever, Rememberway, RenamedUser01302013, Renewer, Rettetast, Riana, Ricardoprojects, Rick Block, Riick, Rjd0060, Rmaheshnaidu, Rmosler2100, RoMo37, RobertIllston, Rocketrod1960, Rockonomics, Roland Kaufmann, Rpsjrcpa, Rsrikanth05, Rwwww, Ryan Vesey, RyanCross, Ryulong, SMC, ST47, Saketmitra, Sandeepsp4u, Santoshwriter, Sara007007, Satellizer, Satori Son, Sayedomer, Scetoaux, Scgtrp, SchreiberBike, SchreyP, Sciurinæ, Scottywong, Seaphoto, Seek54, Sephiroth BCR, Sesu Prime, Sfgiants1995, Sgeo, Shadowjams, Shanes, Shenme, Shiwakant.bharti, ShornAssociates, Shulini, Sigma 7, Skarebo, Skizzik, SkyWalker, Skyezx, Slawekb, Slon02, SmartGuy Old, Smokizzy, Snow Blizzard, SoCalSuperEagle, Soap Poisoning, Sofia Koutsouveli, Some jerk on the Internet, Sonjaaa, SpaceFlight89, SpecMode, Special Cases, Spitfire, Spmeyn, SpuriousQ, Ssmorris, StaleOnion, Stausifr, SteinbDJ, Stephan Leeds, Stephenb, SteveO, Storm Rider, Strike Eagle, Stwalkerster, Suffusion of Yellow, Sumitprksh, SuperSlacker, Symetrix, Synchronism, Syrthiss, Tango, Tarif Ezaz, Tbhotch, Teapeat, TechTWI, Technobadger, TehLonelyNinja, Tentinator, Terrek, Testbells, Tgeairn, The Thing That Should Not Be, TheCatalyst31, TheEpTic, Thedjatclubrock, Thejokerface, Theopolisme, Thetaung, Think outside the box, Thkie, Thrindel, Thumperward, Tiddly Tom, Tide rolls, Tigers132, Titoxd, Tobias Bergemann, Todd Peng, Tolly4bolly, Tombomp, Tony1, Topbanana, Touch Of Light, Toussaint, Tpbradbury, Traxs7, Trenwith, TreveX, Trgaz, Triona, Triwbe, Tropryor, Tslocum, Turabsf, Turgan, TutterMouse, TwoTwoHello, TyA, Tyrol5, UBJ 43X, UU, Ulric1313, Ultimarko, Uncle Dick, UncleDouggie, Vamphemu, VandalCruncher, Vanished user 39948282, Vanished user uih38riiw4hjlsd, Vanished user zq46pw21, Vanka5, Velella, Versageek, Versus22, VictorAnyakin, Visaforu, Visor, VooDooChild, Vrenator, W Nowicki, Wa3frp, Wadamja, Waggers, Walter.bender, Wannabemodel, Waryklingon, Watchdog9, Wbm1058, Weariness, Webclient101, Weezey, Welsh, Whereizben, WhisperToMe, Widr, WikHead, Wikipelli, Wikipixel, Wilde Jagd, Wimt, Wine Guy, Winston Chuen-Shih Yang, Wizardist, Wknight94, Wolfkeeper, Woohookitty, Wtmitchell, Wtsao, Wuhwuzdat, Wyn.junior, Xompanthy, YassineMrabet, Yintan, Zac67, Zawthet, Zedex7, Zenibus, Zentraleinheite, Zsoftua, Zzuuzz, טבעת-זרם, ترجمان05, شات صوتي, محبوب عالم, 3361 anonymous edits

Cryptographic hash function *Source*: https://en.wikipedia.org/w/index.php?oldid=612307255 *Contributors*: 83d40m, A930913, AP61, ATBS, Aetheling, Alienus, Amit 9b, Amol kulkarni, Analoguedragon, AndyKali, Anirvan, Apokrif, Appleseed, Arakunem, ArchiSchmedes, Are you ready for IPv6?, Arjun024, ArnoldReinhold, Arvindn, Astronautics, Avinava, BDD, Bachrach44, Basisplan0815, Bdragon, BenFrantzDale, BenjaminGittins, BiT, Blotowij, Bobo192, Bomazi, Bonev, Boredzo, Bpeps, Brian Gunderson, BrianWren, Bryan Derksen, Bsmntbombdood, C4chandu, Caltas, Capricorn42, Cenarium, CesarB, CesarB's unpriviledged account, Chalst, Champloo11, Charles Matthews, Chuunen Baka, Ciphergoth, Ciphergoth2, Clark89, Claw of Slime, Cmcqueen1975, Colonies Chris, Coolhandscot, CryptoDerk, Cube444, CygnusPius, Daemorris, DaishiHarada, Damian Yerrick, Danhash, Danwix, Darts123, DavidCary, Davidgothberg, Dbfirs, Dcljr, Deeb, Doctorhook, Doremo, Download, Ej, Eliz81, Enviroboy, Epicgenius, Erianna, Erodium, Extropian314, FT2, Feedmecereal, Fils du Soleil, Firealwaysworks, Fltnsplr, Fredrik, FrenchIsAwesome, Fresheneesz, Froth, Fuzzypeg, Gaius Cornelius, Garsd, Gavia immer, Ggia, Giftlite, Guruparan18, H2g2bob, Haeinous, Harmil, Hymek, Ianhowlett, Imjustmatthew, Imran, Infomade, Inkling, Instinct, Intgr, JWilk, Jafet, Jamelan, January, Jasonsewall, Javidjamae, JensAlfke, Jesse Viviano, Jianhui67, Jimbobl, Jok2000, Jorge Stolfi, Jsaenznoval, JuanPechiar, Kotra, Kravietz, Kyz, Lambiam, Laurinavicius, Lee Carre, Leobold1, Leonard G., Leszek Jańczuk, Lichtspiel, Lightst, Lotje, Lowellian, Lunkwill, Luzmcosta, MIT Trekkie, Maciej Czyżewski, Mandarax, MarioS, Marios.agathangelou, Maslen, Materialscientist, Matt Crypto, Maurice Carbonaro, Maxal, Maxiwheat, Mdd4696, Mellery, Michael Hardy, Michaelfavor, Mindmatrix, Mitch Ames, Mmernex, MoleculeUpload, Mrand, Musko47, Myria, N5iln, Nbarth, Niceguyedc, Nipisiquit, NoDepositNoReturn, NormHardy, Nxavar, Ohnoitsjamie, Oli Filth, OnBeyondZebrax, Optimist on the run, Osndok, Oswald Glinkmeyer, Ott2, Oxwil, Paranoid, Patriot8790, Paul Ebermann, Paul Foxworthy, Pgimeno, Phil Boswell, Pingveno, Plfernandez, PseudonympH, Quelrod, Quintus, Rabbler, Ratsbane, RavelTwig, Rich Farmbrough, RistoLaanoja, Robertgreer, RobinK, Rsrikanth05, Ruptor, SMC, ST47, Salvidrim!, Samboy, Schlapfm, Schneier, ShaunMacPherson, Sietse Snel, Simetrical, Sligocki, SmallPotatoes, Sroc, SteveJothen, Strongriley, Superm401, SwisterTwister, Sylletka, TAKUMI YAMAWAKI, Taxman, Teddyb, Tehsha, Tentinator, Thinking Stone, Tjfulopp, Tjwood, Tom Lougheed, TooTallSid, TreasuryTag, Tsihonglau, Twotwotwo, Twredfish, Twsx, VBGFscJUn3, Vssun, WLU, Wedgefish, Whisky drinker, Widr, Wigie, Wikipedian314, Wolfmankurd, Wordsmith, Ww, Xvsn, YUL89YYZ, ZipoBibrok5x10^8, Zsinj, Zundark, קול ציון, תומר א., مونا بشيري, 362 anonymous edits

Cryptographic nonce *Source*: https://en.wikipedia.org/w/index.php?oldid=590254811 *Contributors*: Amr.rs, Arvindn, Beefyt, Beneluxboy, Cloudswrest, ComputerGeake, Cybercobra, DBooth, Daheadhunter, Davidgothberg, Elf, Finlay McWalter, Garsd, Gil Gamesh, Hagoth, Jlittlet, Julesd, Lee J Haywood, Loadmaster, Løde, Maurocchi, Mellowmarshall, Nabieh, Nikos 1993, Oren0, Petr Matas, Qutezuce, Red Slash, Rrburke, Schmiditwo, Svick, TreasuryTag, Tyleregeto, Vroo, William Avery, Wrs1864, ZeroOne, 36 anonymous edits

Cryptography *Source*: https://en.wikipedia.org/w/index.php?oldid=614950086 *Contributors*: (, 00mitpat, 137.205.8.xxx, 1dragon, 2D, 2pem, 5 albert square, A bit iffy, ABCD, APH, Aaron Kauppi, Abach, Abdullais4u, Academic Challenger, Acroterion, Adam7117, AdjustShift, Adjusting, Aeon1006, Agateller, Ahoerstemeier, Aitias, Akerans, Alan smithee, Alansohn, Alidad1261,

Alieseraj, Altenmann, Anclation, Andres, Antandrus, Antura, Anubhab91, AnyFile, Anypodetos, Aoidh, Ap, Ap375-NJITWILL, ArnoldReinhold, Artusstormwind, Arvindn, Auyongcheemeng, Avenue, AxelBoldt, B Fizz, BeEs1, Beland, Beleary, Bender235, Bernd Paysan, BiT, Bidabadi, Bill431412, Binksternet, Bkassay, Blackvault, Bletchley, Blethering Scot, Blokhead, Bobanobahoba, Bobo192, Boleslav Bobcik, Bongwarrior, Boyette, Brad101, Brad7777, Braincricket, Brandon5485, Branger, Brentdax, Brighterorange, Brion VIBBER, Browner87, Bte99, CBR1kboy, CL, CRGreathouse, CWY2190, Cadence-, Calcton, Caltas, Capricorn42, Card, Cayelr, Cedrus-Libani, Cferrero, Chandu iiet, Chaojoker, Chaos, CharlesHBennett, Chas zzz brown, ChemGardener, Chetvorno, Chris 73, Chris the speller, Chrisbx1, Chungc, Church074, Ciacchi, Cimon Avaro, Ckape, ClementSeveillac, Clickey, Closedmouth, Cluckkid, CommonsDelinker, Complex01, Conversion script, Corpx, Cralar, CrazyComputers, Crazycomputers, Crazyquesadilla, Crownjewel82, CryptoDerk, Cyde, Cyrius, DPdH, DRLB, DSRH, DVD R W, Dachshund, Daedelus, DanDao, Dante Alighieri, DarkBard, Darth Mike, Dave6, David Eppstein, David G Brault, David Nicoson, DavidJablon, DavidSaff, DavidWBrooks, Davidgothberg, Dbenbenn, DeathLoofah, Deeahbz, Delfeye, Delifisek, Demian12358, Denis Fadeev, Detach, Dev-NJITWILL, DevSolar, Dgies, Dgreen34, Dhar, Dhollm, Dhp1080, Diannaa, Dienlei, Difbobatl, Dlrohrer2003, Dmsar, Dogposter, Donner60, Donreed, Dori, DoubleBlue, DouglasCalvert, Dr. Sunglasses, Dr1819, DrDnar, Drift chambers, Drink666, Dtgm, Duncan.france, Dysprosia, Dzu33, E=MC^2, Echartre, Ed g2s, Edggar, Egg, Eltzermay, Endymi0n, Episcopus, Euphoria, Evercat, Excirial, Exir Kamalabadi, FF2010, Faizan, Farnik, Fatespeaks, Ferdinand Pienaar, Fg, Fieldday-sunday, FrankFlanagan, Frapter, Fredrik, FreplySpang, Fshtea, G Rose, GABaker, Gaius Cornelius, Galoubet, Geni, Geoffreybernardo, Geometry guy, Georg Muntingh, GeorgeLouis, Ghettoblaster, Giacomo.vacca, Gianfranco, Giftlite, Gilabrand, Gilliam, GimmeFuel, Glenn, Goatasaur, Gogo Dodo, GoingBatty, Gouldja, Gowr, Grafikm fr, Graft, Graham87, Grandexandi, GrantWishes, Greatdebtor, Gubbubu, Gus Buonafalce, H2g2bob, Hagedis, HamburgerRadio, Hamerbro, Hamiltondaniel, Hans Dunkelberg, HappyCamper, Harley peters, Harryboyles, Harrymph, Havanafreestone, Hdrugge, Hectorian, Heinze, Herk1955, Hermitage17, Heron, Heryu, Hibbleton, HippoMan, Hmains, Hollerme, Homunq, Hong ton po, Hoo man, Hotcrocodile, Htaccess, Hut 8.5, IAMTrust, Iago4096, Immunize, Impaciente, Imran, InShaneee, Infomade, IsmaelLuceno, Ivan Bajlo, Ixfd64, J.delanoy, JIP, JRM, JYolkowski, JaGa, Jacobolus, Jacoplane, Jacopo, Jafet, Jagged 85, Jahoe, James Banogon, Jameshfisher, Japs 88, Jdforrester, Jeffq, Jeremy Visser, Jericho4.0, Jessicag12, Jianhui67, Jimgawn, Jimmaths, Jitse Niesen, Jj137, Joel Rennie, Joel7687, John Vandenberg, John Yesberg, John of Reading, JohnBlackburne, JohnDoe4000, JohnMac777, Johnuniq, Joineir, Jok2000, Jordanbailey123456789, Jorunn, JoshuaZ, Jpmelos, Jramio, Jrockley, Jsdeancoearthlink.net, JuPitEer, Judgesurreal777, Julesd, K1Bond007, KConWiki, KFP, Kakofonous, Kapanidze, Karada, Kazov, Kernel.package, King of Hearts, Komponisto, KooIkirby, Kozuch, Krj373, Ksn, Kushalbiswas777, Kuszi, Kylet, LC, Lakshmin, LaukkuTheGreit, Laurentius, LeaveSleaves, Legoktm, Leranedo, Leszek Jańczuk, Liftarn, Lightlowemon, Lightmouse, LoganK, Logologist, Lotje, Lotte Monz, Lpmusix, Ludraman, Luna Santin, Lunchscale, M5, Mabdul, Macedonian, Madchester, Madmax8712, MagneticFlux, Mangojuice, ManiF, Manop, Manscher, MarSch, MarioS, Mark Renier, Mark Zinthefer, Maslen, MasterOfHisOwnDomain, Materialscientist, MathMartin, Matt Crypto, Mauls, Maurice Carbonaro, Maury Markowitz, Mav, MaxMad, Maxt, Maxí, Mayevski, Mblumber, Mboverload, Meco, Meelar, Mercurish, MezzoMezzo, Michael Devore, Michael Hardy, Michael miceli, Michi.bo, Mike Segal, Mindmatrix, Minna Sora no Shita, Miserlou, Misza13, MithrandirAgain, Mm32pc, Mmernex, Mmustafa, Mo0, Mobius, Mohdavary, Mojo Hand, MoleRat, Molerat, Monkeyman, Monkeymonkey11, MooCowz69, Moonriddengirl, Moverton, MrOllie, Mrstoltz, Msaied75, Msanford, Msh210, Mspraveen, Mukkakukaku, Mvoorzanger, Myria, Mzajac, NHRHS2010, NYKevin, Nageh, NawlinWiki, Neilc, Nemu, Ner102, Netoholic, Nevilley, Nfearnley, Nightswatch, Nihil novi, Nihonjoe, Nikai, Nikola1891, Ninja1123, Norwikian, Novum, NoychoH, Ntsimp, ONEder Boy, Oddity-, Oerjan, Ohnoitsjamie, Oiarbovnb, OldishTim, Oleg Alexandrov, Omicronpersei8, Omnipaedista, Orhanozkilic, Ortolan88, OutRIAAge, PPerviz, Pakaran, Pale blue dot, Pallas44, Papergrl, Patrick, Paul August, Pbsouthwood, Pegasus1138, Per Honor et Gloria, Permethius, PerryTachett, Peruvianllama, Peter Delmonte, Peter Isotalo, Peyna, Pezanos, Phayzfaustyn, Phr, PierreAbbat, Pigman, Pinethicket, PleaseSendMoneyToWikipedia, Plrk, Porkolt60, Powerslide, Praveen pillay, PrimeHunter, Proidiot, Prsephone1674, Quartekoen, Qxz, R Math, R00stare, Radagast3, Raghunath88, Raseman, Rashad9607, Rasmus Faber, Raul654, Rbellin, Rdsmith4, RedWolf, Reddy212, Rememberway, Remi0o, Rettetast, Revolver, Rich Farmbrough, Rich5411, Richard Burr, Richwales, Rjwilmsi, Roadrunner, RobertG, Robertsteadman, Rochdalehornet, Rohitnwg, Ross Fraser, RudyB, Ryanwammons, SCCC, SDC, SJP, Saber Cherry, Sachin Hariharan1992, Saga City, Salih, Sam Edward c, Sansbras, Saoirse11, Schlafly, Schneelocke, Sciguystfm, Scircle, Seamusandrosy, Sean.nobles, Secrefy, Securiger, Serlin, Sfdan, Shammie23, Shenron, Shirik, Shmitra, Siddhant, Sidmow, Sietse, SimonLyall, Sinar, Sjakkalle, Skamecrazy123, Skippydo, Skizzik, Skunkboy74, Slayemin, Slightsmile, Slipperyweasel, Smallman12q, Snori, Snowolf, Socialworkerking, Some jerk on the Internet, Sonam.r.88, Sovietmah, Speck-Made, Sprishi, Starwiz, Stefanomione, Stephen G. Brown, Stevertigo, StevieNic, Strigoides, Super-Magician, Superm401, Surachit, Sv1xv, Symane, Syphertext, TJRC, Ta bu shi da yu, TakuyaMurata, Tao, TastyPoutine, Taw, Ted Longstaffe, TedColes, Tempshill, Tentinator, TerraFrost, The Anome, The wub, The1physicist, Thehockeydude44, Theresa knott, Thiseye, Thomas.W, Thumperward, TiMike, Tide rolls, Timrollpickering, Timwi, Tobby72, Tobias Bergemann, Tom Reedy, Tomasz Dolinowski, Touisiau, Tpbradbury, Treisijs, Turnstep, TutterMouse, Tw mama, Udopr, Uncle Lemon, Unmerklich, Uriyan, Vadim Makarov, Vanished user 1234567890, VernoWhitney, Vkareh, Volfy, Vrenator, Vuong Ngan Ha, Vyse, WPIsFlawed, WannabeAmatureHistorian, Wavelength, Wbrameld, Welsh, Wereldburger758, Wes!, WhatisFeelings?, Who-is-me, WiHkibew, Widr, Wiki5d, Wikiborg, Wikiklrsc, WikipedianMarlith, Wilson.canadian, Winston Chuen-Shih Yang, WojPob, Wolfkeeper, Wonderstruck, Woohookitty, Wrs1864, Wsu-dm-a, Ww, XP105, Xnquist, Xompanthy, Xtremejames183, YUL89YYZ, Yakudza, Yekrats, Yintan, Yob kejor, ZachPruckowski, Zeno of Elea, Ziko, ZipoBibrok5x10^8, Zntrip, ZoeB, Zundark, Zyarb, ‎ب, كاشف عقيل ,محمد علي العراقي, 919 anonymous edits

Decentralization *Source*: https://en.wikipedia.org/w/index.php?oldid=613614671 *Contributors*: 237online, Aaclarkcdr, Abhishek, AdjustShift, Akuindo, Allens, Andy Dingley, Ariedartin, AstroChemist, Beetstra, Bender235, Bgwhite, Biruitorul, Bmicomp, Brianga, Briskwalk, Byelf2007, Cacophony, Camembert, Carolmooredc, Chendy, Chester Markel, Chopchopwhitey, Chris the speller, Cometstyles, Cordless Larry, D.h, D6, Daniel Mueller, Darth Panda, Dawn Bard, Dbabbitt, DeathShadow24, Dickintine, Dikaalnas, DoriSmith, Dreispt, Dyne316, Dysprosia, Ecdpm, Eduen, Elipongo, Emefaush, Enchanter, Enoch the red, Errarel, Europewatchman, Eustress, Feil0014, Ffaker, Fraggle81, Francs2000, Gaius Cornelius, Ginsuloft, Good Olfactory, Hallows AG, Hetar, Icairns, Jacob1207, Je.est.un.autre, Jim.henderson, John of Reading, Jonathan Winsky, Jonathandeamer, Jonesey95, Josh Parris, Kbs.sidhu, Kevin Baas, Khalid, Lexor, LilHelpa, Magicxcian, Mais oui!, Materialscientist, Matthew Yeager, Maurreen, Maziotis, Mbiama Assogo Roger, Mdd, Michael Hardy, Michaelcornelle, Mild Bill Hiccup, MirkoS18, MushroomCloud, Myasuda, Mydogategodshat, NebY, Neoconfederate, Nikodemos, Nilmerg, Nirvana2013, Ohnoitsjamie, PM Poon, Parhamr, PetrochemicalPete, Phil Boswell, Pm master, Pnm, Prachi.jnu, Prof.matthew, Pterre, Quarl, QuickClown, R. S. Shaw, RainbowCrane, Rajankila, Rearden9, Reedy, RenamedUser01302013, Rjwilmsi, Rlry82, Roadrunner, Ruraldecentralization, Sarrus, Shawnc, Skomorokh, Smallman12q, SorenSoren, Srich32977, Srleffler, Sun Creator, TMC1982, Tabletop, The Missing Piece, Tmdspfld, Tonyfaull, Trident13, Trovatore, Vchorozopoulos, Wile E. Heresiarch, Xerographica, Yakoo, Yoyo12123, Zachorious, Zoz, Јованвб, Табалдыев Ысламбек, 186 ,טרול רפאים anonymous edits

Digital signature *Source*: https://en.wikipedia.org/w/index.php?oldid=614082885 *Contributors*: .:Ajvol:., 5 albert square, Aalsheb@hotmail.com, Acdx, Acprisip, Adrian, Adriatikus, Ahoerstemeier, Akarkera, Akkida, Akma72, AlephGamma, Alex ruff, Alex756, Alvin Seville, Annastacy, Anomie, Antti29, Armando, ArnoldReinhold, Arthur Rubin, Arvindn, AxelBoldt, Bah23, Bentogoa, BernardZ, BiT, Bigdavesmith, Binarybits, Bluemoose, Bobo192, Bradshaws1, Bryan Derksen, Bsmith2008, Bugbee, Burlefot, C0dergirl, CKlunck, Cabouchonne, Calder, Can't sleep, clown will eat me, Canadian Monkey, CecilWard, Ceyockey, Chaojoker, Chealer, Cherkash, Chris the speller, ClementSeveillac, Cpartners, Crazycomputers, Curps, Cybercobra, DARTH SIDIOUS 2, DVdm, David Shay, DavidJablon, Davidgothberg, Dawnseeker2000, Devayon, Dicti0nary0, Dipendra Pal, Discospinster, DivineAlpha, Dokaspar, DonkeyKong64, Dzu33, Edcolins, Edward, Elocktech, Epbr123, Equendil, Erianna, Etienne.navarro, Excirial, Farbicky, Fdbtwiki, FlyingToaster, Frau K, Fritz Saalfeld, Funky Monkey, GCW50, Gadgetinspector, Gaius Cornelius, Gbiten, Gbroiles, Gcdinsmore, Gdo01, GeoffMacartney, Gerry Ashton, Gerson1875, Ggia, Ghettoblaster, Giftlite, Gilliam, Ginsengbomb, Gogo Dodo, GoingBatty, Guysha, Hamdhan777, Hariva, Hccrle, Hymek, Infomade, Intgr, Isaacbowman, Isnow, J-Star, Jaceto, Jc3s5h, Jkcpop, Joffeloff, Jok2000, Jonathansuh, Josang, Josve05a, KAtremer, Klamber, Klausner, Kozuch, Labscientist, LapoLuchini, Liberty4u, Ligulem, Liisatln, Liridon, Lotte Monz, Lošmi, Lugia2453, MER-C, Magioladitis, MajorVariola, Malinaccier, Mangojuice, MarioS, Mark J, Markush8, Martin.langhoff, Materialscientist, Matt Crypto, Mboverload, Mdann52, Memark, Mfwitten, Mgault999, Michael Hardy, Michael James Boyle, Michael miceli, Michallispashidis, Mikez302, Modulatum, Mordemur, Mormegil, MountainMan11, Mouse Nightshirt, MrOllie, Mrceylon, Mrothwell, Myanw, Nachovaca, Nageh, Niceguyedc, Nichalp, Nickj, Nikai, Ninly, Nisiguti, Nonnompow, Ntsimp, Ontargettaxsmile, Osamanaseem, Pak21, Palosirkka, Peashy, Pekaje, Pharaoh of the Wizards, Philip Trueman, PierreAbbat, Pinethicket, Prabhuhari, ProfessorBaltasar, Rahulbud, Ramel.levin, Raven4x4x, RedWolf, RedWordSmith, Reetep, RexNL, Rhoerbe, Rhsatrhs, Rich Farmbrough, Rijndael, Robinsonc78, Robman94, RobyWayne, Rohasnagpal, Ronhjones, Ronz, Rootxploit, SCWM, Sam Hocevar, Samueldc, Sandstein, SapiusN, SchnitzelMannGreek, Scottham15, Seizethedave, Serkon, Settinghead, Shadowjams, Shoseido, Siddhant, Sintaku, Sjjupadhyay, Skippydo, Some jerk on the Internet, Sparky132, Spitfire, Sspecter, Steinsky, Stephenb, Stuartroebuck, Sunnykapoor135, Svetovid, Tabbelio, Taejo, TastyPoutine, Technopat, Thryduulf, Tkynerd, Tobias Bergemann, Toolnut, TreasuryTag, Turnstep, Txuspe, TyA, VBGFscJUn3, Varuna, Vary, Vashtihorvat, Venkatakrishnanks, Viola16, ViriiK, Vito Genovese, Viv5431, Vrenator, Vssun, WaltTFB, WereSpielChequers, Widr, Wiki alf, Wine Guy, Wintermute314, Wotnarg, Wtmitchell, Ww, Xiquet, Xyzmo, Ykhwong, Z.E.R.O., Zazpot, Zeimusu, Zorro CX, 617 anonymous edits

Distributed hash table *Source*: https://en.wikipedia.org/w/index.php?oldid=609753550 *Contributors*: Alfio, Allquixotic, Altered Walter, Anescient, Anthony, Apoc2400, AstReseach, Athymik, Baa, Bernhard Bauer, Bpringlemeir, Br, Brkt, Bryan Derksen, Bubba hotep, Cacheonix, Cburnett, Charles Matthews, ChrisGualtieri, Chrismiceli, Chy168, Clsin, Cojoco, Corti, CosineKitty, Crystallina, Cwolfsheep, CyberSkull, DJPhazer, DerHexer, Diberri, Dsimic, Dto, Edward, Ehn, Elauminri, Elf, EmilSit, Enzo Aquarius, Eras-mus, Eric.weigle, Erik Sandberg, F, Fcrick, FlyHigh, Frap, GPHemsley, Garas, Gary, Godrickwok, Goofi1781, Gpierre, Greywiz, Götz, Haakon, Hadal, Happyrabbit, Harryboyles, Hbpencil, Hu12, Iammisc, Imeshev, Ingenthr, Intgr, Irrbloss, Irvine.david, Itai, Jamesday, Jda, Jensocan, Jnlin, John13535, Johnleach, JonHarder, Karnesky, Kbrose, Khalid hassani, Khiladi 2010, KnightRider, Knuckles, Liamzebedee, Luqui, M.B, MMuzammils, Mabdul, Maix, Mcld, Md4567, Mdd, Meneth, Mgrinich, Michael miceli, Mike Rosoft, MikeSchmid111, Minority Report, Miym, Monkeyjunky, Morte, Mpias, Nad, Naff89, Nazlfrag, Nealmcb, NeilN, Nethgirb, Neustradamus, Nhorton, Noodles-sb, OPless, Ohnoitsjamie, Old Death, OrangeDog, Oskilian, PetrIvanov, Pgk, Phil Boswell, Raghavendra.talur, ReinoutS, Retsejesiw, Roger pack, Rogerdpack, Ronz, Sae1962, Sanpitch, Scgtrp, Seano1, Search4Lancer, ShaunMacPherson, Shentino, Squalho, Ssspera, Tabletop, TakuyaMurata, TedPavlic, The Anome, The Belgain, TheJosh, Thingg, Thv, Tomaxer, Tombeo, Tomdo08, UtherSRG, W Nowicki, Wayne Slam, Wk muriithi, X1987x, Xiong Chiamiov, Xosé, Xubupt, Yramesh, 308 anonymous edits

Graph (mathematics) *Source*: https://en.wikipedia.org/w/index.php?oldid=614566252 *Contributors*: 3ICE, A.amitkumar, ABF, Abdull, Ahoerstemeier, Aitias, Ajraddatz, Aknorals, Akutagawa10, Algont, Altenmann, Amintora, Anabus, Anand jeyahar, Andrewsky00, Andros 1337, Andyman100, Athenray, Barras, BenRG, Bethnim, Bhadani, BiT, Bkell, Bobo192, Booyabazooka, Borgx, Brentdax, Burnin1134, CRGreathouse, Can't sleep, clown will eat me, Catgut, Cbdorsett, Cbuckley, CentroBabbage, Ch'marr, Chewings72, Chinasaur, Chmod007, Chris the speller, Chronist, Citrus538, Cjfsyntropy, Cornflake pirate, Corti, Crisófilax, Cwkmail, Cybercobra, D.Lazard, DARTH SIDIOUS 2, DRAGON BOOSTER, Danrah, David Eppstein, Davidfstr, Dbenbenn, Dcoetzee, Ddxc, Debamf, Debeolaurus, Den fjättrade ankan, Deor, Dicklyon, Discospinster, Djk3, Dockfish, Dreamster, Dtrebbien, Dureo, Dysprosia, E mraedarab, Edgars2007, Editor70, Ekren, Erhudy, Eric Lengyel, Falcon8765, Gaius Cornelius, Gandalf61, Gauge, Gene.arboit, George100, Giftlite, Grolmusz, Gutin, Gwaihir, HMSSolent, Hairy Dude, Hannes Eder, Hans Adler, Hans Dunkelberg, Harp, Headbomb, Henry Delforn (old), Huynl, Ijdejter, Ilia Kr., Ilya, Inka 888, Insanity Incarnate, J.delanoy, JNW, JP.Martin-Flatin, JaconaFrere, Jason Quinn, Jeff Erickson, Jiang, Joeblakesley, Joel B. Lewis, Jokes Free4Me, Jon Awbrey, Jon har, JonDePlume, Joriki, Josve05a, Jpeeling, Jpiw, JuPitEer, Jwanders, Karada, Karlscherer3, Kerrick Staley, Kine, Kingmash, Knutux, Kracekumar, Kruusamägi, Kuru, L Kensington, Lambyte, Liao, Libcub, LuckyWizard, Lugia2453, MER-C, MRG90, Maghnus, Magister Mathematicae,

Mahanga, Manning Bartlett, Marc van Leeuwen, Maschen, Mate2code, Materialscientist, MathMartin, Matt Crypto, Mayooranathan, McGeddon, McKay, Mdd, Michael Hardy, Michael Slone, MikeBorkowski, Minder2k, Mindmatrix, Mitmaro, Miym, Mountain, Mrjohncummings, Myasuda, Mymyhoward16, Nat2, Neilc, Netrapt, Nihonjoe, Nina Cerutti, Nowhither, Nsk92, Ohnoitsjamie, Oliphaunt, Oxymoron83, Pafcu, Paleorthid, Patrick, Paul August, PaulTanenbaum, Peter Kwok, Peterl, Phegyi81, PhotoBox, Pinethicket, Possum, Powerthirst123, Prunesqualer, Quaeler, R'n'B, Radagast3, RandomAct, RaseaC, Repied, Requestion, Rettetast, Rgclegg, Rich Farmbrough, Rjwilmsi, RobertBorgersen, Robertsteadman, RobinK, Royerloic, Salix alba, Sdrucker, Shadowjams, Shanel, Siddhant, Silversmith, SixWingedSeraph, Someguy1221, SophomoricPedant, Sswn, Stevertigo, Stevetihi, Struthious Bandersnatch, Stux, Suchap, Super-Magician, Svick, THEN WHO WAS PHONE?, Tamfang, Tangi-tamma, Tbc2, Tempodivalse, Tgv8925, The Anome, Theone256, TimBentley, Timflutre, Timrollpickering, Tman159, Tobias Bergemann, Tom harrison, Tomhubbard, Tomo, Tompw, Tomruen, Tosha, Tslocum, Twri, Tyw7, UKoch, Ulric1313, Urdutext, Vacio, Vanischenu, Vaughan Pratt, VictorPorton, Voedin, Void-995, W, Wandrer2, Wesley Moy, West.andrew.g, Wgunther, White Trillium, WikiDao, Wikidsp, Willking1979, Wknight94, Wshun, XJaM, Xavexgoem, Xiong, Yath, Yecril, Yitzhak, Ylloh, Yloreander, Zachlipton, Zaslav, Zenek.k, Zero0000, Zocky, Zven, Канеюку, Пика Пика, 466 anonymous edits

Hash table *Source*: https://en.wikipedia.org/w/index.php?oldid=613350441 *Contributors*: ASchmoo, Aberdeen01, Acdx, AdamRetchless, Adrianwn, AdventurousSquirrel, Ahoerstemeier, Ahy1, Ajo Mama, Akuchling, AlecTaylor, Alksub, Allstarecho, Aloksukhwani, Altenmann, Andreas Kaufmann, Antaeus FeIdspar, Antaeus Feldspar, Anthony Appleyard, Anurmi, Apanag, Arbalest Mike, Arlolra, Arpitm, Ashwin, Askewchan, AxelBoldt, Axlrosen, AznBurger, Baka toroi, Baliame, BenFrantzDale, Berean Hunter, Bevo, BlckKnght, Bobo192, Bug, C4chandu, CRGreathouse, CWenger, CanadianLinuxUser, CanisRufus, Carmichael, CecilWard, CesarB's unpriviledged account, Cfailde, Cgma, Ched, Cic, Cntras, Cobi, Cometstyles, Conversion script, Cribe, CryptoDerk, Cryptoid, Cybercobra, DNewhall, DSatz, Damian Yerrick, DanielWaterworth, David Eppstein, DavidCary, Davidfstr, Davidgothberg, Dcoetzee, Decltype, Deeday-UK, Demonkoryu, Denispir, Derek Parnell, Derek farn, Deshraj, Deveedutta, Digwuren, Djszapi, Dmbstudio, Dmcomer, Donner60, Doug Bell, Drbreznjev, Drilnoth, DuineSidhe, Dysprosia, Ecb29, Eddof13, Emimull, EncMstr, Erel Segal, Esb, Everyking, FeralOink, Filu, Floodyberry, Fragglet, Frap, Frecklefoot, Fredrik, Frehley, Fresheneesz, Furrykef, Gadget850, Gareth Jones, Giftlite, Glrx, GregorB, Gremel123, Groxx, Gulliveig, Gustavb, HFuruseth, Happyuk, Helios2k6, Helix84, Hetori, Hosamaly, Hydrox, Ibbn, Iekpo, IgushevEdward, Imran, Incompetence, Intgr, Iron Wallaby, IronGargoyle, JJuran, JLaTondre, Jan Spousta, Jarble, JeepdaySock, Jim1138, JimJJewett, Jk2q3jrklse, Johnuniq, Jorge Stolfi, Josephsieh, Josephskeller, Justin W Smith, JustinWick, Karuthedam, Kastchei, Kaustuv, Kbdank71, Kbrose, Khalid, Kinu, Knutux, Kogorman, Krakhan, Kungfuadam, LOL, LapoLuchini, Larry V, Lavenderbunny, Linguica, Luqui, Luxem, MER-C, Magnus Bakken, MaxEnt, Mayrel, Mcom320, Mdd, Me and, MegaHasher, Meneth, Michael Hardy, Mike.aizatsky, Mild Bill Hiccup, Miles, Mipadi, Mongol, Mousehousemd, MrOllie, Mrwojo, Nanshu, Narendrak, Neilc, Nethgirb, Neurodivergent, Niceguyedc, Nightkhaos, Nixdorf, Nneonneo, Not-just-yeti, Nuno Tavares, ObfuscatePenguin, Om Sao, Omegatron, Oravec, Ouishoebean, Pagh, Pakaran, Patmorin, Paul Kube, Paul Mackay, Paulsheer, Pbruneau, Peter Horn, PeterCanthropus, Pgan002, Pheartheceal, Piccolomomo, Pichpich, Pixor, PizzaMargherita, Pnm, Purplie, QrczakMK, QuantifiedElf, R3m0t, Radagast83, Raph Levien, Rawafmail, ReiniUrban, Rememberway, Rgamble, Rich.lewis, Rjwilmsi, Sae1962, Sam Hocevar, Sandos, Saxton, Scandum, SchreyP, Schwarzbichler, Sebleblanc, Secretlondon, Seizethedave, Shadowjams, Shafigoldwasser, Shanes, Shmageggy, Shuchung, Simonsarris, Simulationelson, SiobhanHansa, Sleske, Sligocki, Sonjaaa, Spacemanaki, Spinningspark, Stannered, Super48paul, Svick, Sycomonkey, T Long, TShilo12, Tackline, TakuyaMurata, Teacup, Teapeat, Th1rt3en, Thailyn, The Anome, TheTraveler3, Thermon, Ticklemepink42, Tikiwont, Tjdw, Tmferrara, Tomchiukc, Tostie14, Trappist the monk, Triston J. Taylor, Triwbe, UtherSRG, Varuna, Velociostrich, Voedin, W Nowicki, Waltpohl, Watcher, Wavelength, Wernher, Wikilolo, Winecellar, Wmahan, Wolfkeeper, Wolkykim, Woohookitty, Wrp103, X7q, Zundark, 557 anonymous edits

Hypercube *Source*: https://en.wikipedia.org/w/index.php?oldid=608713226 *Contributors*: 3.14159265358pi, =CJK=, ANONYMOUS COWARD0xC0DE, Aenar, Algotr, Altenmann, Amichaelnoll, Anion24, Asdfzx, Atlantia, BalooUrsidae, BenBaker, Bluerasberry, Brad7777, Brianhe, Brydustin, CBM, Caltas, ClickRick, Closedmouth, Cybercobra, DancingPenguin, David Eppstein, Deltahedron, Distortiondude, Dr Hudson, Elpizothegreat, EnigmaticCode, Feezo, Frieko9000, Gandalf61, Gavin.perch, Giftlite, Ginsuloft, GreyCat, Guoguo12, Gwpriester, Insomnia64, InverseHypercube, J.delanoy, JFBGT, JackSchmidt, JohnBlackburne, Jon Awbrey, Joule36e5, Jpbowen, Justin W Smith, Jwz, Keenan Pepper, Kiefer.Wolfowitz, Kingpin13, Koavf, Lagrange613, Leevanjackson, LibLord, MSGJ, Macaddct1984, Maksim-e, Mapar007, Mark Arsten, Martynas Šalčiūnas, Mate2code, Mateuszica, MathStar123, Maurice Carbonaro, McSly, Merlinsorca, Michael Hardy, Mithroramir, Miym, MosheA, Nadyes, Neverquick, Nneonneo, Oddbodz, Oleg Alexandrov, Onorem, Paxinum, Pinethicket, Pleasantville, Plexos, Pokajanje, Pol098, Professor Fiendish, Professorjohnas, Proofreader77, Qef, RDBury, Rettetast, Rjwilmsi, Rocchini, Ryanisfootdrums, SCriBu, Sandip90, Spinningspark, Stannered, Tamfang, Tentacles, Tetrabrain, Tetracube, Thane, The Thing That Should Not Be, TheFeds, Thelehmanlip, Tide rolls, Tmp6154, Tomruen, Unique and proud of it, Vipinhari, Wafulz, Wbm1058, Wprlh, Wvbailey, Yugsdrawkcabeht, Zaslav, 182 ,דוד שי anonymous edits

Hypercube graph *Source*: https://en.wikipedia.org/w/index.php?oldid=605420108 *Contributors*: ANONYMOUS COWARD0xC0DE, Bender2k14, Bharel, Booyabazooka, David Eppstein, Ekuurh, G.perarnau, Gaius Cornelius, Giftlite, Grafen, Hermel, Herr Satz, InverseHypercube, Itai, JLeander, JoergenB, Kjkjava, Koko90, LokiClock, Magioladitis, Mysid, Rjwilmsi, Robinh, Salix alba, Stdazi, Tetracube, Tktktk, Tomo, Tomruen, Trovatore, Twri, Xnn, Zaslav, 15 anonymous edits

Kademlia *Source*: https://en.wikipedia.org/w/index.php?oldid=612378687 *Contributors*: Abrahamsonj, AdamBanko, Adoniscik, Alfio, AlistairMcMillan, Alkarex, Analoguedragon, Antigrav, Bender235, Bpringlemeir, Bryan Derksen, Callidior, Canderson7, Carlo.Ierna, Cdc, Chmod007, Cojoco, Confuseme, Connelly, Cyrius, DStoykov, Dadu, DataWraith, Deeahbz, DouglasReay, Dysprosia, Ehn, Elecnix, EoGuy, Family Guy Guy, Firealwaysworks, Fnaucamp, Forage, Franl, Frap, Furrykef, GPHemsley, Gary, Haakon, Harryboyles, Heavenlyhash, Hhielscher, Ian Burnet, Ianneub, InterruptorJones, Intgr, Irvine.david, Isnow, Ivan Shmakov, Jh559, Jossi, Joyous!, Kagato, Kbrose, Kinema, LQ, Leuk he, Liamzebedee, Lion10, Lkesteloot, Lyle Stephan, MCBastos, MER-C, Ma51ne64, Mernen, Michael Birk, Miym, Mkweise, Mnml, Mojodaddy, MoreNet, Nard the Bard, Neonil, Nethgirb, Old Death, Oneiros, Pac72, PacoBell, Palfrey, Paranoid, Phylake, Plisk, Ramix, Schmloof, Schneelocke, Shaddack, ShaunMacPherson, Stormie, Strait, Stybn, Syerfule, Tazpa, Technobadger, Timwi, Titusn, Twsx, Upholder, Viajero, Vindicator, Vorash, Wdyoung, Xzaranos, Zanetu, Zgadot, Zvika, דולב, 石庭豐, 182 anonymous edits

Merkle tree *Source*: https://en.wikipedia.org/w/index.php?oldid=606088566 *Contributors*: Arthur Rubin, Azaghal, Blaisorblade, CesarB's unpriviledged account, Chris the speller, Ciphergoth, Cybercobra, Darthshak, DataWraith, Davidgothberg, Derek Ross, Doctorhook, Dsimic, Duncan.Hull, Feross, GargoyleMT, Gartenzwerg 99, Gdr, Giftlite, Gil Schmidt, Glom2215, Gwern, Haakon, Headbomb, Htmlapps, Intgr, Jeffrey Anthony, Jim Douglas, Jlguenego, Jorgbrown, Jpkotta, Jptvgrey, Kazvorpal, Kmag, Korrawit, Kroepke, M, Marudubshinki, Matt Crypto, Michael Hardy, Nonnompow, Nowhere man, Nysin, Oyigit, Pete279, Pmokeefe, Purebill, Quinn d, Qwertyus, Rashless, Rconan, Rhys.rhaven, RistoLaanoja, Rmashhadi, Ruud Koot, Sklein, Slashdottir, Sperxios, Tamfang, Thattommyhall, Theeldest, Thv, Tr00st, Ullner, Zack, 74 anonymous edits

Peer-to-peer *Source*: https://en.wikipedia.org/w/index.php?oldid=609727662 *Contributors*: -lulu-, 2bar, 4nT0, 4pq1injbok, A8UDI, ABF, ACNS, Aarghdvaark, Aaronchall, Abab99, Abce2, Abrech, Absalom23, Abune, Adammw, Adrian.benko, Aervanath, Ahoerstemeier, Ahtih, Aimini, Akamad, AkashValliath, Akkinenirajesh, Alainr345, Albany NY, Alcuin, Aldie, AlexMyltsev, Alexkon, Aliendude5300, AlistairMcMillan, Allyant, AlyssaG92, Amwebb, Andre Engels, Andrewpmk, Anishsane, Antelan, Anthony, Antoniad, Apoc2400, Arangana, Arm-1234, ArneBab, Arny, Atbk, Atif.t2, Audriusa, Augest, Augman85, Authalic, AxelBoldt, Az1568, Bagatelle, Bahahs, Bane004, Bayerischermann, Ben-Zin, Benad, Bentogoa, Bernino, Bhadani, Bhuston, BiH, Bigjimr, Bikeborg, Binksternet, Bkonrad, Bloodshedder, Bluezy, Bmicomp, Boivie, Bomac, Bondolo, Bongwarrior, Borgx, Bovineone, BrainMagMo, Bramp, Branko, BrokenSegue, BrotherE, Brunelstudy, Bslede, Btrest, Buryfc, Bwishon, CBCompton, Caiaffa, Caltas, Cambrasa, CambridgeBayWeather, Canderson7, CanisRufus, Carlconrad, Carlosguitar, Cazer78, Cctoide, Cecilia Hecht, Celique, CesarB, Cespo4, Ceyockey, CharlesC, Charleswiles, Chazza1113, Chris the speller, ChrisGualtieri, Chuq, Chuunen Baka, Cjdkoh, Ck lostsword, ClanCC, Classicrockfan42, Cldnails, Closedmouth, Cmr08, CoMePrAdZ, CobaltBlue, Codename Lisa, Coelacan, Coinchon, Coldfire82, Collabi, Cometstyles, Computerjoe, Connelly, Conny, Conti, Conversion script, Coolblaze03, Copsewood, CortlandKlein, CosineKitty, CountZer0, Courcelles, Cp123127, Cpl Syx, Creative1980, Cshear, Cuahl, Cuaxdon, Cutter, Cwilso, Cwolfsheep, CyberSkull, Cyko 01, Cynical, Cyrius, DMG413, DRTllbrg, Dadu, Damian Yerrick, Daniel Olsen, DanielCD, Dattebayo321, Davewho2, David Levy, DavidCary, DeadEyeArrow, Deathmolor, Deeahbz, Deelkar, Dekimasu, Delirium, Desmarie17, Dfe6543, Diegocr, Dilumb, Dinesh smita, Diomidis Spinellis, Discospinster, Dk, DocWatson42, Donner60, Dougofborg, Drew R. Smith, Drkhataniar, Ducknish, Dustin gayler, Dysfunktion, E. Sn0 =31337=, ECELonghorn, Echoghost, Ed g2s, Edggar, Editerjhon, Edward, Eekoo, Ekilson, Elbperle, Elchupachipmunk, Eliezerb, Ellmist, Eltomzo, Emufarmers, Endlessnameless, Epbr123, Epicgenius, Eric119, Evansp, Evil Monkey, Evildeathmath, Ewawer, Ewlyahoocom, Exile.mind, F15 sanitizing eagle, FF2010, Falsifian, Fan-1967, Farshad83, FatalError, Feedloadr, Feierbach, Felix.rivas, Fennec, FenrirTheWolf, Flamingblur, Flyer22, Folajimi, Forage, FrYGuY, Fram, Fred Gandt, Frieda, Frosted14, Funandtrvl, Funky Monkey, Furrykef, G8yingri, Gabbe, Gabriel Kielland, Gabrielgmendonca, Gail, Gaius Cornelius, Garas, GargoyleMT, Gaurav1146, Gbeeker, Gen0cide, Gene Nygaard, Generic69, Geniac, Gerbrant, Ghacks, Giftlite, Gilliam, Gillyweed, Gimmetrow, Glenn, Gobonobo, Gogo Dodo, Gounis, Gracefool, GraemeL, GrayEagle1, Greg Lindahl, Grick, Ground Zero, Gwendal (usurped), H0riz0n, Haakon, Haham hanuka, HalfShadow, Hansix, Hardistyar, Harisankarh, Harjk, Harryboyles, Haseo9999, Heron, Hgfernan, Hobartimus, Hopper96, Hpfreak26, I dream of horses, I'mDown, Iainscott, Ice77cool, IceKarma, Icedog, Icseaturtles, Idleguy, Ifny, Ihsuss, Ilario, Imeriki al-Shimoni, InfinityB, Ingjerdj, Inkypaws, Iridescent, Irvine.david, Itai, Ivo, J grider65, JForget, JHunterJ, Jack Greenmaven, Jackfork, Jackiechen01, Jakester23jj, JamesBWatson, Jamesday, Jareha, Jeff G., Jeffrd10, Jennavecia, Jeremyb, Jersyko, Jiddisch, Jim McKeeth, Jimmytharpe, Jkielty82, Jludwig, Jmundo, Johann Wolfgang, Johnleemk, Johnteslade, Jon Backenstose, JonHarder, Jonik, Joshua Andersen, Josve05a, Joyous!, Jprg1966, Jrash, Jrdioko, Jreconomy, Jsnow, Julesd, Just zis Guy, you know?, Juststreamit, K.lee, K6ka, Karam.Anthony.K, Karrade, Kbdank71, Kbrose, KelleyCook, Kelly Martin, Kennyluck, Kevinzhouyan, Kevyn, Kgfleischmann, Kingturtle, Kku, Knownot, Knutux, Koenige, Koman90, Konstable, Korath, Korou, Kos1337tt, Kozuch, Kpjas, Kristen Eriksen, Kry, Kukini, Kumarbhatia, Kwertii, Kwirky88, Kzm, LBarsov, Laboriousme, Lairdp, Lajena, Larrybowler, Laudaka, Lawman3516, Layona1, LeaveSleaves, LedgendGamer, Lee1026, LeoNomis, Lesldock, Lesser Cartographies, Leuko, Lewu, Lexor, Liao, Lifefeed, Liftarn, Lightmouse, LilHelpa, LinuxPickle, Little Mountain 5, LittleDan, LjL, Lo2u, Loadmaster, Lockeownzj00, Lord JoNil, Lord dut, Loren36, LotharZ, Lousyd, Lradrama, Ludditesoft, Lugia2453, M, M 3bdelqader, MC707, MONGO, Mac, MadEwokHerd, Magioladitis, Mahanga, Mangojuice, ManiaQ, Manja Neuhaus, Manu3d, Materialscientist, MatrixFrog, Matthewedwards, Matthewleslie, Mattsag, Mattsenate, Mattycl, Mauro Bieg, Mavol, Maxamegalon2000, Maximaximax, Maximus Rex, Maxwell bernard, Mbauwens, Meembo, Melaen, Merope, Mesoderm, Miami33139, Miatatroll, Michael Hardy, Michaelbusch, MiguelTremblay, Mike Van Emmerik, Mikeblas, Mindmatrix, Minesweeper, Minghong, Mintleaf, Mirshafie, Miym, Mjohnson, Mkweise, Mlspeten, Mmv-ru, Mo0, Mobilon, Moehre, Moose mangle, Mormegil, Morten Blaabjerg, Morton.lin, MrJones, MrOllie, Mrdoomino, Msoos, Mthibault, Mupet0000, Mushroom, Musicat, Mydogategodshat, Neilapalmer, Neilc, Neiltheffernaniii, Nethgirb, Netkinetic, Netoholic, Nick, Nick Number, Ninly, Niteowlneils, Nixeagle, Nlu, NoIdeaNick, Nonnompow, Noodles-sb, Nshunter, Nuwewsco, Nyhet, OWV, Ocexyz, Octahedron80, Od Mishehu, Ohnoitsjamie, Old Death, Oldiowl, Olivier, Omegatron, Ooskapenaar, Owen, P2pauthor, P2prules, Palica, Paranoid, Patrickzuili, Pausch, Permacultura, Petchboo, Peter Tribe, PeterC, Peyre, Pgr94, Phatom87, Phelfe, PhilHibbs, Philip Trueman, Phoenix-forgotten, Pilotguy, PinchasC, Pinethicket, Ping, Pip2andahalf, Pjbflynn, Pmerson, Postdlf, Ppcoinwikipeercoin, Prab, Pratyeka, Preada, Preeeemo, Preston.lee, Profilemine, Qnext-Support, Quale, Qwm, RL0919, Radagast83, Radio Dan, Raraoul, RayBeckerman, Rchamberlain, Rcomrce, Rearden9, RedMurcury1, Reedy, Reinderien, Remember the dot, RentalicKim, Requestion, Retired username, Rettetast, RexNL, Rexguo, ReyBrujo, Rhobite, Rhododendrites, Rich Farmbrough, Richard Allen, RichardOSmith, Ricky@36, Ringbang, Rjdainty1, Rjstott, Rjwilmsi, Rl, Rocketrod1960, RogersMD, Ronz,

RoseParks, Rosejn, Rosesollere, Rossami, Rossgk, Royboycrashfan, Ruud Koot, Rvalles, RyanKoppelman, Ryulong, S.K., SEWilco, Sachdevj, Sae1962, Sahil sharma2119, Salsb, Sander123, SasiSasi, Satellizer, Schneelocke, SchreiberBike, Scoutersig, Scyth3, Seb26, Secfan, Senator2029, Sendai2ci, Shadowjams, Sharepro, ShaunMacPherson, Shellreef, Shielazhang, Shiftchange, Sietse Snel, Sigma 7, Silvarbullet1, SimonP, Sir Paul, SirJibby, Sirius Wallace, Sjakkalle, Skizzik, Skpande, Sky Attacker, SkyWalker, Sl, Slon02, SmileToday, Smywee, Snied, Soali, Softssa, Solitude, Someguy1221, Sonia, SpeedyGonsales, Splintax, Sschluter, Stanleyuroy, Stedjamulia, Steel, Stephan Leeds, Stephenchou0722, SteveSims, Stjones86, Strait, Sunil Mohan, Sweety Rose, TBloemink, TCL, TUF-KAT, Tad Lincoln, Tawker, Tazmaniacs, Teemuk, Teilolondon, Terribim, Teryx, Texture, Tgeairn, The Anome, The Belgain, The Thing That Should Not Be, The Wordsmith, The stuart, Thommi, Thumperward, Tide rolls, Tiger Brown, Tigrisnaga, Tigstep, Tim Ivorson, Timwi, TittoAssini, Tknew, Tohd8BohaithuGh1, Tom X. Tobin, Tom235, Tomchiukc, Tomdobb, Tony Sidaway, Tothwolf, Totof, Touisiau, TouristPhilosopher, Toussaint, Toytoy, Trek00, Tuxa, Tverbeek, Tyofcore, Ubiquity, Ultra-Loser, Una Smith, Unyoyega, V-apharmd, VBGFscJUn3, VCHunter, Vald, Valermos, VampWillow, Vanished user 31lk45mnzx90, Vanished user 8jq3ijalkdjhviewrie, VanishedUserABC, Vbs, Veinor, Velociped, Versageek, Viajero, Victorgrigas, Vincent stehle, Vironex, Vorash, Vrenator, W.phillips7, Waldir, Wangi, Warlordwolf, Wavelength, Wayne Slam, Wayward, Webpageone.co.uk, Wellreadone, Wesley, Widr, Wik, Wikidenizen, Will Beback Auto, Wiml, Wolfkeeper, Wrs1864, Wtfunkymonkey, Wuyanhuiyishi, Wvithanage, XDnonameXD, Xkoalax, Xpclient, Yacht, Yamakiri, Yandman, Yaronf, Yoelzanger, Youssefsan, Zak123321, Zarex, Zephyrxero, Zicko1, ZimZalaBim, Zso, Zzuuzz, Милан Јелисавчић, 1660 anonymous edits

Public key certificate *Source*: https://en.wikipedia.org/w/index.php?oldid=613240578 *Contributors*: 29Flavors, 32RB17, Acyclopedic, Aerowolf, Alexei Kojenov, Anon lynx, Armando, ArnoldReinhold, Aryaniae, BD2412, BKoteska, BWikime, BazookaJoe, Blutrot, Brentdax, Bryan Derksen, Chatfecter, ClementSeveillac, Codelux, Codename Lisa, Cody Cooper, Daniel.Cardenas, Dark Enigma, Davish Krail, Gold Five, Ddas, DeekGeek, Delta 51, Doberek, Downwards, Dpaking, Dvorak.typist, Ecemaml, Egret, Enix150, Eus Kevin, FlippyFlink, Flyer22, Folajimi, Fuzzy Logic, Gr8dude, Grawity, Gryszkalis, Guysha, H.Marxen, HDCase, Haakon, Haakon K, Haggis, Het, Hlkii, Hundred and half, Hut 8.5, Imeisel, Intgr, Isnow, Itahmed, Ixfd64, JTN, Javidjamae, Jdthood, Jesant13, JidGom, Jlascar, Jonathanbenari, Jordav, Julesd, Kenschry, Klausner, Knmorgan08, Kokamomi, Kpatelseo, Leobold1, LimoWreck, Linktolonkar, Loxlie, MER-C, Magioladitis, Matt Crypto, Mekirkpa, Meteor sandwich yum, Mfreidgeim, Michael Hardy, Mindmatrix, Mnemeson, Mohsens, Moscow Connection, Mr alibebe, Mydogategodshat, Nabieh, Nailed, Nixdorf, Notmicro, Nurg, Paulsheer, PhilipMW, Pinethicket, Pmetzger, Pxma, Q Chris, R.javanmard, RP459, Ra227, Rasmus Faber, Rcbarnes, Remember the dot, Rentzepopoulos, Robertbowerman, Runningboffin, SPS1962W, Schnolle, Shinton, Sinar, Smaines, Steven Weston, Strait, Sunnycomes, Sydius, TOMTOM1974, Tabletop, Teemuk, TonyW, Trammell, Ulrich Müller, Uppfinnarn, Wrs1864, Ww, Xanga, Yaronf, Zarutian, ZimZalaBim, 191 anonymous edits

Public-key cryptography *Source*: https://en.wikipedia.org/w/index.php?oldid=613067447 *Contributors*: 09sacjac, 1exec1, 28421u2232nfenfcenc, 8-Ball, Abdull, AbsolutDan, Adamd1008, Algebraist, Alpha Quadrant, Alphax, Althepal, Alvin-cs, Andre Engels, AndrewGarber, Anomie, Anon lynx, Anubhab91, Apokrif, Aravinthanjohn, Archer3, ArchiSchmedes, Armando, ArnoldReinhold, Arthur Rubin, Axd, AxelBoldt, Baselsm, Bender235, Bigbluefish, Birkett, Blouis79, Bongoramsey, Brec, Brianegge, Buzz-tardis, CBM, CRGreathouse, Calder, Cavan, Ccpearson, Cculpepper, Cedgin, Chenyu, Chester Markel, Chocolateboy, Chomwitt, Chris Capoccia, ChrisGualtieri, ChristopherThorpe, ChromiumCurium, ClementSeveillac, Cloudswrest, Cmswest, Conversion script, Corti, Craig t moore, CryptoDerk, Cryptonaut, DRLB, Daira Hopwood, Daniel.Cardenas, Darksentinel, Darth Sitges, David Oliver, DavidJablon, Davidgothberg, DeadEyeArrow, Delta G, Djordjes, Donner60, Donreed, Dougher, DouglasCalvert, Drphilharmonic, Dsspiegel, Dtwong, Duoduoduo, Duplico, E235, EasyReader010, Elf, Emansije, Erichsu, Etienne.navarro, Excirial, Fat64, Faustnh, FeatherPluma, Ferkel, Fg, Frap, Gabe19, Garde, Geoffreybernardo, Geremia, Ghimireabhinawa, Giftlite, Giles.reid, Gillespie09, GliderMaven, Gogo Dodo, Goncalopp, Graham87, Gtg204y, Half price, Harley peters, Hashbrowncipher, Heckjoe, Hmmwhatsthisdo, Huon, Hvs, Hymek, Ihope127, Incredibly Obese Black Man, InformatoSaurus, Intgr, Isilanes, Isnow, J.delanoy, Jarble, Jbening, Jcnorheim, Jd4v15, Jean-Baptiste Mestelan, Jed 20012, JeremyR, Jfdwolff, Jjron, Jnc, Jobarts, John lindgren, John85, Jokes Free4Me, Jomsr, Jon.weldon, Jonathanstray, Jredmond, Js coron, Jsampaio, Juliano, KHamsun, Kaiserb, Katalaveno, Kember, Khazar2, Kingpin13, Kiwi128, Kkddkk, Klaws, KohanX, Koveras, Kralizec!, Kratos 84, Kvamsi82, La goutte de pluie, LargeBlockCipher, Larry V, LedgendGamer, Lee J Haywood, LittleBenW, Loadmaster, Loren.wilton, Luqui, MPerel, Magioladitis, Magnificat, Manishsahni2000, Manishtripathi.uno, MarioS, Marj Tiefert, Markaci, Markskilbeck, Martarius, Marteau, Martinhellman, MastCell, Materialscientist, Matt Crypto, Mattsan22, Maxis ftw, Mboverload, Mdwyer, Mentisock, Metamorph123, Metaprimer, Meteor sandwich yum, Michael Hardy, Michael miceli, Microprocessor Man, Mikaey, Mike Rosoft, Mindloss, Mmull, Mo ainm, Moe Epsilon, Mr.Vansan, MrArt, MrOllie, Mrendo, Mîthrandir, Nageh, Necroticist, Ninly, Njan, Noodleki, Now3d, Nuobaite, Octahedron80, Olegos, Oliepedia, Onthefrynge, Osamanaseem, Ottomatic99, OutRIAAge, Oxymoron83, Pakaran, Paul August, Peaceray, PearlSt82, Peashy, Pepe123sdifjlk, Peter Hendrickson, Petr Kopač, Petrb, Phaldo, Phatom87, Philipp Weis, Phr, PierreAbbat, Pinethicket, Pinkadelica, Pne, Poodah, Prz, Psychotic Spoon, Qtiger, Quinn d, Quondum, Qwerty0, R Math, R'n'B, RMFan1, Radius, Raph79, Rasmus Faber, Robertgreer, Rossj81, Rprpr, Rror, Rsrikanth05, Runefrost, Ryulong, Sander123, Schlafly, Schoen, Scott McNay, Seipher, Sep34, ShaSheng, Shenoybipin, Siddhant, Simetrical, Skarebo, Skippydo, Skomorokh, Skulvis, Smite-Meister, StevieNic, Stigmj, Stormie, Strbenjr, Stux, Sv1xv, Tagesk, TastyPoutine, Tati-maranhao, Tbc, Tdowling, Teammm, Tech77, Techplex.Engineer, The Rambling Man, TheCoffee, Thefisker, Thegreatgrabber, Thesfisker, Thric3, Tiptoety, TooTallSid, Torla42, TreasuryTag, Tuankiet65, Tumbarumba, Turnstep, UFMace, UNOwenNYC, Ultimate.lost.cause, Uploadvirus, Urdutext, Vary, Vikreykja, Vogone, VoiceOfCalm, Vonzack, Wadlooper, Wavelength, Wesino, Wikidrone, Wilkie2000, Wireless friend, Wmagro, Wolfkeeper, Wshun, Wurtis65, Ww, Yaronf, Youandme, Yuxin19, Zodon, דניאל צבי, Ὁ οἴστρος, 541 anonymous edits

Route poisoning *Source*: https://en.wikipedia.org/w/index.php?oldid=595843446 *Contributors*: Althena, CSW1986, Circeus, Doctorfluffy, Lendorien, Lls71, Mindmatrix, MrRadioGuy, RedWolf, Roshjack47, Svick, The Wiki Octopus, Upholder, Vanished user kjdioejh329io3rksdkj, 23 anonymous edits

Sybil attack *Source*: https://en.wikipedia.org/w/index.php?oldid=579723288 *Contributors*: Brec, Chealer, Daira Hopwood, Dreamyshade, Frap, Gpierre, GregorB, HaeB, Hgkamath, Populus, Quuxplusone, Rotring, Scient, Snowolf, Terrycojones, Voomoo, 32 anonymous edits

Topological space *Source*: https://en.wikipedia.org/w/index.php?oldid=613452764 *Contributors*: Aaron Rotenberg, Acer4666, Algebraist, Amonk1962, Andrew Delong, AnnaFrance, Anonymous Dissident, Arcfrk, Arialblack, Arthur Rubin, Aude, AxelBoldt, BD2412, Bbi5291, Brad7777, Charles Matthews, Ciphers, Conversion script, Cpatra1984, Cstanford.math, DannyAsher, Danog, Dcoetzee, DefLog, Devourer09, Dineshjk, Dolda2000, Dratman, Dysprosia, Eaefremov, Egmontaz, Elwikipedista, Epicgenius, Failure.exe, FlordiaSunshine342, Fredrik, Freeze S, Fropuff, Gaurav Nirala, Gene Ward Smith, Giftlite, Graham87, Hairy Dude, Hirak 99, Improbable keeler, Iridescent, J.delanoy, J04n, JackSchmidt, Jakob.scholbach, Jeroen De Dauw, JerroldPease-Atlanta, Jesin, Jmajeremy, JoanneB, Jschnur, Karada, Kbk, Keenan Pepper, Kku, Konradek, Kupirijo, Kuratowski's Ghost, Kurt Artindagi, Lendu, Lethe, Liuthar, LkNsngth, LoganFromSA, Luqui, Maksim-e, Marc van Woerkom, Mark viking, MathMartin, Mattbr, Maurice Carbonaro, Mesoderm, MiNombreDeGuerra, Michael Hardy, MikeHaskel, Msh210, NawlinWiki, Nizmogtr, Odoncaoa, Oleg Alexandrov, Olivier, Orange Suede Sofa, P0lyglut, Palnot, Patrick, Paul August, Pharaoh of the Wizards, Phys, Plclark, Point-set topologist, Policron, Revolver, Rhobite, Rick Norwood, Roccorossi, RonnieBrown, Saaska, Salgueiro, Sardanaphalus, Sciyoshi, Seahorseruler, SgtThroat, Skakkle, Sligocki, Spambit, Stefan Udrea, Sunayana, Taxman, The Mudge, Tillmo, Tobias Bergemann, Toby Bartels, Tom.hirschowitz, Tompw, Tschwenn, Tsirel, Varuna, Vriullop, Wavelength, Wdevauld, Wikfr, Woohookitty, Wshun, XJaM, Xantharius, YK Times, Zhaoway, Zundark, 166 anonymous edits

Web of trust *Source*: https://en.wikipedia.org/w/index.php?oldid=614694085 *Contributors*: Abscissa, Alphax, ArnoldReinhold, Arvindn, Astronautics, Bblfish, Beland, Bonadea, Btyner, Buxh, Chirags, Christopherlin, Compfreak7, Connelly, D3M1TR13, DVdm, Damian Yerrick, David.Monniaux, Davidgothberg, Dfgriggs, Dionyziz, Dyork, EJNorman, Edgerck, Etienne.navarro, Froid, George8211, Hadal, Hairy Dude, Henkp, Hiihammuk, Hu12, Imran, Intgr, Isaac Dupree, Jad jay2, Janvlug, Jeffq, Jel, Jeremy Visser, Josang, Joy, JzG, KenWalker, KristinKu, LFaraone, Lance W. Haverkamp, LoganLopez, Lunkwill, Madrigal12, Mandarax, Marco Krohn, Matt Crypto, MaxamusCrasious, Michael Hardy, MikeDaigle, Nageh, Namescases, Nealmcb, Paddu, PeterJeremy, PhilipMW, Pnm, Prz, Quinwound, Radicle, Reinyday, Requestion, Reskusic, Rich Farmbrough, Scientus, Scootzer99, Scott, Starnestommy, Startcom, SusanLesch, TastyPoutine, Thv, Tobias Bergemann, TwN, Waveguide1360, WikiPediaAid, Wsoder, Ww, Xanzzibar, Zalgo, أحمد, 78 anonymous edits

X.509 *Source*: https://en.wikipedia.org/w/index.php?oldid=614895097 *Contributors*: Abdull, Aerowolf, Alan J Shea, Alan U. Kennington, Alberge204, AlistairMcMillan, Altermike, Andreiko, Andy Dingley, Angeltoribio, Angusmca, Armando, Atkinson 291, Azollman, Baccala@freesoft.org, Barcex, Bender235, Billinghurst, Bomazi, Buttysquirrel, C960657, Camilo Sanchez, Captain panda, Cjcollier, Cnilep, Commander, Corti, Craiged, Cryptoki, D.w.chadwick, Daniel.Cardenas, Darac, Darth Sitges, David Woolley, DeTreville, Digi-cs, Dkgdkg, DokReggar, Drondent, Dysprosia, Dzlabing, E2eamon, EagleOne, Eidab, Eus Kevin, EvgeniGolov, Eyreland, Feezo, Fenring, Foxj, Frap, Glenn, Gorgonzilla, Grawity, Grendelkhan, Gunn1, Gurch, Hoylen, Hymek, Iida-yosiaki, Imran, Inkling, JTN, JWWalker, Jason Quinn, Jasuus, Jeff J. Snider, JidGom, John Vandenberg, JonHarder, Jonash, Judesba, Jwilkinson, KAtremer, Kdz, Kenschry, Kku, Klamation, Klausner, Kusma, Kw27, Libraequilibra, LightStarch, LilHelpa, Magioladitis, MarkWahl, Matt Crypto, Mesoderm, Meteor sandwich yum, Michael Hardy, Michel SALES, Minghong, MoraSique, Mortense, MrOllie, Mrbakerfield, Msct, Narayan82es, Nealmcb, Neustradamus, Nick Number, Noloader, Nono64, Nopherox, NuclearWizard, Nuwewsco, Orborde, OverlordQ, Pastore Italy, Paul Foxworthy, Pdejuan, Peaceray, PerryTachett, Pgan002, Q Chris, Racaille, Ram Moskovitz, Rasmus Faber, Rhoerbe, Rich Farmbrough, Robert Illes, Rochus, Rohanuk, Rsrikanth05, Samuel J. Howard, Saqib, Schnolle, Scorpiuss, Sega381, Sid.shetye, Sievebrain, Siryendor, SkyRocketMedia, Sligocki, Smyth, Stevag, Stewartadcock, The Thing That Should Not Be, Thierrytung, Tnshibu, Toshiro92, Tuntable, Unixplumber, Vandevenp, Vegarwe, Vspaceg, Wgd, Whmac, Whophd, Wiarthurhu, Widefox, Wik, Wrev, Wrs1864, Ww, Xandi, YUL89YYZ, Zeptomoon, Zeroshell, Zundark, 191 anonymous edits

Image Sources, Licenses and Contributors

File:Bitcoin logo.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Bitcoin_logo.svg *License*: Creative Commons Zero *Contributors*: Bitboy

File:BitcoinSign.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:BitcoinSign.svg *License*: Public Domain *Contributors*: Satoshi Nakamoto

File:USB Erupter.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:USB_Erupter.jpg *License*: Creative Commons Zero *Contributors*: User:FML

file:Bitcoin-coins.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Bitcoin-coins.jpg *License*: Public Domain *Contributors*: User:Casascius

file:Bitcoin paper wallet generated at bitaddress.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Bitcoin_paper_wallet_generated_at_bitaddress.jpg *License*: unknown *Contributors*: Open Source

File:Electrum Bitcoin Wallet.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Electrum_Bitcoin_Wallet.png *License*: GNU General Public License *Contributors*: Chrisarnesen, Hopkinsenior, HowardStrong, Mikepanhu, Rezonansowy, Torsch, Yann, 1 anonymous edits

File:De Waag Bitcoin.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:De_Waag_Bitcoin.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Targaryen

File:CC-BY-SA-icon-80x15.png *Source*: https://en.wikipedia.org/w/index.php?title=File:CC-BY-SA-icon-80x15.png *License*: Creative Commons Attribution 3.0 *Contributors*: Creative Commons

File:Bitcoin Transaction Visual.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Bitcoin_Transaction_Visual.svg *License*: Creative Commons Zero *Contributors*: Graingert

File:Capture-Electrum.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Capture-Electrum.png *License*: Creative Commons Attribution 3.0 *Contributors*: Joise, Torsch

File:Sample Bitcoin paper wallet.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Sample_Bitcoin_paper_wallet.png *License*: GNU Free Documentation License *Contributors*: Canton

File:blockchain.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Blockchain.svg *License*: Creative Commons Attribution 3.0 *Contributors*: original file: Theymos from Bitcoin wiki vectorization:

File:AustrianBitCoinMiningRig.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:AustrianBitCoinMiningRig.jpg *License*: Creative Commons Attribution 2.0 *Contributors*: FlickreviewR, Kencf0618, Targaryen, Tuankiet65

File:Lancelot-A FPGA based bitcoin mining board.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Lancelot-A_FPGA_based_bitcoin_mining_board.jpg *License*: Creative Commons Zero *Contributors*: Targaryen, Xiangfu

File:Avalon-An ASIC base bitcoin machine.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Avalon-An_ASIC_base_bitcoin_machine.jpg *License*: Creative Commons Zero *Contributors*: 张楠赓

File:ASICMINER USB Block Erupter.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:ASICMINER_USB_Block_Erupter.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Targaryen

File:Butterfly_Labs_60GH_Bitcoin_Miner_Single_SC.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Butterfly_Labs_60GH_Bitcoin_Miner_Single_SC.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Joshua Zerlan

File:Bitcoinpaymentverification.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Bitcoinpaymentverification.png *License*: unknown *Contributors*: Satoshi Nakamoto

File:Chord project.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Chord_project.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:MithrandirMage

Image:chord network.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Chord_network.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Seth Terashima (Tetra7 (talk))

Image:chord route.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Chord_route.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Seth Terashima (Tetra7 (talk))

File:Fibreoptic.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Fibreoptic.jpg *License*: GNU Free Documentation License *Contributors*: BigRiz

File:Wireless network.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Wireless_network.jpg *License*: GNU Free Documentation License *Contributors*: Adamantios, FireFlower, MB-one, Porao, Thuresson, Tyler178, WikipediaMaster, 9 anonymous edits

File:ForeRunnerLE 25 ATM Network Interface (1).jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:ForeRunnerLE_25_ATM_Network_Interface_(1).jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Barcex

File:Adsl connections.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Adsl_connections.jpg *License*: GNU Free Documentation License *Contributors*: Asim18

File:NetworkTopologies.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:NetworkTopologies.svg *License*: Public Domain *Contributors*: NetworkTopologies.png: Maksim derivative work: Malyszkz (talk)

File:Network Overlay merged.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Network_Overlay_merged.svg *License*: Creative Commons Attribution-Share Alike *Contributors*: Ludovic.ferre

File:Internet layering.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Internet_layering.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jsoon eu (talk)

File:Internet map 1024.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Internet_map_1024.jpg *License*: Creative Commons Attribution 2.5 *Contributors*: Barrett Lyon The Opte Project

File:XO classroom network.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:XO_classroom_network.jpg *License*: Creative Commons Attribution 2.5 *Contributors*: OLPC

File:PD-icon.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:PD-icon.svg *License*: Public Domain *Contributors*: Alex.muller, Anomie, Anonymous Dissident, CBM, MBisanz, PBS, Quadell, Rocket000, Strangerer, Timotheus Canens, 1 anonymous edits

Image:Cryptographic Hash Function.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Cryptographic_Hash_Function.svg *License*: Public Domain *Contributors*: User:Jorge Stolfi based on Image:Hash_function.svg by Helix84

Image:Merkle-Damgard hash big.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Merkle-Damgard_hash_big.svg *License*: Public Domain *Contributors*: Davidgothberg

Image:Nonce-cnonce-uml.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Nonce-cnonce-uml.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: en:User:Cameltrader

File:Symmetric-key cryptography.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Symmetric-key_cryptography.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Denis Fadeev

Image:Lorenz-SZ42-2.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Lorenz-SZ42-2.jpg *License*: Public domain *Contributors*: Avron, Dbenbenn, Sissssou

Image:Skytala&EmptyStrip-Shaded.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Skytala&EmptyStrip-Shaded.png *License*: GNU Free Documentation License *Contributors*: User:Chrkl

File:16th century French cypher machine in the shape of a book with arms of Henri II.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:16th_century_French_cypher_machine_in_the_shape_of_a_book_with_arms_of_Henri_II.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Uploadalt

File:Encoded letter of Gabriel Luetz d Aramon after 1546 with partial deciphering.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Encoded_letter_of_Gabriel_Luetz_d_Aramon_after_1546_with_partial_deciphering.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Uploadalt

File:Symmetric key encryption.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Symmetric_key_encryption.svg *License*: Creative Commons Zero *Contributors*: User:Phayzfaustyn

Image:International Data Encryption Algorithm InfoBox Diagram.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:International_Data_Encryption_Algorithm_InfoBox_Diagram.svg *License*: Public Domain *Contributors*: Original uploader was Surachit at en.wikipedia

File:Public key encryption.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Public_key_encryption.svg *License*: Public Domain *Contributors*: Athalis, Davidgothberg, 5 anonymous edits

Image:Diffie and Hellman.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Diffie_and_Hellman.jpg *License*: GNU Free Documentation License *Contributors*: ArnoldReinhold, MichaelMaggs, Mu, Phr, William Avery, Ö

Image:Firefox-SSL-padlock.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Firefox-SSL-padlock.png *License*: unknown *Contributors*: Mozilla contributors

Image:Enigma.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Enigma.jpg *License*: Public Domain *Contributors*: user:Jszigetvari

Image:2008-09 Kaiserschloss Kryptologen.JPG *Source*: https://en.wikipedia.org/w/index.php?title=File:2008-09_Kaiserschloss_Kryptologen.JPG *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Ziko

file:Wiktionary-logo-en.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Wiktionary-logo-en.svg *License*: Public Domain *Contributors*: Vectorized by , based on original logo tossed together by Brion Vibber

file:Commons-logo.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Commons-logo.svg *License*: logo *Contributors*: Anomie

File:Alexis de Tocqueville.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Alexis_de_Tocqueville.jpg *License*: Public Domain *Contributors*: MarmadukePercy

File:Decentralization.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Decentralization.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Adam Aladdin

File:East Province districts.png *Source*: https://en.wikipedia.org/w/index.php?title=File:East_Province_districts.png *License*: Public Domain *Contributors*: C41n, Gryffindor, Santosga

File:Portrait of Pierre Joseph Proudhon 1865.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Portrait_of_Pierre_Joseph_Proudhon_1865.jpg *License*: Public Domain *Contributors*: Ecummenic, Thomas Gun, Trzęsacz

File:Living Machine at Port of Portland-interior.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:Living_Machine_at_Port_of_Portland-interior.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Living Machine Systems, L3C

Image:Digital Signature diagram.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Digital_Signature_diagram.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Acdx

File:DHT en.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:DHT_en.svg *License*: Public Domain *Contributors*: Jnlin

File:6n-graf.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:6n-graf.svg *License*: Public Domain *Contributors*: User:AzaToth

file:Directed.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Directed.svg *License*: Public Domain *Contributors*: Grafite, Jcb, Josette, 2 anonymous edits

file:Undirected.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Undirected.svg *License*: Public Domain *Contributors*: JMCC1, Josette, Kilom691, 2 anonymous edits

File:Complete graph K5.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Complete_graph_K5.svg *License*: Public Domain *Contributors*: David Benbennick wrote this file.

File:Hash table 3 1 1 0 1 0 0 SP.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hash_table_3_1_1_0_1_0_0_SP.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jorge Stolfi

File:Hash table 5 0 1 1 1 1 1 LL.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hash_table_5_0_1_1_1_1_1_LL.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jorge Stolfi

File:Hash table 5 0 1 1 1 1 0 LL.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hash_table_5_0_1_1_1_1_0_LL.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jorge Stolfi

File:Hash table 5 0 1 1 1 1 0 SP.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hash_table_5_0_1_1_1_1_0_SP.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jorge Stolfi

File:Hash table average insertion time.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Hash_table_average_insertion_time.png *License*: Public Domain *Contributors*: User:Dcoetzee

File:Hexahedron.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hexahedron.svg *License*: GNU Free Documentation License *Contributors*: User:DTR

File:Hypercube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hypercube.svg *License*: GNU Free Documentation License *Contributors*: A.Savin, Goffrie, Herbythyme, Itai, Kilom691, Mate2code, Mouagip, Mysid, Opensofias, 7 anonymous edits

Image:Dimension levels.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Dimension_levels.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: NerdBoy1392

File:From Point to Tesseract (Looped Version).gif *Source*: https://en.wikipedia.org/w/index.php?title=File:From_Point_to_Tesseract_(Looped_Version).gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Tmp6154

File:CDel_node_1.png *Source*: https://en.wikipedia.org/w/index.php?title=File:CDel_node_1.png *License*: Public Domain *Contributors*: User:Tomruen

File:CDel_4.png *Source*: https://en.wikipedia.org/w/index.php?title=File:CDel_4.png *License*: Public Domain *Contributors*: User:Tomruen

File:CDel_node.png *Source*: https://en.wikipedia.org/w/index.php?title=File:CDel_node.png *License*: Public Domain *Contributors*: User:Tomruen

File:CDel_3.png *Source*: https://en.wikipedia.org/w/index.php?title=File:CDel_3.png *License*: Public Domain *Contributors*: User:Tomruen

File:1-simplex t0.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:1-simplex_t0.svg *License*: Public Domain *Contributors*: Tomruen

File:2-cube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:2-cube.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Tomruen

File:3-cube graph.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:3-cube_graph.svg *License*: Public Domain *Contributors*: Opensofias

File:4-cube graph.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:4-cube_graph.svg *License*: Public Domain *Contributors*: JMCC1, Mate2code, Opensofias

File:5-cube graph.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:5-cube_graph.svg *License*: Public Domain *Contributors*: JMCC1, Opensofias

File:6-cube graph.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:6-cube_graph.svg *License*: Public Domain *Contributors*: JMCC1, Opensofias

File:7-cube graph.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:7-cube_graph.svg *License*: Public Domain *Contributors*: JMCC1, Opensofias

File:8-cube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:8-cube.svg *License*: Public Domain *Contributors*: Athaenara, Incnis Mrsi, JMCC1, Jgmoxness, Opensofias, Tomruen

File:9-cube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:9-cube.svg *License*: Public Domain *Contributors*: Tom Ruen (talk).

File:10-cube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:10-cube.svg *License*: Public Domain *Contributors*: Tom Ruen (talk). Original uploader was Tomruen at en.wikipedia

File:11-cube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:11-cube.svg *License*: Public Domain *Contributors*: Tomruen

File:12-cube.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:12-cube.svg *License*: Public Domain *Contributors*: Tomruen

File:8-cell.gif *Source*: https://en.wikipedia.org/w/index.php?title=File:8-cell.gif *License*: Public Domain *Contributors*: see above

Image:Hypercubestar.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hypercubestar.svg *License*: Public Domain *Contributors*: CommonsDelinker, Incnis Mrsi, Itai, Mate2code, 4 anonymous edits

File:Hypercubeconstruction.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Hypercubeconstruction.png *License*: Public Domain *Contributors*: Original uploader was Stdazi at en.wikipedia

File:Dht example SVG.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Dht_example_SVG.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Limaner

Image:Hash Tree.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Hash_Tree.svg *License*: Creative Commons Zero *Contributors*: User:Azaghal

Image:P2P-network.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:P2P-network.svg *License*: Public Domain *Contributors*: User:Mauro Bieg

Image:Server-based-network.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Server-based-network.svg *License*: GNU Lesser General Public License *Contributors*: User:Mauro Bieg

File:AMule 2.3.1 screenshot (active downloads).png *Source*: https://en.wikipedia.org/w/index.php?title=File:AMule_2.3.1_screenshot_(active_downloads).png *License*: GNU Free Documentation License *Contributors*: Mesoderm

File:Unstructured peer-to-peer network diagram.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Unstructured_peer-to-peer_network_diagram.png *License*: Creative Commons Zero *Contributors*: Mesoderm

File:Structured (DHT) peer-to-peer network diagram.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Structured_(DHT)_peer-to-peer_network_diagram.png *License*: Creative Commons Zero *Contributors*: Mesoderm

File:Yacy-resultados.png *Source*: https://en.wikipedia.org/w/index.php?title=File:Yacy-resultados.png *License*: unknown *Contributors*: User:Hack-Master

File:Torrentcomp small.gif *Source*: https://en.wikipedia.org/w/index.php?title=File:Torrentcomp_small.gif *License*: GNU Free Documentation License *Contributors*: Frigotoni, Holek, Joolz, Mdd, Senator2029, Sputnik, Wikiadd, Wknight94, Ypacaraí, 3 anonymous edits

File:Usage-of-Digital-Certificate.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Usage-of-Digital-Certificate.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Eus Kevin (talk)

File:Public-key-crypto-1.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Public-key-crypto-1.svg *License*: Public domain *Contributors*: KohanX (talk)

Image:Public key shared secret.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Public_key_shared_secret.svg *License*: Public Domain *Contributors*: Davidgothberg

Image:Topological space examples.svg *Source*: https://en.wikipedia.org/w/index.php?title=File:Topological_space_examples.svg *License*: Public Domain *Contributors*: Dcoetzee

File:MSD-Based Trust.png *Source*: https://en.wikipedia.org/w/index.php?title=File:MSD-Based_Trust.png *License*: Creative Commons Zero *Contributors*: wsoder

License

Creative Commons Attribution-Share Alike 3.0
//creativecommons.org/licenses/by-sa/3.0/