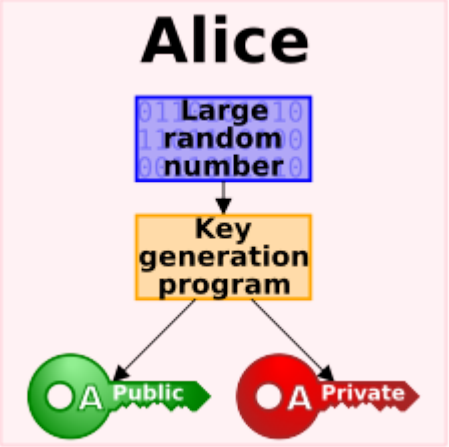
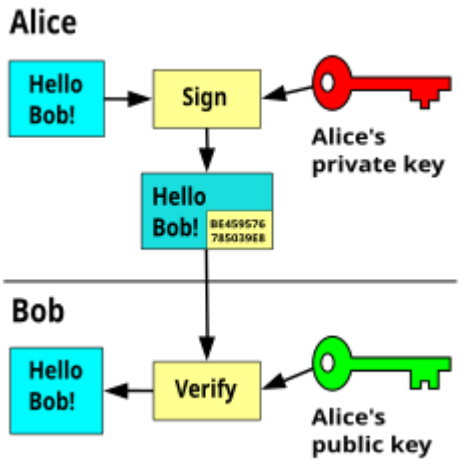
Public-key cryptography

By Contributors to Wikimedia projects

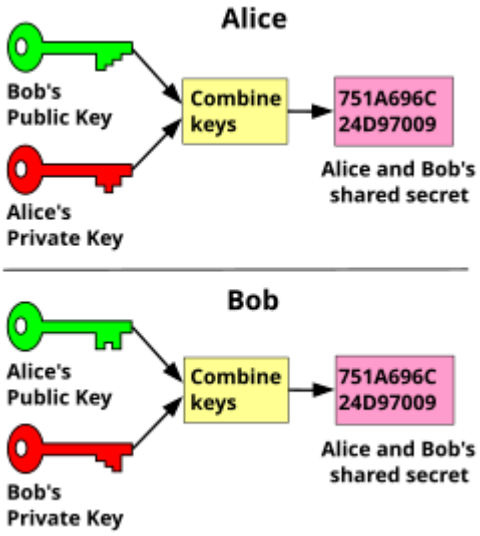
Published: 2001-11-09 · Archived: 2026-04-06 00:38:15 UTC



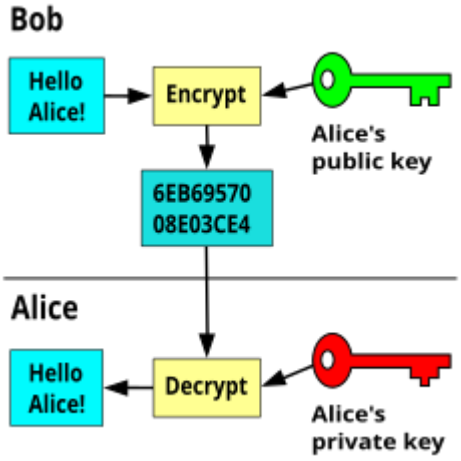
An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.



In this example the message is digitally signed with Alice's private key, but the message itself is not encrypted. 1) Alice signs a message with her private key. 2) Using Alice's public key, Bob can verify that Alice sent the message and that the message has not been modified.



In the Diffie–Hellman key exchange scheme, each party generates a public/private key pair and distributes the public key of the pair. After obtaining an authentic (n.b., this is critical) 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.



In an asymmetric key encryption scheme, anyone can encrypt messages using a public key, but only the holder of the paired private key can decrypt such a message. The security of the system depends on the secrecy of the private key, which must not become known to any other.

Public-key cryptography, or **asymmetric cryptography**, is the field of cryptographic systems that use pairs of related keys. Each key pair consists of a **public key** and a corresponding **private key**.[1][2] Key pairs are generated with algorithms based on mathematical problems termed one-way functions. Security of public-key cryptography depends on keeping the private key secret; the public key can be openly distributed without compromising security.[3] There are many kinds of public-key cryptosystems, with different security goals, including digital signature, Diffie–Hellman key exchange, public-key key encapsulation, and public-key encryption.

Public key algorithms are fundamental security primitives in modern cryptosystems, including applications and protocols that offer assurance of the confidentiality and authenticity of electronic communications and data

storage. They underpin numerous Internet standards, such as Transport Layer Security (TLS), SSH, S/MIME, and PGP. Compared to symmetric cryptography, public-key cryptography can be too slow for many purposes,[4] so these protocols often combine symmetric cryptography with public-key cryptography in hybrid cryptosystems.

Before the mid-1970s, all cipher systems used symmetric key algorithms, in which the same cryptographic key is used with the underlying algorithm by both the sender and the recipient, who must both keep the key secret. Of necessity, the key in every such system had to be exchanged between the communicating parties in some secure way prior to any use of the system – for instance, via a secure channel. This requirement is never trivial and very rapidly becomes unmanageable as the number of participants increases, when secure channels are not available, or when (as is sensible cryptographic practice) keys are frequently changed. In particular, if messages are meant to be secure from other users, a separate key is required for each possible pair of users.

By contrast, in a public-key cryptosystem, the public keys can be disseminated widely and openly, and only the corresponding private keys need be kept secret.

The two best-known types of public key cryptography are digital signature and public-key encryption:

- In a **digital signature** system, a sender can use a private key together with a message to create a *signature*. Anyone with the corresponding public key can verify whether the signature matches the message, but a forger who does not know the private key cannot find any message/signature pair that will pass verification with the public key.[5][6][7]

  For example, a software publisher can create a signature key pair and include the public key in software installed on computers. Later, the publisher can distribute an update to the software signed using the private key, and any computer receiving an update can confirm it is genuine by verifying the signature using the public key. As long as the software publisher keeps the private key secret, even if a forger can distribute malicious updates to computers, they cannot convince the computers that any malicious updates are genuine.

- In a **public-key encryption** system, anyone with a public key can encrypt a message, yielding a *ciphertext*, but only those who know the corresponding private key can decrypt the ciphertext to obtain the original message.[8]

  For example, a journalist can publish the public key of an encryption key pair on a web site so that sources can send secret messages to the news organization in ciphertext.

  Only the journalist who knows the corresponding private key can decrypt the ciphertexts to obtain the sources' messages—an eavesdropper reading email on its way to the journalist cannot decrypt the ciphertexts. However, public-key encryption does not conceal metadata like what computer a source used to send a message, when they sent it, or how long it is.[9][10][11][12] Public-key encryption on its own also does not tell the recipient anything about who sent a message[8]:283[13][14]—it just conceals the content of the message.

Applications built on public-key cryptography include authenticating web servers with TLS, digital cash, password-authenticated key agreement, authenticating and concealing email content with OpenPGP or S/MIME,

and time-stamping services and non-repudiation protocols.

One important issue is confidence/proof that a particular public key is authentic, i.e. that it is correct and belongs to the person or entity claimed, and has not been tampered with or replaced by some (perhaps malicious) third party. There are several possible approaches, including:

A public key infrastructure (PKI), in which one or more third parties – known as certificate authorities – certify ownership of key pairs. TLS relies upon this. This implies that the PKI system (software, hardware, and management) is trust-able by all involved.

A "web of trust" decentralizes authentication by using individual endorsements of links between a user and the public key belonging to that user. PGP uses this approach, in addition to lookup in the domain name system (DNS). The DKIM system for digitally signing emails also uses this approach.

Hybrid cryptosystems

[edit]

Because asymmetric key algorithms are nearly always much more computationally intensive than symmetric ones, it is common to use a public/private *asymmetric* key-exchange algorithm to encrypt and exchange a *symmetric key*, which is then used by symmetric-key cryptography to transmit data using the now-shared *symmetric key* for a symmetric key encryption algorithm. PGP, SSH, and the SSL/TLS family of schemes use this procedure; they are thus called *hybrid cryptosystems*. The initial *asymmetric* cryptography-based key exchange to share a server-generated *symmetric* key from the server to client has the advantage of not requiring that a symmetric key be pre-shared manually, such as on printed paper or discs transported by a courier, while providing the higher data throughput of symmetric key cryptography over asymmetric key cryptography for the remainder of the shared connection.

As with all security-related systems, there are various potential weaknesses in public-key cryptography. Aside from poor choice of an asymmetric key algorithm (there are few that are widely regarded as satisfactory) or too short a key length, the chief security risk is that the private key of a pair becomes known. All security of messages, authentication, etc., encrypted with this private key will then be lost. This is commonly mitigated (such as in recent TLS schemes) by using Forward secrecy capable schemes that generate an ephemeral set of keys during the communication which must also be known for the communication to be compromised.

Additionally, with the advent of quantum computing, many asymmetric key algorithms are considered vulnerable to attacks, and new quantum-resistant schemes are being developed to overcome the problem.[15][16]

Beyond algorithmic or key-length weaknesses, some studies have noted risks when private key control is delegated to third parties. Research on Uruguay's implementation of Public Key Infrastructure under Law 18.600 found that centralized key custody by Trust Service Providers (TSPs) may weaken the principle of private-key secrecy, increasing exposure to man-in-the-middle attacks and raising concerns about legal non-repudiation.[17]

All public key schemes are in theory susceptible to a "brute-force key search attack".[18] However, such an attack is 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 inherently have much lower work factors, making resistance to a brute-force attack (e.g., from longer keys) 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.[*citation needed*] None of these are sufficiently improved to be actually practical, however.

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.[19] As with all cryptographic functions, public-key implementations may be vulnerable to side-channel attacks that exploit information leakage to simplify the search for a secret key. These are often independent of the algorithm being used. Research is underway to both discover, and to protect against, new attacks.

Alteration of public keys

[edit]

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, in all instances, be intercepted, decrypted, and re-encrypted by the attacker using the correct public keys for the different communication segments so as to avoid suspicion.[*citation needed*]

A communication is said to be insecure where data is transmitted in a manner that allows for interception (also called "sniffing"). These terms refer to reading the sender's private data in its entirety. A communication is particularly unsafe when interceptions can not be prevented or monitored by the sender.[20]

A man-in-the-middle attack can be difficult to implement due to the complexities of modern security protocols. However, the task becomes simpler when a sender is using insecure media such as public networks, the Internet, or wireless communication. In these cases an attacker can compromise the communications infrastructure rather than the data itself. A hypothetical malicious staff member at an Internet service provider (ISP) might find a man-in-the-middle attack relatively straightforward. Capturing the public key would only require searching for the key as it gets sent through the ISP's communications hardware; in properly implemented asymmetric key schemes, this is not a significant risk.[*citation needed*]

In some advanced man-in-the-middle attacks, one side of the communication will see the original data while the other will receive a malicious variant. Asymmetric man-in-the-middle attacks can prevent users from realizing their connection is compromised. This remains so even when one user's data is known to be compromised because the data appears fine to the other user. This can lead to confusing disagreements between users such as "it must be on your end!" when neither user is at fault. Hence, man-in-the-middle attacks are only fully preventable when the communications infrastructure is physically controlled by one or both parties; such as via a wired route inside the sender's own building. In summation, public keys are easier to alter when the communications hardware used by a sender is controlled by an attacker.[21][22][23]

Public key infrastructure

[edit]

One approach to prevent such attacks involves the use of a public key infrastructure (PKI); a set of roles, policies, and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption. However, this has potential weaknesses.

For example, the certificate authority issuing the certificate must be trusted by all participating parties to have properly checked the identity of the key-holder, to have ensured the correctness of the public key when it issues a certificate, to be secure from computer piracy, and to 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 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. An attacker who penetrates an authority's servers and obtains its store of certificates and keys (public and private) would be able to spoof, masquerade, decrypt, and forge transactions without limit, assuming that they were able to place themselves in the communication stream.

Despite its theoretical and potential problems, Public key infrastructure is widely used. Examples include TLS and its predecessor SSL, which are commonly used to provide security for web browser transactions (for example, most websites utilize TLS for HTTPS).

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.

Unencrypted metadata

[edit]

Most of the available public-key encryption software does not conceal metadata in the message header, which might include the identities of the sender and recipient, the sending date, subject field, and the software they use etc. Rather, only the body of the message is concealed and can only be decrypted with the private key of the intended recipient. This means that a third party could construct quite a detailed model of participants in a communication network, along with the subjects being discussed, even if the message body itself is hidden.

However, there has been a recent demonstration of messaging with encrypted headers, which obscures the identities of the sender and recipient, and significantly reduces the available metadata to a third party.[24] The concept is based around an open repository containing separately encrypted metadata blocks and encrypted

messages. Only the intended recipient is able to decrypt the metadata block, and having done so they can identify and download their messages and decrypt them. Such a messaging system is at present in an experimental phase and not yet deployed. Scaling this method would reveal to the third party only the inbox server being used by the recipient and the timestamp of sending and receiving. The server could be shared by thousands of users, making social network modelling much more challenging.

During the early history of cryptography, two parties would rely upon a key that they would exchange by means of a secure, but non-cryptographic, method such as a face-to-face meeting, or a trusted courier. This key, which both parties must then keep absolutely secret, could then be used to exchange encrypted messages. A number of significant practical difficulties arise with this approach to distributing keys.

In his 1874 book *The Principles of Science*, William Stanley Jevons wrote:[25]

> Can the reader say what two numbers multiplied together will produce the number 8,616,460,799?[26] I think it unlikely that anyone but myself will ever know.[25]

Here he described the relationship of one-way functions to cryptography, and went on to discuss specifically the factorization problem used to create a trapdoor function. In July 1996, mathematician Solomon W. Golomb said: "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."[27]

Classified discovery

[edit]

In 1970, James H. Ellis, a British cryptographer at the UK Government Communications Headquarters (GCHQ), conceived of the possibility of "non-secret encryption", (now called public key cryptography), but could see no way to implement it.[28][29][30]

In 1973, his colleague Clifford Cocks implemented what has become known as the RSA encryption algorithm, giving a practical method of "non-secret encryption", and in 1974 another GCHQ mathematician and cryptographer, Malcolm J. Williamson, developed what is now known as Diffie–Hellman key exchange. The scheme was also passed to the US's National Security Agency.[31] Both organisations had a military focus and only limited computing power was available in any case; the potential of public key cryptography remained unrealised by either organization. According to Ralph Benjamin:

> I judged it most important for military use ... if you can share your key rapidly and electronically, you have a major advantage over your opponent. Only at the end of the evolution from Berners-Lee designing an open internet architecture for CERN, its adaptation and adoption for the Arpanet ... did public key cryptography realise its full potential.[31]

These discoveries were not publicly acknowledged until the research was declassified by the British government in 1997.[32]

In 1976, an asymmetric key cryptosystem was published 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.[33] This was the first published practical method for establishing a shared secret-key over an authenticated (but not confidential) 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 only published in 1978. This makes asymmetric encryption a rather new field in cryptography although cryptography itself dates back more than 2,000 years.[34]

In 1977, a generalization of Cocks's scheme was independently invented by Ron Rivest, Adi Shamir and Leonard Adleman, all then at MIT. The latter authors published their work in 1978 in Martin Gardner's Scientific American column, and the algorithm came to be known as RSA, from their initials.[35] RSA uses exponentiation modulo a product of two very large primes, to encrypt and decrypt, performing both public key encryption and public key digital signatures. Its security is connected to the extreme difficulty of factoring large integers, a problem for which there is no known efficient general technique. A description of the algorithm was published in the Mathematical Games column in the August 1977 issue of Scientific American.[36]

Since the 1970s, a large number and variety of encryption, digital signature, key agreement, and other techniques have been developed, including the Rabin signature, ElGamal encryption, DSA and ECC.

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
- Elliptic-curve cryptography
    - Elliptic Curve Digital Signature Algorithm (ECDSA)
    - Elliptic-curve Diffie–Hellman (ECDH)
    - Ed25519 and Ed448 (EdDSA)
    - X25519 and X448 (ECDH/EdDH)
- 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 yet widely adopted include:

- NTRUEncrypt cryptosystem
- Kyber
- McEliece cryptosystem

Examples of notable – yet insecure – asymmetric key algorithms include:

- Merkle–Hellman knapsack cryptosystem

Examples of protocols using asymmetric key algorithms include:

- S/MIME
- GPG, an implementation of OpenPGP, and an Internet Standard
- EMV, EMV Certificate Authority
- IPsec
- PGP
- ZRTP, a secure VoIP protocol
- Transport Layer Security standardized by IETF and its predecessor Secure Socket Layer
- SILC
- SSH
- Bitcoin
- Off-the-Record Messaging

- Books on cryptography
- GNU Privacy Guard
- Identity-based encryption (IBE)
- Key escrow
- Key-agreement protocol
- PGP word list
- Post-quantum cryptography
- Pretty Good Privacy
- Pseudonym
- Public key fingerprint
- Public key infrastructure (PKI)
- Quantum computing
- Quantum cryptography
- Secure Shell (SSH)
- Symmetric-key algorithm
- Threshold cryptosystem
- Web of trust

1. ^ *R. Shirey (August 2007). Internet Security Glossary, Version 2. Network Working Group. doi:10.17487/RFC4949. RFC 4949. Informational.*
2. ^ *Bernstein, Daniel J.; Lange, Tanja (14 September 2017). "Post-quantum cryptography". Nature. **549** (7671): 188–194. Bibcode:2017Natur.549..188B. doi:10.1038/nature23461. ISSN 0028-0836. PMID 28905891. S2CID 4446249.*
3. ^ *Stallings, William (3 May 1990). Cryptography and Network Security: Principles and Practice. Prentice Hall. p. 165. ISBN 9780138690175.*
4. ^ *Alvarez, Rafael; Caballero-Gil, Cándido; Santonja, Juan; Zamora, Antonio (27 June 2017). "Algorithms for Lightweight Key Exchange". Sensors. **17** (7): 1517. doi:10.3390/s17071517. ISSN 1424-8220. PMC 5551094. PMID 28654006.*

5. ^ *Menezes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A. (October 1996). "Chapter 8: Public-key encryption". Handbook of Applied Cryptography (PDF). CRC Press. pp. 425–488. ISBN 0-8493-8523-7. Retrieved 8 October 2022.*
6. ^ *Bernstein, Daniel J. (1 May 2008). "Protecting communications against forgery". Algorithmic Number Theory (PDF). Vol. 44. MSRI Publications. §5: Public-key signatures, pp. 543–545. Retrieved 8 October 2022.*
7. ^ *Bellare, Mihir; Goldwasser, Shafi (July 2008). "Chapter 10: Digital signatures". Lecture Notes on Cryptography (PDF). p. 168. Archived (PDF) from the original on 20 April 2022. Retrieved 11 June 2023.*
8. ^ Jump up to: ***a*** ***b*** *Menezes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A. (October 1996). "8: Public-key encryption". Handbook of Applied Cryptography (PDF). CRC Press. pp. 283–319. ISBN 0-8493-8523-7. Retrieved 8 October 2022.*
9. ^ *Danezis, George; Diaz, Claudia; Syverson, Paul (2010). "Chapter 13: Anonymous Communication". In Rosenberg, Burton (ed.). Handbook of Financial Cryptography and Security (PDF). Chapman & Hall/CRC. pp. 341–390. ISBN 978-1420059816. "Since PGP, beyond compressing the messages, does not make any further attempts to hide their size, it is trivial to follow a message in the network just by observing its length."*
10. ^ *Rackoff, Charles; Simon, Daniel R. (1993). "Cryptographic defense against traffic analysis". Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing. STOC '93: ACM Symposium on the Theory of Computing. Association for Computing Machinery. pp. 672–681. doi:10.1145/167088.167260. "Now, certain types of information cannot reasonably be assumed to be concealed. For instance, an upper bound on the total volume of a party's sent or received communication (of any sort) is obtainable by anyone with the resources to examine all possible physical communication channels available to that party."*
11. ^ *Karger, Paul A. (May 1977). "11: Limitations of End-to-End Encryption". Non-Discretionary Access Control for Decentralized Computing Systems (S.M. thesis). Laboratory for Computer Science, Massachusetts Institute of Technology. hdl:1721.1/149471. "The scenario just described would seem to be secure, because all data is encrypted before being passed to the communications processors. However, certain control information must be passed in cleartext from the host to the communications processor to allow the network to function. This control information consists of the destination address for the packet, the length of the packet, and the time between successive packet transmissions."*
12. ^ *Chaum, David L. (February 1981). Rivest, R. (ed.). "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". Communications of the ACM. **24** (2). Association for Computing Machinery. "Recently, some new solutions to the "key distribution problem" (the problem of providing each communicant with a secret key) have been suggested, under the name of public key cryptography. Another cryptographic problem, the "traffic analysis problem" (the problem of keeping confidential who converses with whom, and when they converse), will become increasingly important with the growth of electronic mail."*
13. ^ *Davis, Don (2001). "Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML". Proceedings of the 2001 USENIX Annual Technical Conference. USENIX. pp. 65–78. "Why is naïve Sign & Encrypt insecure? Most simply, S&E is vulnerable to "surreptitious forwarding:" Alice signs & encrypts for Bob's eyes, but Bob re-encrypts Alice's signed message for Charlie to see. In the end, Charlie believes Alice wrote to him directly, and can't detect Bob's subterfuge."*

14. ^ *An, Jee Hea (12 September 2001). Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses (Technical report). IACR Cryptology ePrint Archive. 2001/079. Retrieved 24 November 2024.*
15. ^ *Escribano Pablos, José Ignacio; González Vasco, María Isabel (April 2023). "Secure post-quantum group key exchange: Implementing a solution based on Kyber". IET Communications. **17** (6): 758–773. doi:10.1049/cmu2.12561. hdl:10016/37141. ISSN 1751-8628. S2CID 255650398.*
16. ^ *Stohrer, Christian; Lugrin, Thomas (2023), Mulder, Valentin; Mermoud, Alain; Lenders, Vincent; Tellenbach, Bernhard (eds.), "Asymmetric Encryption", Trends in Data Protection and Encryption Technologies, Cham: Springer Nature Switzerland, pp. 11–14, doi:10.1007/978-3-031-33386-6_3, ISBN 978-3-031-33386-6* {{citation}} : CS1 maint: work parameter with ISBN (link)
17. ^ *Sabiguero, Ariel; Vicente, Alfonso; Esnal, Gonzalo (November 2024). "Let There Be Trust". 2024 IEEE URUCON. doi:10.1109/URUCON63440.2024.10850093.*
18. ^ *Paar, Christof; Pelzl, Jan; Preneel, Bart (2010). Understanding Cryptography: A Textbook for Students and Practitioners. Springer. ISBN 978-3-642-04100-6.*
19. ^ *Shamir, Adi (November 1982). "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem". 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982). pp. 145–152. doi:10.1109/SFCS.1982.5.*
20. ^ *Tunggal, Abi (20 February 2020). "What Is a Man-in-the-Middle Attack and How Can It Be Prevented – What is the difference between a man-in-the-middle attack and sniffing?". UpGuard. Retrieved 26 June 2020.*[self-published source?]
21. ^ *Tunggal, Abi (20 February 2020). "What Is a Man-in-the-Middle Attack and How Can It Be Prevented - Where do man-in-the-middle attacks happen?". UpGuard. Retrieved 26 June 2020.*[self-published source?]
22. ^ *martin (30 January 2013). "China, GitHub and the man-in-the-middle". GreatFire. Archived from the original on 19 August 2016. Retrieved 27 June 2015.*[self-published source?]
23. ^ *percy (4 September 2014). "Authorities launch man-in-the-middle attack on Google". GreatFire. Retrieved 26 June 2020.*[self-published source?]
24. ^ *Bjorgvinsdottir, Hanna; Bentley, Phil (24 June 2021). "Warp2: A Method of Email and Messaging with Encrypted Addressing and Headers". arXiv:1411.6409 [cs.CR].*
25. ^ Jump up to: ***a*** ***b*** *Jevons, W.S. (1874). The Principles of Science: A Treatise on Logic and Scientific Method. Macmillan & Co. p. 141. Retrieved 18 January 2024.*
26. ^ *Weisstein, E.W. (2024). "Jevons' Number". MathWorld. Retrieved 18 January 2024.*
27. ^ *Golob, Solomon W. (1996). "On Factoring Jevons' Number". Cryptologia. **20** (3): 243. doi:10.1080/0161-119691884933. S2CID 205488749.*
28. ^ *Ellis, James H. (January 1970). "The Possibility of Secure Non-secret Digital Encryption" (PDF). CryptoCellar. Retrieved 18 January 2024.*
29. ^ *Ellis, James H. (January 1970). "The Possibility of Secure Non-secret Digital Encryption". The George Washington University. Retrieved 8 December 2025.*
30. ^ *Sawer, Patrick (11 March 2016). "The unsung genius who secured Britain's computer defences and paved the way for safe online shopping". The Telegraph.*
31. ^ Jump up to: ***a*** ***b*** *Espiner, Tom (26 October 2010). "GCHQ pioneers on birth of public key crypto". ZDNet.*
32. ^ *Singh, Simon (1999). The Code Book. Doubleday. pp. 279–292.*
33. ^ *Diffie, Whitfield; Hellman, Martin E. (November 1976). "New Directions in Cryptography" (PDF). IEEE Transactions on Information Theory. **22** (6): 644–654. Bibcode:1976ITIT...22..644D.*

CiteSeerX 10.1.1.37.9720. doi:10.1109/TIT.1976.1055638. Archived (PDF) from the original on 29 November 2014.

34. ^ *"Asymmetric encryption". IONOS Digitalguide. Retrieved 9 June 2022.*
35. ^ *Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" (PDF). Communications of the ACM. **21** (2): 120–126. CiteSeerX 10.1.1.607.2677. doi:10.1145/359340.359342. S2CID 2873616. Archived from the original (PDF) on 17 December 2008. Retrieved 15 November 2019.*
36. ^ *Robinson, Sara (June 2003). "Still Guarding Secrets after Years of Attacks, RSA Earns Accolades for its Founders" (PDF). SIAM News. **36** (5).*

- *Hirsch, Frederick J. "SSL/TLS Strong Encryption: An Introduction". Apache HTTP Server. Retrieved 17 April 2013.*. The first two sections contain a very good introduction to public-key cryptography.
- *Ferguson, Niels; Schneier, Bruce (2003). Practical Cryptography. Wiley. ISBN 0-471-22357-3.*
- *Katz, Jon; Lindell, Y. (2007). Introduction to Modern Cryptography. CRC Press. ISBN 978-1-58488-551-1.*
- *Menezes, A. J.; van Oorschot, P. C.; Vanstone, Scott A. (1997). Handbook of Applied Cryptography. Taylor & Francis. ISBN 0-8493-8523-7.*
- IEEE 1363: Standard Specifications for Public-Key Cryptography
- Christof Paar, Jan Pelzl, "Introduction to Public-Key Cryptography", 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.
- *Salomaa, Arto (1996). Public-Key Cryptography (2 ed.). Berlin: Springer. 275. doi:10.1007/978-3-662-03269-5. ISBN 978-3-662-03269-5. S2CID 24751345.*

- Oral history interview with Martin Hellman, 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
