Cryptography & its Applications

Kaylee Hartmann

First Edition, 2012

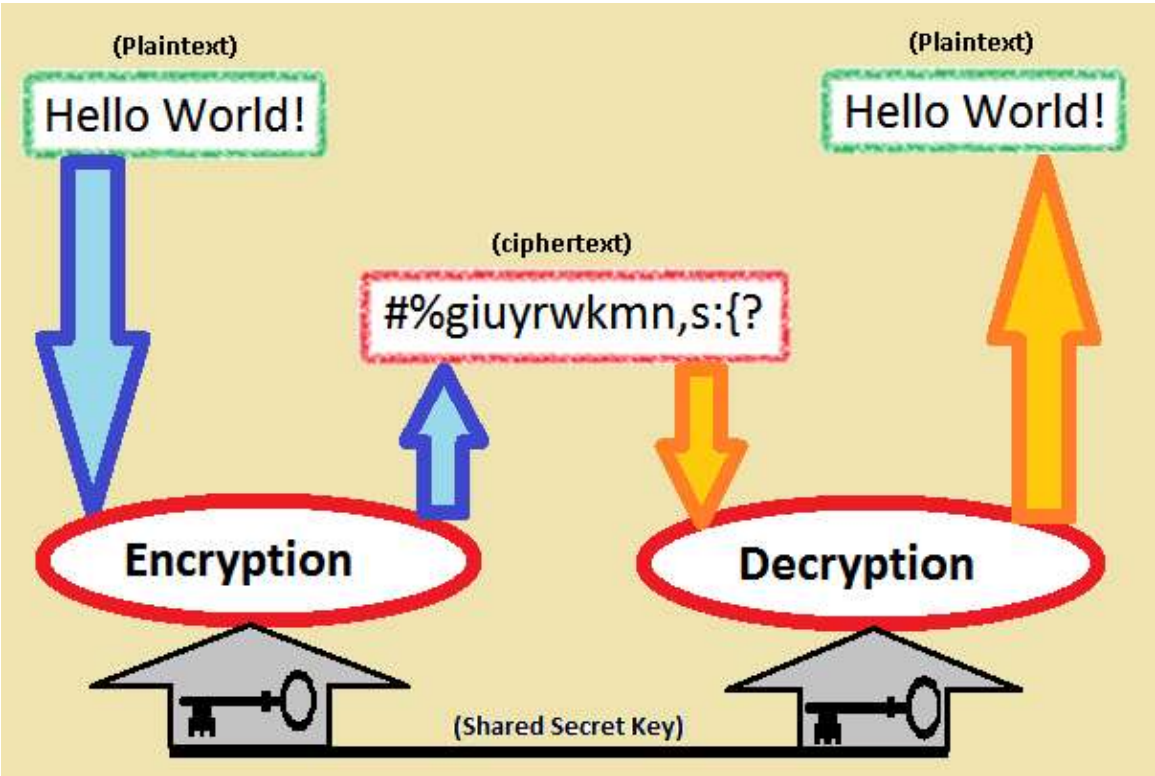
ISBN 978-81-323-1701-2

Published by:
Learning Press
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

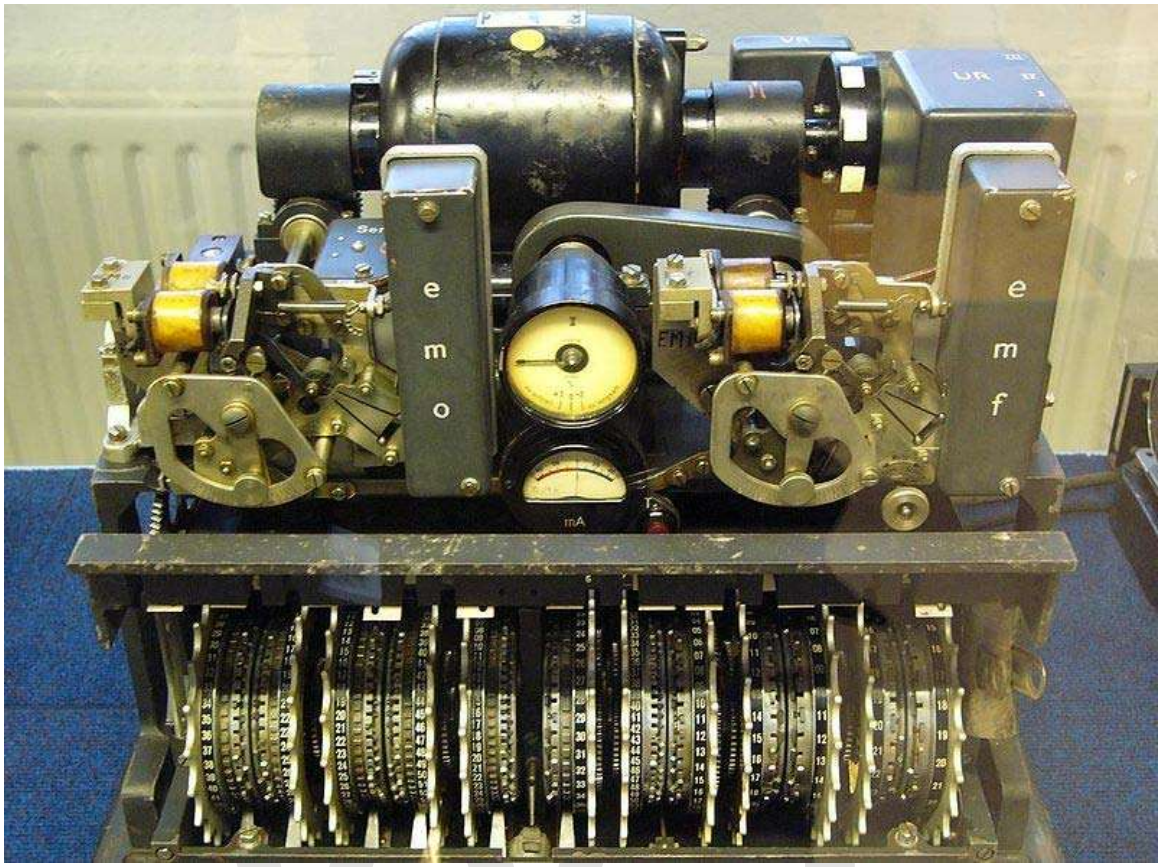
Table of Contents

Chapter 1

Cryptography



Simple explanation of encryption and decryption methods



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

Cryptography (or **cryptology**; from Greek κρυπτός, *kryptos*, "hidden, secret"; and γράφειν, *gráphin*, "writing", or -λογία, *-logia*, , "study", respectively) is the practice and study of hiding information. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

Cryptology prior to the modern age was almost synonymous with *encryption*, the conversion of information from a readable state to apparent nonsense. The sender retained the ability to decrypt the information and therefore avoid unwanted persons being able to read it. Since WWI and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

Modern cryptography follows a strongly scientific approach, and designs cryptographic algorithms around computational hardness assumptions that are assumed hard to break by an adversary. Such systems are not unbreakable in theory but it is infeasible to do so for any practical adversary. Information-theoretically secure schemes that provably cannot be broken exist but they are less practical than computationally-secure mechanisms. An example of such systems is the one-time pad.

Alongside the advancement in cryptology-related technology, the practice has raised a number of legal issues, some of which remain unresolved.

Terminology

Until modern times cryptography referred almost exclusively to *encryption*, which is the process of converting ordinary information (called plaintext) into unintelligible gibberish (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 parameter (ideally known only to the communicants) for a specific message exchange context. 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. English is more flexible than several other languages in which *cryptology* (done by cryptologists) is always used in the second sense above.

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

History of cryptography and cryptanalysis

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* (rhymes with "Italy"), an early cipher device

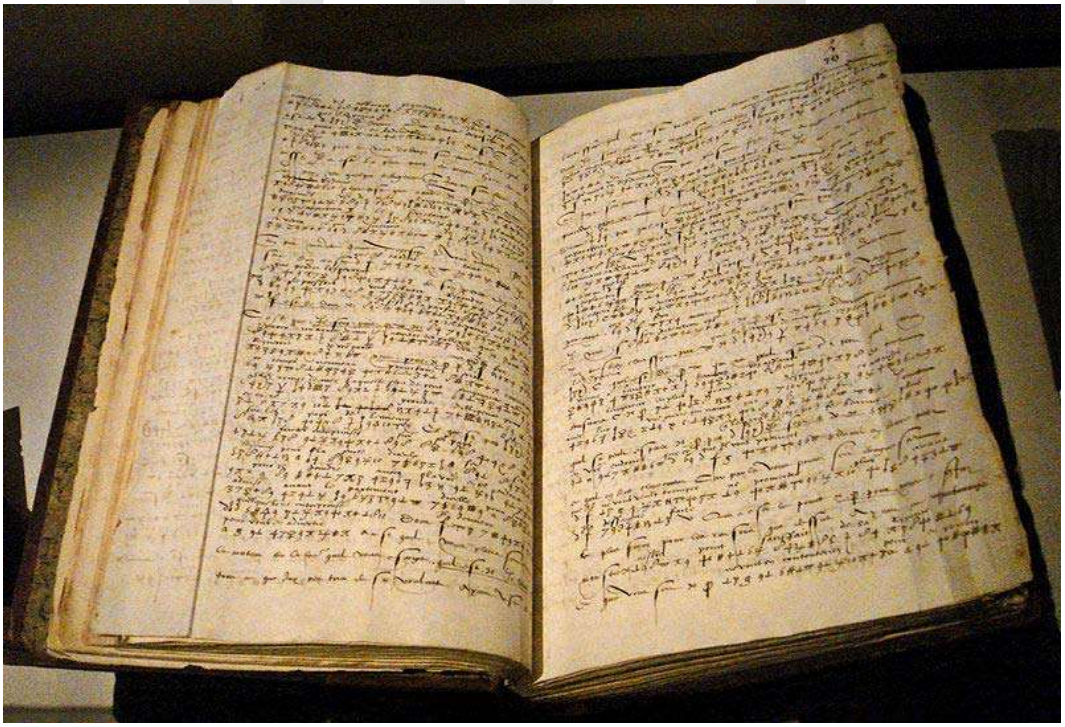
The earliest forms of secret writing required little more than local pen and paper analogs, as most people could not read. More literacy, or literate opponents, required actual cryptography. The main classical cipher types are transposition ciphers, which rearrange the order of letters in a message (e.g., '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 offered little confidentiality from enterprising opponents, and still do. 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. It was named after Julius Caesar who is reported to have used it, with a shift of 3, to communicate with his generals during his military campaigns, just like EXCESS-3 code in boolean algebra. There is record of several early Hebrew ciphers as well. The earliest known use of cryptography is some carved ciphertext on stone in Egypt (ca 1900 BC), but this may have been done for the amusement of literate observers. The next oldest is bakery recipes from Mesopotamia. Cryptography is recommended in the Kama Sutra as a way for lovers to communicate without inconvenient discovery.

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). 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. Another Greek method was developed by Polybius (now called the "Polybius Square"). 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, 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. Al-Kindi wrote a book on cryptography entitled *Risalah fi Istikhraj al-Mu'amma* (*Manuscript for the Deciphering Cryptographic Messages*), in which described the first cryptanalysis techniques, including some for polyalphabetic ciphers.

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

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 polyalphabetic ciphers of this type remained partially vulnerable to extended frequency analysis techniques.

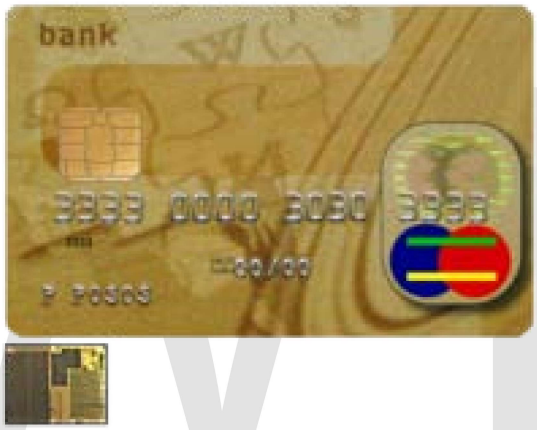
Although frequency analysis is a powerful and general technique against many ciphers, encryption has still been often effective in practice; many a would-be cryptanalyst was unaware of the technique. Breaking a message without using frequency analysis essentially required knowledge of the cipher used 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' 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 '20s 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.

The computer era

The development of digital computers and electronics after WWII 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. Alternate methods of attack (bribery, burglary, threat, torture, ...) have become more attractive in consequence.



Credit card with smart-card capabilities. The 3-by-5-mm chip embedded in the card is shown, enlarged. Smart cards combine low cost and portability with the power to compute cryptographic algorithms.

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,; 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 this or that assumption about implementation or practical use is 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.

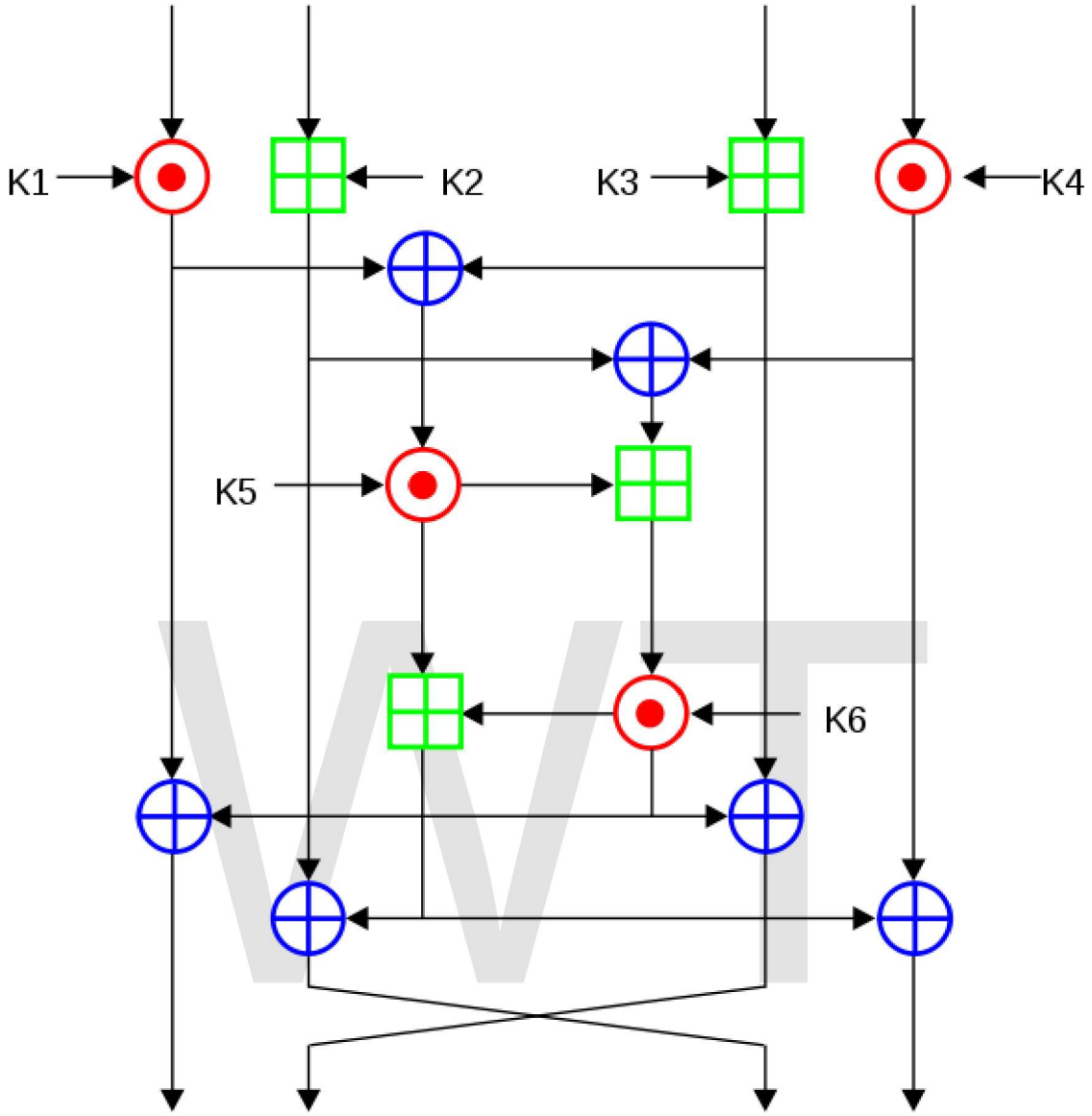
Essentially, prior to the early 20th century, cryptography was chiefly concerned with linguistic and lexicographic patterns. Since then the emphasis has shifted, and cryptography now makes extensive use of mathematics, including aspects of information theory, computational complexity, statistics, combinatorics, 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; 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.

Modern cryptography

The modern field of cryptography can be divided into several areas of study.

Symmetric-key cryptography

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.



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

The modern study of symmetric-key ciphers relates mainly to the study of block ciphers and stream ciphers and to their applications. A block cipher is, in a sense, a modern embodiment of Alberti's polyalphabetic cipher: block ciphers take as input a block of plaintext and a key, and output a block of ciphertext of the same size. Since messages are almost always longer than a single block, some method of knitting together successive blocks is required. Several have been developed, some with better security in one aspect or another than others. They are the modes of operation and must be carefully considered when using a block cipher in a cryptosystem.

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). 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 to e-mail privacy and secure remote access. Many other block ciphers have been designed and released, with considerable variation in quality.

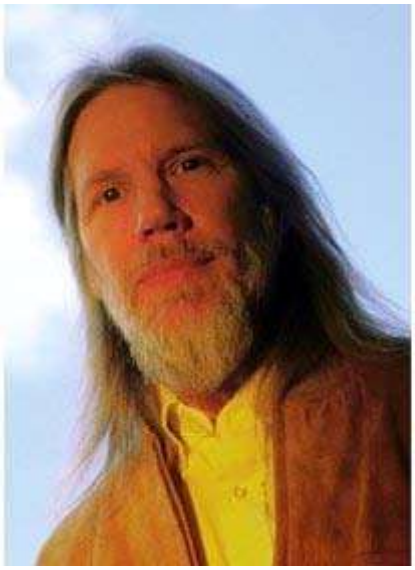
Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state which changes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

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 is underway and meant to select a new U.S. national standard, to be called SHA-3, by 2012.

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

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 straight 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.




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. 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. The historian David Kahn described public-key cryptography as "the most revolutionary new concept in the field since polyalphabetic substitution emerged in the Renaissance".

In public-key cryptosystems, the public key may be freely distributed, while its paired private key must remain secret. The *public key* is typically used for encryption, while the *private* or *secret key* is used for decryption. Diffie and Hellman showed that public-key cryptography was possible by presenting the Diffie–Hellman key exchange protocol.

In 1978, Ronald Rivest, Adi Shamir, and Len Adleman invented RSA, another public-key system.

In 1997, it finally became publicly known that asymmetric key cryptography had been invented by James H. Ellis at GCHQ, a British intelligence organization, and that, in the early 1970s, both the Diffie–Hellman and RSA algorithms had been previously developed (by Malcolm J. Williamson and Clifford Cocks, respectively).

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.

Padlock icon from the Firefox Web browser, meant to indicate a page has been sent in SSL or TLS-encrypted protected form. However, such an icon is not a guarantee of security; any subverted browser might mislead a user by displaying such an icon when a transmission is not actually being protected by SSL or TLS.

In addition to encryption, public-key cryptography can be used to implement digital signature schemes. A digital signature is reminiscent of an ordinary signature; they both have the characteristic that they are 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

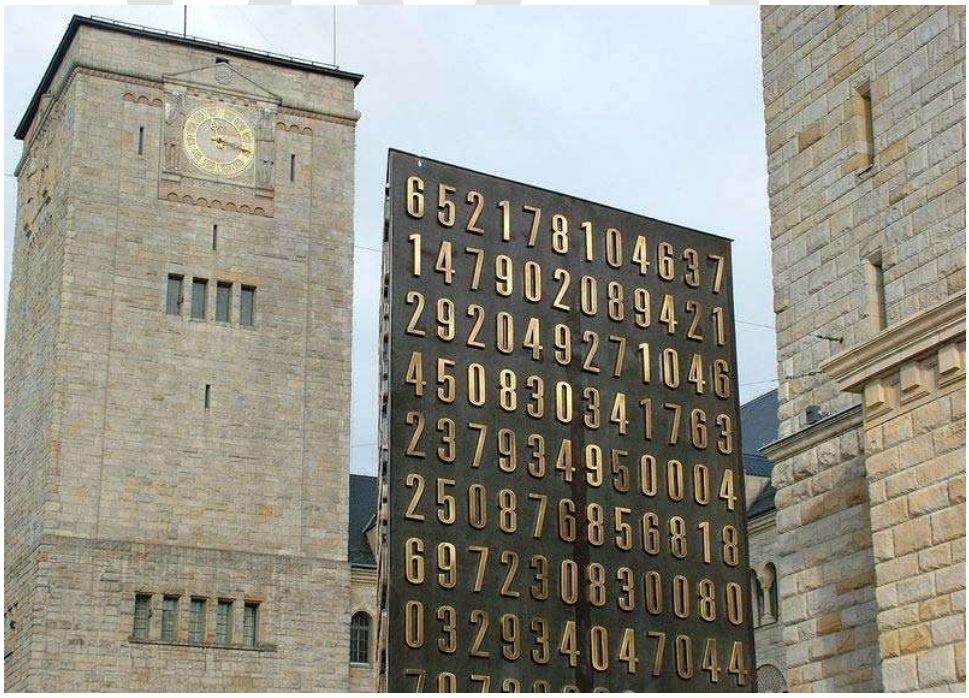
Variants of the Enigma machine, used by Germany's military and civil authorities from the late 1920s through World War II, implemented a complex electro-mechanical polyalphabetic cipher. Breaking and reading of the Enigma cipher at Poland's Cipher Bureau, for 7 years before the war, and subsequent decryption 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. 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 *use* 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 showing can be made currently, as of today, 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).



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

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 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. This is a considerable improvement on brute force attacks.

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, 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 systems, signcryption systems, etc. Some more 'theoretical' cryptosystems include interactive proof systems, (like zero-knowledge proofs,), systems for secret sharing, etc.

Until recently, most security properties of most cryptosystems were demonstrated using empirical techniques, or using ad hoc reasoning. Recently, 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).

Legal issues

Prohibitions

Cryptography has long been of interest to intelligence gathering and law enforcement agencies. Actually secret communications may be criminal or even treasonous; those whose communications are open to inspection may be less likely to be either. 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 relaxed many of these. In China, 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. 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

In the 1990s, there were several challenges to US export regulations of cryptography. One involved Philip Zimmermann's Pretty Good Privacy (PGP) encryption program; it was released in the US, together with its source code, and found its way onto the Internet in June 1991. After a complaint by RSA Security (then called RSA Data Security, Inc., or RSADSI), Zimmermann was criminally investigated by the Customs Service and the FBI for several years. No charges were ever filed, however. Also, Daniel 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.

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 are now much less strictly regulated than in the past 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. In practice today, since the relaxation in US export restrictions, and because almost every personal computer connected to the Internet, everywhere in the world, includes US-sourced web browsers such as Mozilla Firefox or Microsoft Internet Explorer, almost every Internet user worldwide has access to quality cryptography (i.e., when using sufficiently long keys with properly operating and unsubverted software, etc.) in their browsers; examples are Transport Layer Security or SSL stack. The Mozilla Thunderbird and Microsoft Outlook E-mail client programs similarly can connect to IMAP or POP servers 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.

NSA involvement

Another contentious issue connected to cryptography in the United States is the influence of the National Security Agency on cipher development and policy. 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. DES was designed to be resistant to differential cryptanalysis, a powerful and general cryptanalytic technique known to NSA and IBM, that became publicly known only when it was rediscovered in the late 1980s. According to Steven Levy, IBM rediscovered differential cryptanalysis, but kept the technique secret at NSA's request. The technique became publicly known only when Biham and Shamir re-rediscovered 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 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 was then classified (the cipher, called Skipjack, though it was declassified in 1998 long after the Clipper initiative lapsed). The secret cipher caused concerns that 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' 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

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 Professor 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.

Chapter 2

History of Cryptography

The **history of cryptography** begins thousands of years ago. Until recent decades, it has been the story of what might be called classic cryptography — that is, of methods of encryption that use pen and paper, or perhaps simple mechanical aids. In the early 20th century, the invention of complex mechanical and electromechanical machines, such as the Enigma rotor machine, provided more sophisticated and efficient means of encryption; and the subsequent introduction of electronics and computing has allowed elaborate schemes of still greater complexity, most of which are entirely unsuited to pen and paper.

The development of cryptography has been paralleled by the development of cryptanalysis — the "breaking" of codes and ciphers. The discovery and application, early on, of frequency analysis to the reading of encrypted communications has on occasion altered the course of history. Thus the Zimmermann Telegram triggered the United States' entry into World War I; and Allied reading of Nazi Germany's ciphers shortened World War II, in some evaluations by as much as two years.

Until the 1970s, secure cryptography was largely the preserve of governments. Two events have since brought it squarely into the public domain: the creation of a public encryption standard (DES), and the invention of public-key cryptography.

Classical cryptography

The earliest known use of cryptography is found in non-standard hieroglyphs carved into monuments from Egypt's Old Kingdom (ca 4500+ years ago). These are not thought to be serious attempts at secret communications, however, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers. These are examples of still other uses of cryptography, or of something that looks (impressively if misleadingly) like it. Some clay tablets from Mesopotamia somewhat later are clearly meant to protect information—they encrypt recipes, presumably commercially valuable. Later still, Hebrew scholars made use of simple monoalphabetic substitution ciphers (such as the Atbash cipher) beginning perhaps around 500 to 600 BC

A Scytale, an early device for encryption.

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). Herodotus tells us of secret messages physically concealed beneath wax on wooden tablets or as a tattoo on a slave's head concealed by regrown hair, though these are not properly examples of cryptography per se as the message, once known, is directly readable; this is known as steganography. Another Greek method was developed by Polybius (now called the "Polybius Square"). The Romans knew something of cryptography (e.g., the Caesar cipher and its variations). There is ancient mention of a book about Roman military cryptography (especially Julius Caesar's); it has been, unfortunately, lost.

Medieval cryptography

The first page of al-Kindi's manuscript *On Deciphering Cryptographic Messages*, containing the first descriptions of cryptanalysis and frequency analysis.

It was probably religiously motivated textual analysis of the Qur'an which led to the invention of the frequency analysis technique for breaking monoalphabetic substitution ciphers by Al-Kindi, an Arab mathematician, sometime around AD 800 (Ibrahim Al-Kadi −1992). It was the most fundamental cryptanalytic advance until WWII. Al-Kindi wrote a book on cryptography entitled *Risalah fi Istikhraj al-Mu'amma* (*Manuscript for the Deciphering Cryptographic Messages*), in which he described the first cryptanalysis techniques, including some for polyalphabetic ciphers, cipher classification, Arabic phonetics and syntax, and, most importantly, gave the first descriptions on frequency analysis. He also covered methods of encipherments, cryptanalysis of certain encipherments, and statistical analysis of letters and letter combinations in Arabic.

Ahmad al-Qalqashandi (1355–1418) wrote the *Subh al-a 'sha*, a 14-volume encyclopedia which included a section on cryptology. This information was attributed to Taj ad-Din Ali ibn ad-Duraihim ben Muhammad ath-Tha 'alibi al-Mausili who lived from 1312 to 1361, but whose writings on cryptography have been lost. The list of ciphers in this work included both substitution and transposition, and for the first time, a cipher with multiple substitutions for each plaintext letter. Also traced to Ibn al-Duraihim is an exposition on and worked example of cryptanalysis, including the use of tables of letter frequencies and sets of letters which can not occur together in one word.

Essentially all ciphers remained vulnerable to the cryptanalytic technique of frequency analysis until the development of the polyalphabetic cipher, and many remained so thereafter. The polyalphabetic cipher was most clearly explained by Leon Battista Alberti

around the year 1467, for which he was called the "father of Western cryptology". Johannes Trithemius, in his work *Poligraphia*, invented the tabula recta, a critical component of the Vigenère cipher. The French cryptographer Blaise de Vigenere devised a practical poly alphabetic system which bears his name, the Vigenère cipher.

In Europe, cryptography became (secretly) more important as a consequence of political competition and religious revolution. For instance, in Europe during and after the Renaissance, citizens of the various Italian states—the Papal States and the Roman Catholic Church included—were responsible for rapid proliferation of cryptographic techniques, few of which reflect understanding (or even knowledge) of Alberti's polyalphabetic advance. 'Advanced ciphers', even after Alberti, weren't as advanced as their inventors / developers / users claimed (and probably even themselves believed). They were regularly broken. This over-optimism may be inherent in cryptography for it was then, and remains today, fundamentally difficult to accurately know how vulnerable your system actually is. In the absence of knowledge, guesses and hopes, as may be expected, are common.

Cryptography, cryptanalysis, and secret agent/courier betrayal featured in the Babington plot during the reign of Queen Elizabeth I which led to the execution of Mary, Queen of Scots. An encrypted message from the time of the Man in the Iron Mask (decrypted just prior to 1900 by Étienne Bazeries) has shed some, regrettably non-definitive, light on the identity of that real, if legendary and unfortunate, prisoner.

Outside of Europe, after the end of the Muslim Golden Age at the hand of the Mongols, cryptography remained comparatively undeveloped. Cryptography in Japan seems not to have been used until about 1510, and advanced techniques were not known until after the opening of the country to the West beginning in the 1860s. During the 1920s, it was Polish naval officers who assisted the Japanese military with code and cipher development.

Cryptography from 1800 to World War II

Although cryptography has a long and complex history, it wasn't until the 19th century that it developed anything more than ad hoc approaches to either encryption or cryptanalysis (the science of finding weaknesses in crypto systems). Examples of the latter include Charles Babbage's Crimean War era work on mathematical cryptanalysis of polyalphabetic ciphers, redeveloped and published somewhat later by the Prussian Friedrich Kasiski. Understanding of cryptography at this time typically consisted of hard-won rules of thumb; see, for example, Auguste Kerckhoffs' cryptographic writings in the latter 19th century. Edgar Allan Poe used systematic methods to solve ciphers in the 1840s. In particular he placed a notice of his abilities in the Philadelphia paper *Alexander's Weekly (Express) Messenger*, inviting submissions of ciphers, of which he proceeded to solve almost all. His success created a public stir for some months. He later wrote an essay on methods of cryptography which proved useful as an introduction for novice British cryptanalysts attempting to break German codes and ciphers during World

War I, and a famous story, *The Gold-Bug*, in which cryptanalysis was a prominent element.

Cryptography, and its misuse, were involved in the plotting which led to the execution of Mata Hari and in the conniving which led to the travesty of Dreyfus' conviction and imprisonment, both in the early 20th century. Fortunately, cryptographers were also involved in exposing the machinations which had led to Dreyfus' problems; Mata Hari, in contrast, was shot.

In World War I the Admiralty's Room 40 broke German naval codes and played an important role in several naval engagements during the war, notably in detecting major German sorties into the North Sea that led to the battles of Dogger Bank and Jutland as the British fleet was sent out to intercept them. However its most important contribution was probably in decrypting the Zimmermann Telegram, a cable from the German Foreign Office sent via Washington to its ambassador Heinrich von Eckardt in Mexico which played a major part in bringing the United States into the war.

In 1917, Gilbert Vernam proposed a teletype cipher in which a previously-prepared key, kept on paper tape, is combined character by character with the plaintext message to produce the cyphertext. This led to the development of electromechanical devices as cipher machines.

Mathematical methods proliferated in the period prior to World War II (notably in William F. Friedman's application of statistical techniques to cryptanalysis and cipher development and in Marian Rejewski's initial break into the German Army's version of the Enigma system) in 1932.

World War II cryptography

The Enigma machine was widely used by Nazi Germany; its cryptanalysis by the Allies provided vital Ultra intelligence.

By World War II, mechanical and electromechanical cipher machines were in wide use, although—where such machines were impractical—manual systems continued in use. Great advances were made in both cipher design and cryptanalysis, all in secrecy. Information about this period has begun to be declassified as the official British 50-year secrecy period has come to an end, as US archives have slowly opened, and as assorted memoirs and articles have appeared.

The Germans made heavy use, in several variants, of an electromechanical rotor machine known as Enigma. Mathematician Marian Rejewski, at Poland's Cipher Bureau, in December 1932 deduced the detailed structure of the German Army Enigma, using mathematics and limited documentation supplied by Captain Gustave Bertrand of French military intelligence. This was the greatest breakthrough in cryptanalysis in a thousand years and more, according to historian David Kahn. Rejewski and his mathematical Cipher Bureau colleagues, Jerzy Różycki and Henryk Zygalski, continued reading Enigma and keeping pace with the evolution of the German Army machine's components and encipherment procedures. As the Poles' resources became strained by the changes being introduced by the Germans, and as war loomed, the Cipher Bureau, on the Polish General Staff's instructions, on 25 July 1939, at Warsaw, initiated French and British intelligence representatives into the secrets of Enigma decryption.

Soon after World War II broke out on 1 September 1939, key Cipher Bureau personnel were evacuated southeastward; on 17 September, as the Soviet Union attacked Poland, they crossed into Romania. From there they reached Paris, France; at PC Bruno, near Paris, they continued breaking Enigma, collaborating with British cryptologists at Bletchley Park as the British got up to speed on breaking Enigma. In due course, the British cryptographers—whose ranks included many chess masters and mathematics dons such as Gordon Welchman, Max Newman, and Alan Turing the conceptual founder of modern computing — substantially advanced the scale and technology of Enigma decryption.

At the end of the War, on 19 April 1945, Britain's top military officers were told that they could never reveal that the German Enigma cipher had been broken because it would give the defeated enemy the chance to say they "were not well and fairly beaten".

US Navy cryptographers (with cooperation from British and Dutch cryptographers after 1940) broke into several Japanese Navy crypto systems. The break into one of them, JN-25, famously led to the US victory in the Battle of Midway; and to the publication of that fact in the Chicago Tribune shortly after the battle, though the Japanese seem not to have noticed for they kept using the JN-25 system. A US Army group, the SIS, managed to break the highest security Japanese diplomatic cipher system (an electromechanical 'stepping switch' machine called Purple by the Americans) even before WWII began. The Americans referred to the intelligence resulting from cryptanalysis, perhaps especially that from the Purple machine, as 'Magic'. The British eventually settled on 'Ultra' for intelligence resulting from cryptanalysis, particularly that from message traffic protected by the various Enigmas. An earlier British term for Ultra had been 'Boniface' in an attempt to suggest, if betrayed, that it might have an individual agent as a source.

The German military also deployed several mechanical attempts at a one-time pad. Bletchley Park called them the Fish ciphers, and Max Newman and colleagues designed and deployed the Heath Robinson, and then the world's first programmable digital electronic computer, the Colossus, to help with their cryptanalysis. The German Foreign Office began to use the one-time pad in 1919; some of this traffic was read in WWII

partly as the result of recovery of some key material in South America that was discarded without sufficient care by a German courier.

The Japanese Foreign Office used a locally developed electrical stepping switch based system (called Purple by the US), and also had used several similar machines for attaches in some Japanese embassies. One of these was called the 'M-machine' by the US, another was referred to as 'Red'. All were broken, to one degree or another, by the Allies.

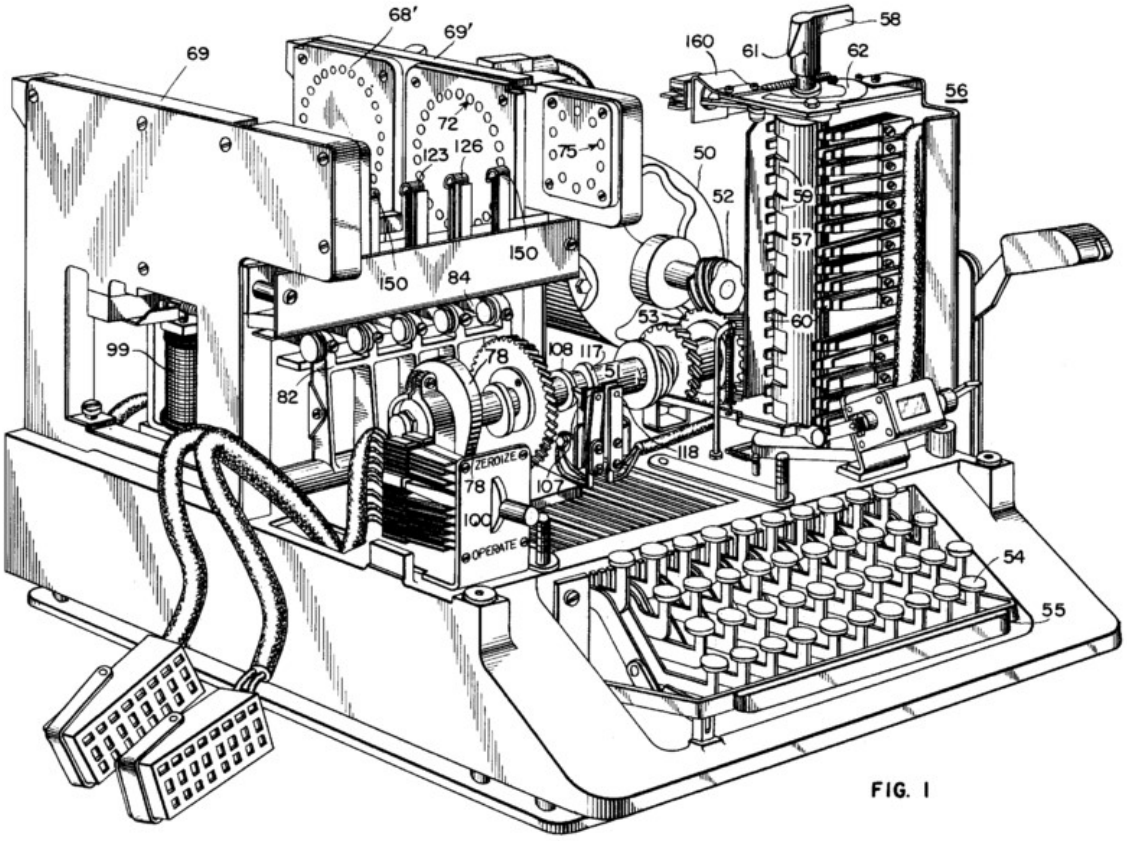


FIG. 1

SIGABA is described in U.S. Patent 6,175,625, filed in 1944 but not issued until 2001.

Allied cipher machines used in WWII included the British TypeX and the American SIGABA; both were electromechanical rotor designs similar in spirit to the Enigma, albeit with major improvements. Neither is known to have been broken by anyone during the War. The Poles used the Lacida machine, but its security was found to be less than intended (by Polish Army cryptographers in the UK), and its use was discontinued. US troops in the field used the M-209 and the still less secure M-94 family machines. British SOE agents initially used 'poem ciphers' (memorized poems were the encryption/decryption keys), but later in the War, they began to switch to one-time pads.

The VIC cipher (used at least until 1957 in connection with Rudolf Abel's NY spy ring) was a very complex hand cipher, and is claimed to be the most complicated known to have been used by the Soviets, according to David Kahn in *Kahn on Codes*.

Modern cryptography

Both cryptography and cryptanalysis have become far more mathematical since World War II. Even so, it has taken the wide availability of computers, and the Internet as a communications medium, to bring effective cryptography into common use by anyone other than national governments or similarly large enterprises.

Shannon

The era of modern cryptography really begins with Claude Shannon, arguably the father of mathematical cryptography, with the work he did during WWII on communications security. In 1949 he published *Communication Theory of Secrecy Systems* in the *Bell System Technical Journal* and a little later the book *The Mathematical Theory of Communication* (expanding on an earlier article "A Mathematical Theory of Communication") with Warren Weaver. Both included results from his WWII work. These, in addition to his other works on information and communication theory established a solid theoretical basis for cryptography and also for much of cryptanalysis. And with that, cryptography more or less disappeared into secret government communications organizations such as NSA, GCHQ, and their equivalents elsewhere. Very little work was again made public until the mid 1970s, when everything changed.

An encryption standard

The mid-1970s saw two major public (i.e., non-secret) advances. First was the publication of the draft Data Encryption Standard in the U.S. *Federal Register* on 17 March 1975. The proposed DES cipher was submitted by a research group at IBM, at the invitation of the National Bureau of Standards (now NIST), in an effort to develop secure electronic communication facilities for businesses such as banks and other large financial organizations. After 'advice' and modification by NSA, acting behind the scenes, it was adopted and published as a Federal Information Processing Standard Publication in 1977 (currently at FIPS 46-3). DES was the first publicly accessible cipher to be 'blessed' by a national agency such as NSA. The release of its specification by NBS stimulated an explosion of public and academic interest in cryptography.

The aging DES was officially replaced by the Advanced Encryption Standard (AES) in 2001 when NIST announced FIPS 197. After an open competition, NIST selected Rijndael, submitted by two Belgian cryptographers, to be the AES. DES, and more secure variants of it (such as Triple DES), are still used today, having been incorporated into many national and organizational standards. However, its 56-bit key-size has been shown to be insufficient to guard against brute force attacks (one such attack, undertaken by the cyber civil-rights group Electronic Frontier Foundation in 1997, succeeded in 56 hours—the story is in *Cracking DES*, published by O'Reilly and Associates). As a result, use of straight DES encryption is now without doubt insecure for use in new cryptosystem designs, and messages protected by older cryptosystems using DES, and indeed all messages sent since 1976 using DES, are also at risk. Regardless of DES' inherent quality, the DES key size (56-bits) was thought to be too small by some even in 1976,

perhaps most publicly by Whitfield Diffie. There was suspicion that government organizations even then had sufficient computing power to break DES messages; clearly others have achieved this capability.

Public key

The second development, in 1976, was perhaps even more important, for it fundamentally changed the way cryptosystems might work. This was the publication of the paper New Directions in Cryptography by Whitfield Diffie and Martin Hellman. It introduced a radically new method of distributing cryptographic keys, which went far toward solving one of the fundamental problems of cryptography, key distribution, and has become known as Diffie-Hellman key exchange.

Prior to that time, all useful modern encryption algorithms had been 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 it secret. All of the electromechanical machines used in WWII were of this logical class, as were the Caesar and Atbash ciphers and essentially all cipher systems throughout history. The 'key' for a code is, of course, the codebook, which must likewise be distributed and kept secret, and so shares most of the same problems in practice.

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 (the term usually used is 'via a secure channel') such as a trustworthy courier with a briefcase handcuffed to a wrist, or face-to-face contact, or a loyal carrier pigeon. This requirement is never trivial and very rapidly becomes unmanageable as the number of participants increases, or when secure channels aren't available for key exchange, 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. A system of this kind is known as a secret key, or symmetric key cryptosystem. D-H key exchange (and succeeding improvements and variants) made operation of these systems much easier, and more secure, than had ever been possible before in all of history.

In contrast, asymmetric key encryption uses a pair of mathematically related keys, each of which decrypts the encryption performed using the other. Some, but not all, of these algorithms have the additional property that one of the paired keys cannot be deduced from the other by any known method other than trial and error. An algorithm of this kind is known as a public key or asymmetric key system. Using such an algorithm, only one key pair is needed per user. By designating one key of the pair as private (always secret), and the other as public (often widely available), no secure channel is needed for key exchange. So long as the private key stays secret, the public key can be widely known for a very long time without compromising security, making it safe to reuse the same key pair indefinitely.

For two users of an asymmetric key algorithm to communicate securely over an insecure channel, each user will need to know their own public and private keys as well as the other user's public key. Take this basic scenario: Alice and Bob each have a pair of keys they've been using for years with many other users. At the start of their message, they exchange public keys, unencrypted over an insecure line. Alice then encrypts a message using her private key, and then re-encrypts that result using Bob's public key. The double-encrypted message is then sent as digital data over a wire from Alice to Bob. Bob receives the bit stream and decrypts it using his own private key, and then decrypts that bit stream using Alice's public key. If the final result is recognizable as a message, Bob can be confident that the message actually came from someone who knows Alice's private key (presumably actually her if she's been careful with her private key), and that anyone eavesdropping on the channel will need Bob's private key in order to understand the message.

Asymmetric algorithms rely for their effectiveness on a class of problems in mathematics called one-way functions, which require relatively little computational power to execute, but vast amounts of power to reverse, if reversal is possible at all. A classic example of a one-way function is multiplication of very large prime numbers. It's fairly quick to multiply two large primes, but very difficult to find the factors of the product of two large primes. Because of the mathematics of one-way functions, most possible keys are bad choices as cryptographic keys; only a small fraction of the possible keys of a given length are suitable, and so asymmetric algorithms require very long keys to reach the same level of security provided by relatively shorter symmetric keys. The need to both generate the key pairs, and perform the encryption/decryption operations make asymmetric algorithms computationally expensive, compared to most symmetric algorithms. Since symmetric algorithms can often use any sequence of (random, or at least unpredictable) bits as a key, a disposable *session key* can be quickly generated for short-term use. Consequently, it is common practice to use a long asymmetric key to exchange a disposable, much shorter (but just as strong) symmetric key. The slower asymmetric algorithm securely sends a symmetric session key, and the faster symmetric algorithm takes over for the remainder of the message.

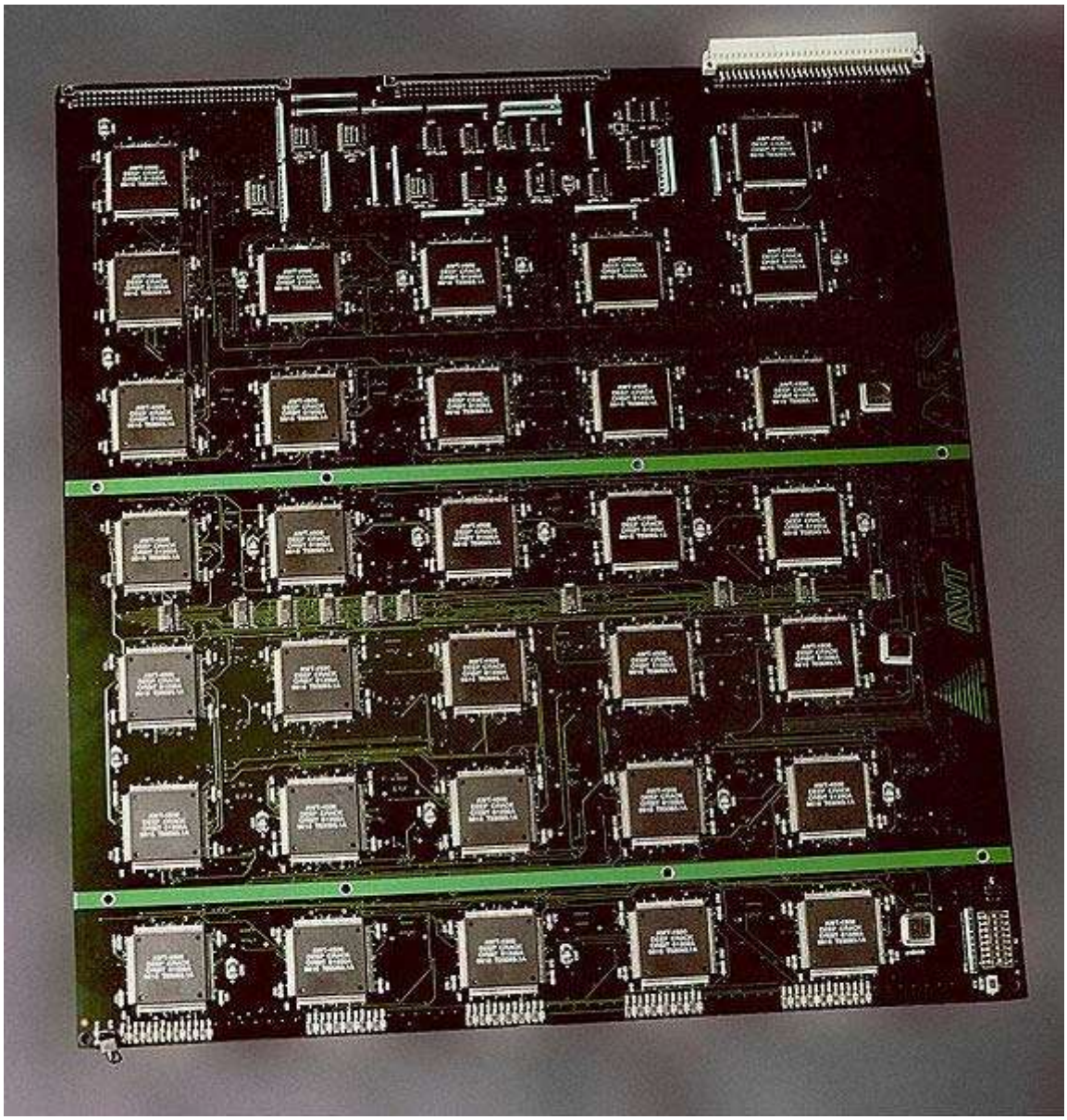
Asymmetric key cryptography, Diffie-Hellman key exchange, and the best known of the public key / private key algorithms (i.e., what is usually called the RSA algorithm), all seem to have been independently developed at a UK intelligence agency before the public announcement by Diffie and Hellman in 1976. GCHQ has released documents claiming they had developed public key cryptography before the publication of Diffie and Hellman's paper. Various classified papers were written at GCHQ during the 1960s and 1970s which eventually led to schemes essentially identical to RSA encryption and to Diffie-Hellman key exchange in 1973 and 1974. Some of these have now been published, and the inventors (James H. Ellis, Clifford Cocks, and Malcolm Williamson) have made public (some of) their work.

Cryptography politics

The public developments of the 1970s broke the near monopoly on high quality cryptography held by government organizations. For the first time ever, those outside government organizations had access to cryptography not readily breakable by anyone (including governments). Considerable controversy, and conflict, both public and private, began more or less immediately. It has not yet subsided. In many countries, for example, export of cryptography is subject to restrictions. Until 1996 export from the U.S. of cryptography using keys longer than 40 bits (too small to be very secure against a knowledgeable attacker) was sharply limited. As recently as 2004, former FBI Director Louis Freeh, testifying before the 9/11 Commission, called for new laws against public use of encryption.

One of the most significant people favoring strong encryption for public use was Phil Zimmermann. He wrote and then in 1991 released PGP (Pretty Good Privacy), a very high quality crypto system. He distributed a freeware version of PGP when he felt threatened by legislation then under consideration by the US Government that would require backdoors to be included in all cryptographic products developed within the US. His system was released worldwide shortly after he released it in the US, and that began a long criminal investigation of him by the US Government Justice Department for the alleged violation of export restrictions. The Justice Department eventually dropped its case against Zimmermann, and the freeware distribution of PGP has continued around the world. PGP even eventually became an open Internet standard (RFC 2440 or OpenPGP).

Modern cryptanalysis



Modern cryptanalysts sometimes harness large numbers of integrated circuits. This board is part of the EFF DES cracker, which contained over 1800 custom chips and could brute force a DES key in a matter of days.

While modern ciphers like AES and the higher quality asymmetric ciphers are widely considered unbreakable, poor designs and implementations are still sometimes adopted and there have been important cryptanalytic breaks of deployed crypto systems in recent years. Notable examples of broken crypto designs include DES, the first Wi-Fi encryption scheme WEP, the Content Scrambling System used for encrypting and controlling DVD use, the A5/1 and A5/2 ciphers used in GSM cell phones, and the CRYPTO1 cipher used in the widely deployed MIFARE Classic smart cards from NXP Semiconductors, a spun off division of Philips Electronics. All of these are symmetric

ciphers. Thus far, not one of the mathematical ideas underlying public key cryptography has been proven to be 'unbreakable', and so some future mathematical analysis advance might render systems relying on them insecure. While few informed observers foresee such a breakthrough, the key size recommended for security as best practice keeps increasing as increased computing power required for breaking codes becomes cheaper and more available.

Chapter 3

Transposition Cipher and Substitution Cipher

Transposition cipher

In cryptography, a **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed. Mathematically a bijective function is used on the characters' positions to encrypt and an inverse function to decrypt.

Following are some implementations.

Rail Fence cipher

The Rail Fence cipher is a form of transposition cipher that gets its name from the way in which it is encoded. In the rail fence cipher, the plaintext is written downwards on successive "rails" of an imaginary fence, then moving up when we get to the bottom. The message is then read off in rows. For example, using three "rails" and a message of 'WE ARE DISCOVERED. FLEE AT ONCE', the cipherer writes out:

```
W . . . E . . . C . . . R . . . L . . . T . . . E
. E . R . D . S . O . E . E . F . E . A . O . C .
. . A . . . I . . . V . . . D . . . E . . . N . .
```

Then reads off:

```
WECRL TEERD SOEEF EAOCA IVDEN
```

(The cipherer has broken this ciphertext up into blocks of five to help avoid errors.)

Route cipher

In a route cipher, the plaintext is first written out in a grid of given dimensions, then read off in a pattern given in the key. For example, using the same plaintext that we used for rail fence:

```
W R I O R F E O E
E E S V E L A N J
A D C E D E T C X
```

The key might specify "spiral inwards, clockwise, starting from the top right". That would give a cipher text of:

```
EJXCTEDECDAEWRIORFEONALEVSE
```

Route ciphers have many more keys than a rail fence. In fact, for messages of reasonable length, the number of possible keys is potentially too great to be enumerated even by modern machinery. However, not all keys are equally good. Badly chosen routes will leave excessive chunks of plaintext, or text simply reversed, and this will give cryptanalysts a clue as to the routes.

An interesting variation of the route cipher was the Union Route Cipher, used by Union forces during the American Civil War. This worked much like an ordinary route cipher, but transposed whole words instead of individual letters. Because this would leave certain highly sensitive words exposed, such words would first be concealed by code. The cipher clerk may also add entire null words, which were often chosen to make the ciphertext humorous.

Columnar transposition

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word `ZEBRAS` is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

In a regular columnar transposition cipher, any spare spaces are filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword. For example, suppose we use the keyword `ZEBRAS` and the message `WE ARE DISCOVERED. FLEE AT ONCE`. In a regular columnar transposition, we write this into the grid as:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
```

```
E Q K J E U
```

Providing five nulls (QKJEU) at the end. The ciphertext is then read off as:

```
EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE
```

In the irregular case, the columns are not completed by nulls:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

This results in the following ciphertext:

```
EVLNA CDTES EAROF ODEEC WIREE
```

To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. Then he can write the message out in columns again, then re-order the columns by reforming the key word.

Columnar transposition continued to be used for serious purposes as a component of more complex ciphers at least into the 1950's.

Double transposition

A single columnar transposition could be attacked by guessing possible column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible anagrams. Thus to make it stronger, a double transposition was often used. This is simply a columnar transposition applied twice. The same key can be used for both transpositions, or two different keys can be used.

As an example, we can take the result of the irregular columnar transposition in the previous section, and perform a second encryption with a different keyword, STRIPE, which gives the permutation "564231":

```
5 6 4 2 3 1
E V L N A C
D T E S E A
R O F O D E
E C W I R E
E
```

As before, this is read off columnwise to give the ciphertext:

```
CAEEN SOIAE DRLEF WEDRE EVTOC
```

If multiple messages of exactly the same length are encrypted using the same keys, they can be anagrammed simultaneously. This can lead to both recovery of the messages, and to recovery of the keys (so that every other message sent with those keys can be read).

During World War I, the German military used a double columnar transposition cipher, changing the keys infrequently. The system was regularly solved by the French, naming it Übchi, who were typically able to quickly find the keys once they'd intercepted a number of messages of the same length, which generally took only a few days. However, the French success became widely-known and, after a publication in *Le Matin*, the Germans changed to a new system on 18 November 1914.

During World War II, the double transposition cipher was used by Dutch Resistance groups, the French Maquis and the British Special Operations Executive (SOE), which was in charge of managing underground activities in Europe. It was also used by agents of the American Office of Strategic Services and as an emergency cipher for the German Army and Navy.

Until the invention of the VIC cipher, double transposition was generally regarded as the most complicated cipher that an agent could operate reliably under difficult field conditions.

Myszkowski transposition

A variant form of columnar transposition, proposed by Émile Victor Théodore Myszkowski in 1902, requires a keyword with recurrent letters. In usual practice, subsequent occurrences of a keyword letter are treated as if the next letter in alphabetical order, *e.g.,* the keyword TOMATO yields a numeric keystring of "532164."

In Myszkowski transposition, recurrent keyword letters are numbered identically, TOMATO yielding a keystring of "432143."

```
4 3 2 1 4 3
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

Plaintext columns with unique numbers are transcribed downward; those with recurring numbers are transcribed left to right:

```
ROFOA CDTED SEEEA CWEIV RLENE
```

Disrupted transposition

In a disrupted transposition, certain positions in a grid are blanked out, and not used when filling in the plaintext. This breaks up regular patterns and makes the cryptanalyst's job more difficult.

Grilles

Another form of transposition cipher uses *grilles*, or physical masks with cut-outs, rather than a mathematical algorithm. This can produce a highly irregular transposition over the period specified by the size of the grille, but requires the correspondents to keep a physical key secret. Grilles were first proposed in 1550, and were still in military use for the first few months of World War One.

Detection and cryptanalysis

Since transposition does not affect the frequency of individual symbols, simple transposition can be easily detected by the cryptanalyst by doing a frequency count. If the ciphertext exhibits a frequency distribution very similar to plaintext, it is most likely a transposition. This can then often be attacked by anagramming - sliding pieces of ciphertext around, then looking for sections that look like anagrams of English words, and solving the anagrams. Once such anagrams have been found, they reveal information about the transposition pattern, and can consequently be extended.

Simpler transpositions also often suffer from the property that keys very close to the correct key will reveal long sections of legible plaintext interspersed by gibberish. Consequently such ciphers may be vulnerable to optimum seeking algorithms such as genetic algorithms.

A detailed description of the cryptanalysis of a German transposition cipher can be found in chapter 7 of Herbert Yardley's "The American Black Chamber."

Combinations

Transposition is often combined with other techniques. For example, a simple substitution cipher combined with a columnar transposition avoids the weakness of both. Replacing high frequency ciphertext symbols with high frequency plaintext letters does not reveal chunks of plaintext because of the transposition. Anagramming the transposition does not work because of the substitution. The technique is particularly powerful if combined with fractionation (see below). A disadvantage is that such ciphers are considerably more laborious and error prone than simpler ciphers.

Fractionation

Transposition is particularly effective when employed with fractionation - that is, a preliminary stage that divides each plaintext symbol into several ciphertext symbols. For example, the plaintext alphabet could be written out in a grid, then every letter in the message replaced by its co-ordinates. Another method of fractionation is to simply convert the message to Morse code, with a symbol for spaces as well as dots and dashes.

When such a fractionated message is transposed, the components of individual letters become widely separated in the message, thus achieving Claude E. Shannon's diffusion. Examples of ciphers that combine fractionation and transposition include the bifid cipher, the trifid cipher, the ADFGVX cipher and the VIC cipher.

Another choice would be to replace each letter with its binary representation, transpose that, and then convert the new binary string into the corresponding ASCII characters. Looping the scrambling process on the binary string multiple times before changing it into ASCII characters would likely make it harder to break. Many modern block ciphers use more complex forms of transposition related to this simple idea.

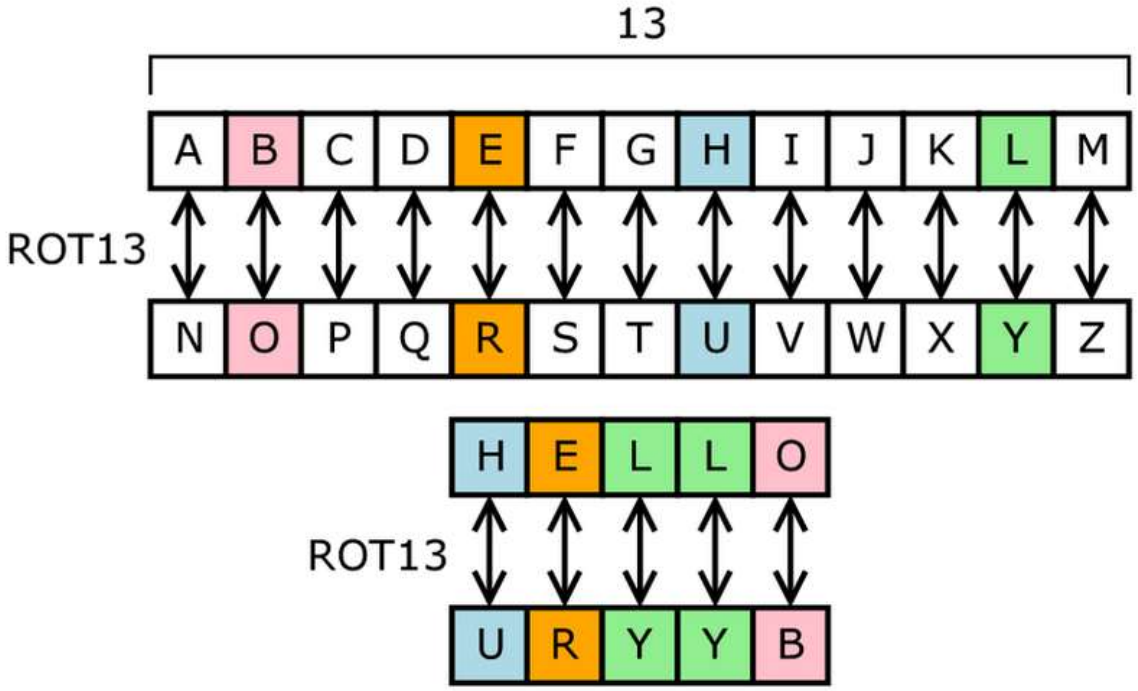
Substitution cipher

In cryptography, a **substitution cipher** is a method of encryption by which units of plaintext are replaced with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a **simple substitution cipher**; a cipher that operates on larger groups of letters is termed **polygraphic**. A **monoalphabetic cipher** uses fixed substitution over the entire message, whereas a **polyalphabetic cipher** uses a number of substitutions at different times in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice-versa.

Simple substitution



ROT13 is a Caesar cipher, a type of substitution cipher. In ROT13, the alphabet is rotated 13 steps.

Substitution over a single letter—**simple substitution**—can be demonstrated by writing out the alphabet in some order to represent the substitution. This is termed a **substitution alphabet**. The cipher alphabet may be shifted or reversed (creating the Caesar and Atbash ciphers, respectively) or scrambled in a more complex fashion, in which case it is called a *mixed alphabet* or *deranged alphabet*. Traditionally, mixed alphabets are created by first writing out a keyword, removing repeated letters in it, then writing all the remaining letters in the alphabet.

Examples

Using this system, the keyword "`zebras`" gives us the following alphabets:

Plaintext alphabet: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
Ciphertext alphabet: `ZEBRASCDFGHIJKLMNOPQTUVWXY`

A message of

```
flee at once. we are discovered!
```

enciphers to

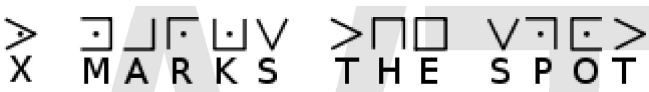
```
SIAA ZQ LKBA. VA ZOA RFPBLUAOAR!
```

Traditionally, the ciphertext is written out in blocks of fixed length, omitting punctuation and spaces; this is done to help avoid transmission errors and to disguise word boundaries from the plaintext. These blocks are called "groups", and sometimes a "group count" (i.e., the number of groups) is given as an additional check. Five letter groups are traditional, dating from when messages used to be transmitted by telegraph:

```
SIAAZ QLKBA VAZOA RFPBL UAOAR
```

If the length of the message happens not to be divisible by five, it may be padded at the end with "nulls". These can be any characters that decrypt to obvious nonsense, so the receiver can easily spot them and discard them.

The ciphertext alphabet is sometimes different from the plaintext alphabet; for example, in the pigpen cipher, the ciphertext consists of a set of symbols derived from a grid. For example:



Such features make little difference to the security of a scheme, however — at the very least, any set of strange symbols can be transcribed back into an A-Z alphabet and dealt with as normal.

In lists and catalogues for sales people sometimes a very simple encryption is used to replace numeric digits by letters.

Plain digits: `1234567890`
Ciphertext alphabet: `MAKEPROFIT`

Example: MAT would be used to represent 120.

Security for simple substitution ciphers

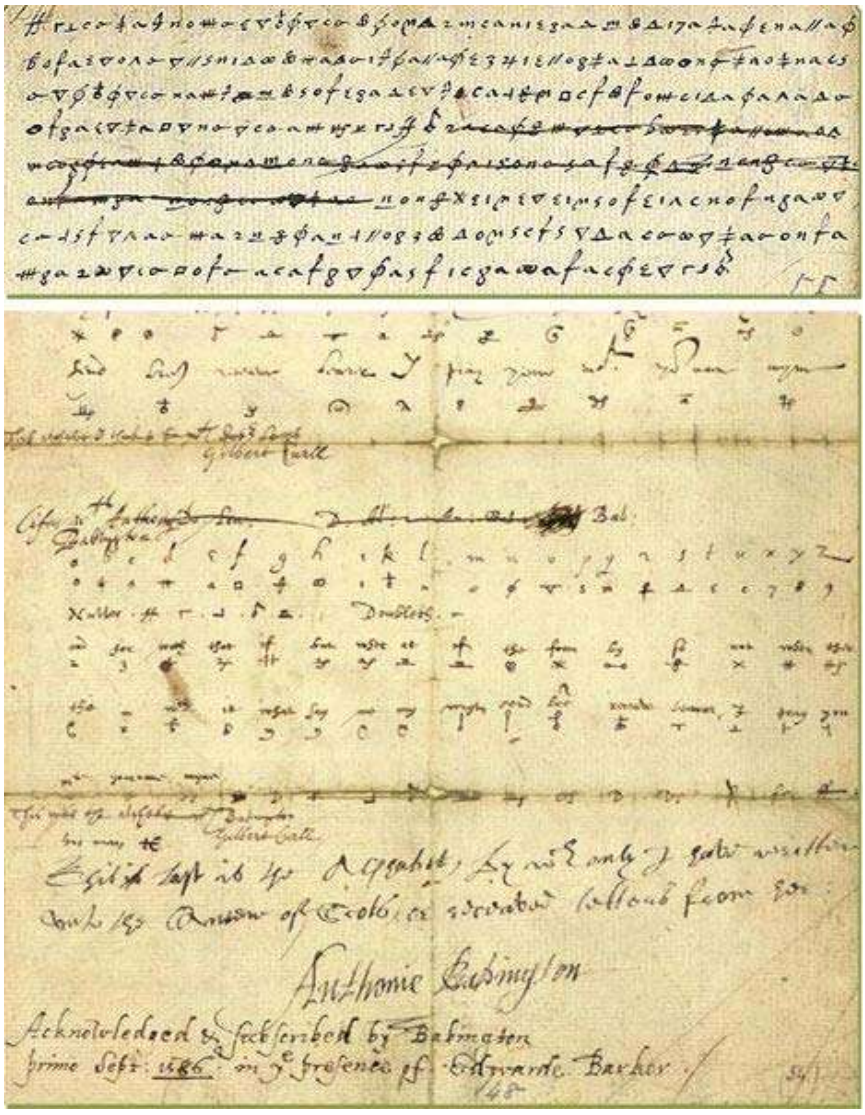
A disadvantage of this method of derangement is that the last letters of the alphabet (which are mostly low frequency) tend to stay at the end. A stronger way of constructing a mixed alphabet is to perform a columnar transposition on the ordinary alphabet using the keyword, but this is not often done.

Although the number of possible keys is very large ($26! \approx 2^{88.4}$, or about 88 bits), this cipher is not very strong, being easily broken. Provided the message is of reasonable length (see below), the cryptanalyst can deduce the probable meaning of the most common symbols by analyzing the frequency distribution of the ciphertext—frequency analysis. This allows formation of partial words, which can be tentatively filled in, progressively expanding the (partial) solution. In some cases, underlying words can also

be determined from the pattern of their letters; for example, *attract*, *osseous*, and words with those two as the root are the only common English words with the pattern *ABBCADB*. Many people solve such ciphers for recreation, as with cryptogram puzzles in the newspaper.

According to the unicity distance of English, 27.6 letters of ciphertext are required to crack a mixed alphabet simple substitution. In practice, typically about 50 letters are needed, although some messages can be broken with fewer if unusual patterns are found. In other cases, the plaintext can be contrived to have a nearly flat frequency distribution, and much longer plaintexts will then be required by the user.

Homophonic substitution



The forged nomenclator message used in the Babington Plot.

An early attempt to increase the difficulty of frequency analysis attacks on substitution ciphers was to disguise plaintext letter frequencies by *homophony*. In these ciphers, plaintext letters map to more than one ciphertext symbol. Usually, the highest-frequency plaintext symbols are given more equivalents than lower frequency letters. In this way, the frequency distribution is flattened, making analysis more difficult.

Since more than 26 characters will be required in the ciphertext alphabet, various solutions are employed to invent larger alphabets. Perhaps the simplest is to use a numeric substitution 'alphabet'. Another method consists of simple variations on the existing alphabet; uppercase, lowercase, upside down, etc. More artistically, though not necessarily more securely, some homophonic ciphers employed wholly invented alphabets of fanciful symbols.

An interesting variant is the **nomenclator**. Named after the public official who announced the titles of visiting dignitaries, this cipher combined a small codebook with large homophonic substitution tables. Originally the code was restricted to the names of important people, hence the name of the cipher; in later years it covered many common words and place names as well. The symbols for whole words (*codewords* in modern parlance) and letters (cipher in modern parlance) were not distinguished in the ciphertext. The Rossignols' Great Cipher used by Louis XIV of France was one; after it went out of use, messages in French archives were unbroken for several hundred years.

Nomenclators were the standard fare of diplomatic correspondence, espionage, and advanced political conspiracy from the early fifteenth century to the late eighteenth century; most conspirators were and have remained less cryptographically sophisticated. Although government intelligence cryptanalysts were systematically breaking nomenclators by the mid-sixteenth century, and superior systems had been available since 1467, the usual response to cryptanalysis was simply to make the tables larger. By the late eighteenth century, when the system was beginning to die out, some nomenclators had 50,000 symbols.

Nevertheless, not all nomenclators were broken; today, cryptanalysis of archived ciphertexts remains a fruitful area of historical research.

The Beale Ciphers are another example of a homophonic cipher. This is a fascinating story of buried treasure that was described in the 1819-21 period by use of a ciphered text that was keyed to the Declariation of Independence. Here each ciphertext character was represented by a number. The number was determined by taking the plaintext character and finding a word in the Declariation of Independence that started with that character and using the numerical position of that word in the Declaration of Independence as the encrypted form of that letter. Since many words in the Declaration of Independence start with the same letter, the encryption of that character could be any of the numbers associated with the words in the Declaration of Independence that start with that letter. Deciphering the encrypted text character X (which is a number) is as simple as looking up the the Xth word of the Declaration of Independence and using the first letter of that word as the decrypted character.

Another homophonic cipher was described by Stahl and was one of the first attempts to provide for computer security of data systems in computers through encryption. In Stahl's method, since plaintext and ciphertext were stored as binary strings of digits, he constructed the cipher in such a way that the number of homophones for a given character was in proportion to the frequency of the character, thus making frequency analysis much more difficult.

Polyalphabetic substitution

Polyalphabetic substitution ciphers were first described in 1467 by Leone Battista Alberti in the form of disks. Johannes Trithemius, in his book *Steganographia* (Ancient Greek for "hidden writing") introduced the now more standard form of a *tableau*. A more sophisticated version using mixed alphabets was described in 1563 by Giovanni Battista della Porta in his book, *De Furtivis Literarum Notis* (Latin for "On concealed characters in writing").

In a polyalphabetic cipher, multiple cipher alphabets are used. To facilitate encryption, all the alphabets are usually written out in a large table, traditionally called a *tableau*. The tableau is usually 26×26, so that 26 full ciphertext alphabets are available. The method of filling the tableau, and of choosing which alphabet to use next, defines the particular polyalphabetic cipher. All such ciphers are easier to break than once believed, as substitution alphabets are repeated for sufficiently large plaintexts.

One of the most popular was that of Blaise de Vigenère. First published in 1585, it was considered unbreakable until 1863, and indeed was commonly called *le chiffre indéchiffrable* (French for "indecipherable cipher").

In the Vigenère cipher, the first row of the tableau is filled out with a copy of the plaintext alphabet, and successive rows are simply shifted one place to the left. (Such a simple tableau is called a *tabula recta*, and mathematically corresponds to adding the plaintext and key letters, modulo 26.) A keyword is then used to choose which ciphertext alphabet to use. Each letter of the keyword is used in turn, and then they are repeated again from the beginning. So if the keyword is 'CAT', the first letter of plaintext is enciphered under alphabet 'C', the second under 'A', the third under 'T', the fourth under 'C' again, and so on. In practice, Vigenère keys were often phrases several words long.

In 1863, Friedrich Kasiski published a method (probably discovered secretly and independently before the Crimean War by Charles Babbage) which enabled the calculation of the length of the keyword in a Vigenère ciphered message. Once this was done, ciphertext letters that had been enciphered under the same alphabet could be picked out and attacked separately as a number of semi-independent simple substitutions - complicated by the fact that within one alphabet letters were separated and did not form complete words, but simplified by the fact that usually a *tabula recta* had been employed.

As such, even today a Vigenère type cipher should theoretically be difficult to break if mixed alphabets are used in the tableau, if the keyword is random, and if the total length

of ciphertext is less than 27.6 times the length of the keyword. These requirements are rarely understood in practice, and so Vigenère enciphered message security is usually less than might have been.

Other notable polyalphabetics include:

- The Gronsfeld cipher. This is identical to the Vigenère except that only 10 alphabets are used, and so the "keyword" is numerical.
- The Beaufort cipher. This is practically the same as the Vigenère, except the *tabula recta* is replaced by a backwards one, mathematically equivalent to ciphertext = key - plaintext. This operation is *self-inverse*, whereby the same table is used for both encryption and decryption.
- The autokey cipher, which mixes plaintext with a key to avoid periodicity.
- The running key cipher, where the key is made very long by using a passage from a book or similar text.

Modern stream ciphers can also be seen, from a sufficiently abstract perspective, to be a form of polyalphabetic cipher in which all the effort has gone into making the keystream as long and unpredictable as possible.

Polygraphic substitution

In a polygraphic substitution cipher, plaintext letters are substituted in larger groups, instead of substituting letters individually. The first advantage is that the frequency distribution is much flatter than that of individual letters (though not actually flat in real languages; for example, 'TH' is much more common than 'XQ' in English). Second, the larger number of symbols requires correspondingly more ciphertext to productively analyze letter frequencies.

To substitute *pairs* of letters would take a substitution alphabet 676 symbols long (26^2 = 676). In the same *De Furtivis Literarum Notis* mentioned above, della Porta actually proposed such a system, with a 20 x 20 tableau (for the 20 letters of the Italian/Latin alphabet he was using) filled with 400 unique glyphs. However the system was impractical and probably never actually used.

The earliest practical **digraphic cipher** (pairwise substitution), was the so-called Playfair cipher, invented by Sir Charles Wheatstone in 1854. In this cipher, a 5 x 5 grid is filled with the letters of a mixed alphabet (two letters, usually I and J, are combined). A digraphic substitution is then simulated by taking pairs of letters as two corners of a rectangle, and using the other two corners as the ciphertext. Special rules handle double letters and pairs falling in the same row or column. Playfair was in military use from the Boer War through World War II.

Several other practical polygraphics were introduced in 1901 by Felix Delastelle, including the bifid and four-square ciphers (both digraphic) and the trifid cipher (probably the first practical trigraphic).

The Hill cipher, invented in 1929 by Lester S. Hill, is a polygraphic substitution which can combine much larger groups of letters simultaneously using linear algebra. Each letter is treated as a digit in base 26: A = 0, B =1, and so on. (In a variation, 3 extra symbols are added to make the basis prime.) A block of n letters is then considered as a vector of n dimensions, and multiplied by a n x n matrix, modulo 26. The components of the matrix are the key, and should be random provided that the matrix is invertible in $\mathbb{Z}_{26}^{n}$(to ensure decryption is possible). A Hill cipher of dimension 6 was once implemented mechanically.

The Hill cipher is vulnerable to a known-plaintext attack because it is completely linear, so it must be combined with some non-linear step to defeat this attack. The combination of wider and wider weak, linear diffusive steps like a Hill cipher, with non-linear substitution steps, ultimately leads to a substitution-permutation network (e.g. a Feistel cipher), so it is possible — from this extreme perspective — to consider modern block ciphers as a type of polygraphic substitution.

Mechanical substitution ciphers

Between circa World War I and the widespread availability of computers (for some governments this was approximately the 1950s or 1960s; for other organizations it was a decade or more later; for individuals it was no earlier than 1975), mechanical implementations of polyalphabetic substitution ciphers were widely used. Several inventors had similar ideas about the same time, and rotor cipher machines were patented four times in 1919. The most important of the resulting machines was the Enigma, especially in the versions used by the German military from approximately 1930. The Allies also developed and used rotor machines (eg, SIGABA and Typex).

All of these were similar in that the substituted letter was chosen electrically from amongst the huge number of possible combinations resulting from the rotation of several letter disks. Since one or more of the disks rotated mechanically with each plaintext letter enciphered, the number of alphabets used was substantially more than astronomical. Early versions of these machine were, nevertheless, breakable. William F. Friedman of the US Army's SIS early found vulnerabilities in Hebern's rotor machine, and GC&CS's Dillwyn Knox solved versions of the Enigma machine (those without the "plugboard") well before WWII began. Traffic protected by essentially all of the German military Enigmas was broken by Allied cryptanalysts, most notably those at Bletchley Park, beginning with the German Army variant used in the early 1930s. This version was broken by inspired mathematical insight by Marian Rejewski in Poland.

No messages protected by the SIGABA and Typex machines were ever, so far as is publicly known, broken.

The one-time pad

One type of substitution cipher, the one-time pad, is quite special. It was invented near the end of WWI by Gilbert Vernam and Joseph Mauborgne in the US. It was

mathematically proven unbreakable by Claude Shannon, probably during WWII; his work was first published in the late 1940s. In its most common implementation, the one-time pad can be called a substitution cipher only from an unusual perspective; typically, the plaintext letter is combined (not substituted) in some manner (eg, XOR) with the key material character at that position.

The one-time pad is, in most cases, impractical as it requires that the key material be as long as the plaintext, *actually* random, used once and *only* once, and kept entirely secret from all except the sender and intended receiver. When these conditions are violated, even marginally, the one-time pad is no longer unbreakable. Soviet one-time pad messages sent from the US for a brief time during WWII used non-random key material. US cryptanalysts, beginning in the late 40s, were able to, entirely or partially, break a few thousand messages out of several hundred thousand.

In a mechanical implementation, rather like the ROCKEX equipment, the one-time pad was used for messages sent on the Moscow-Washington *hot line* established after the Cuban missile crisis.

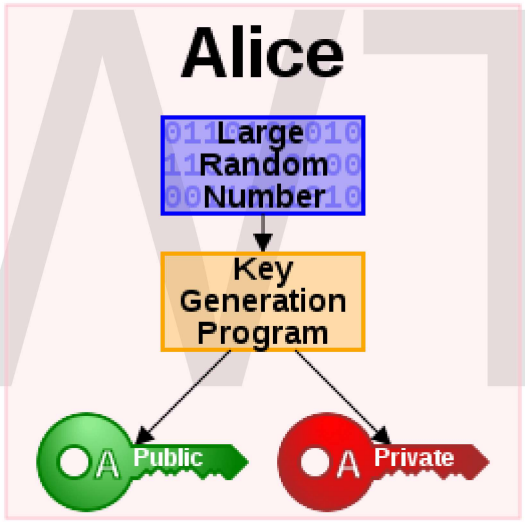
Substitution in modern cryptography

Substitution ciphers as discussed above, especially the older pencil-and-paper hand ciphers, are no longer in serious use. However, the cryptographic concept of substitution carries on even today. From a sufficiently abstract perspective, modern bit-oriented block ciphers (eg, DES, or AES) can be viewed as substitution ciphers on an enormously large binary alphabet. In addition, block ciphers often include smaller substitution tables called S-boxes.

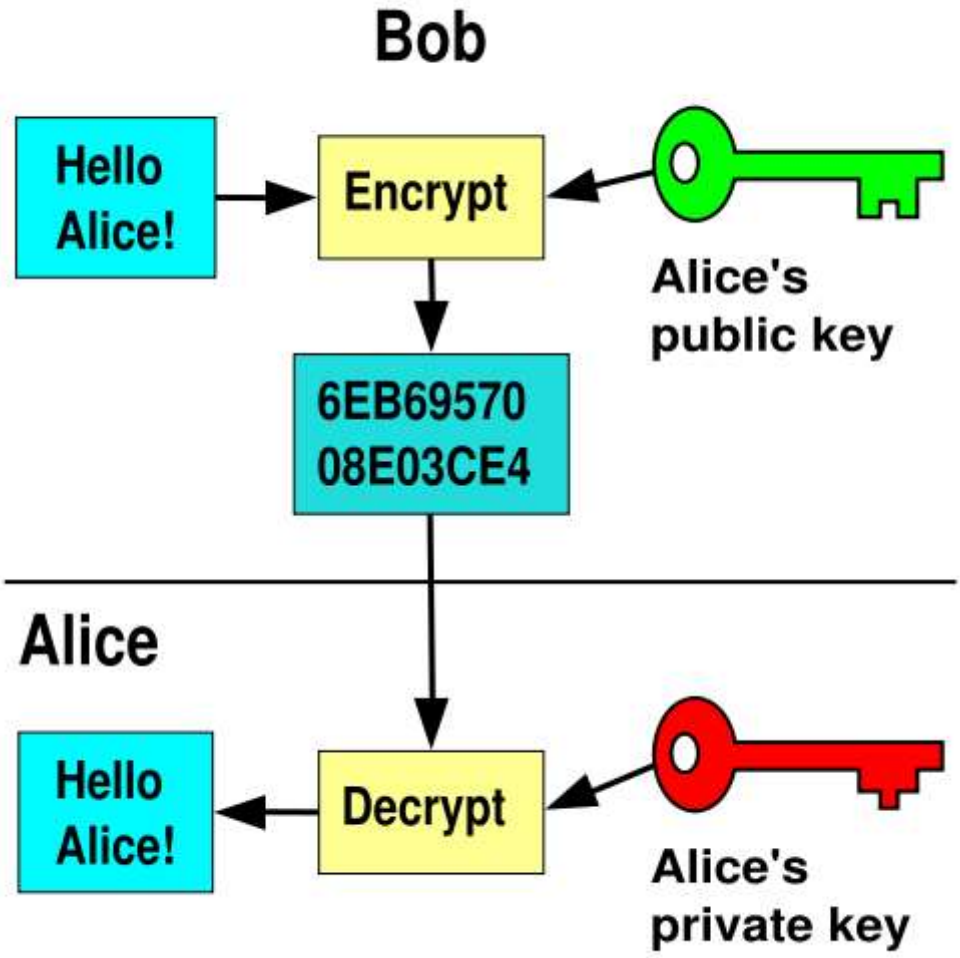
Chapter 4

Public-Key Cryptography and Cryptanalysis

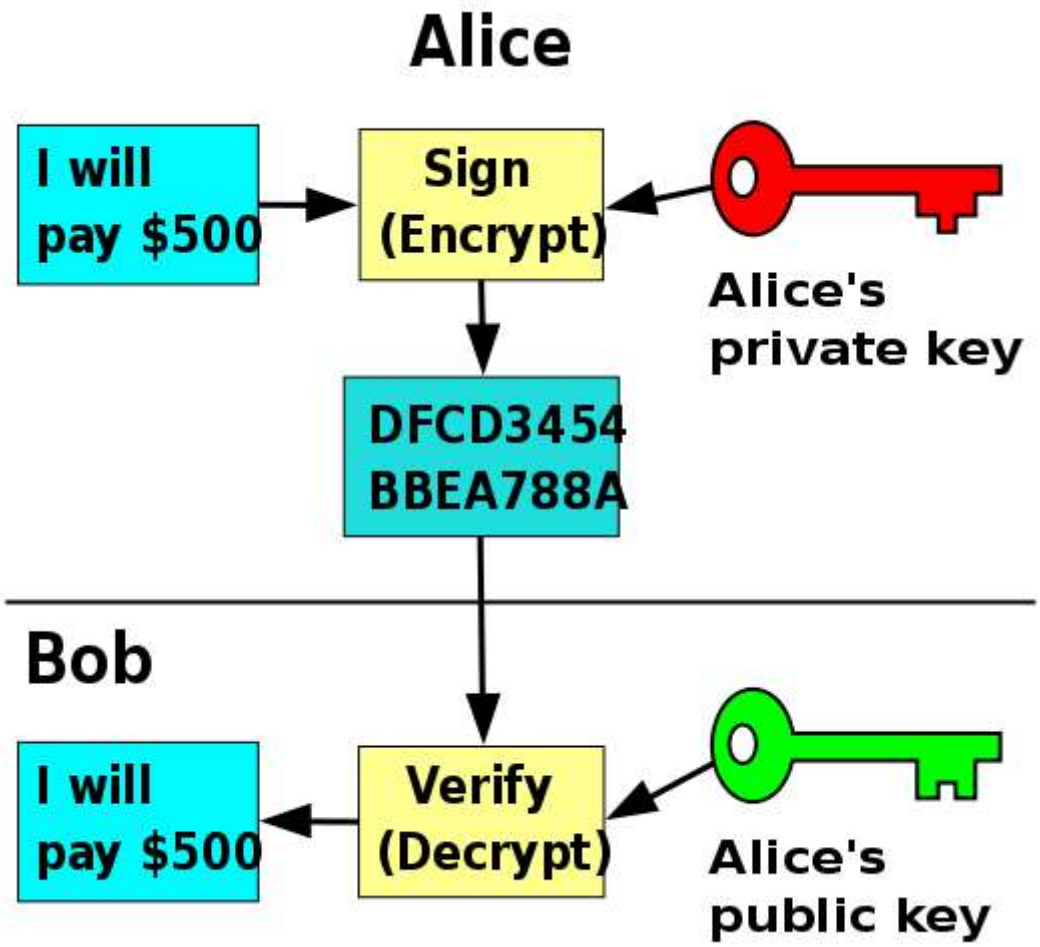
Public-key cryptography



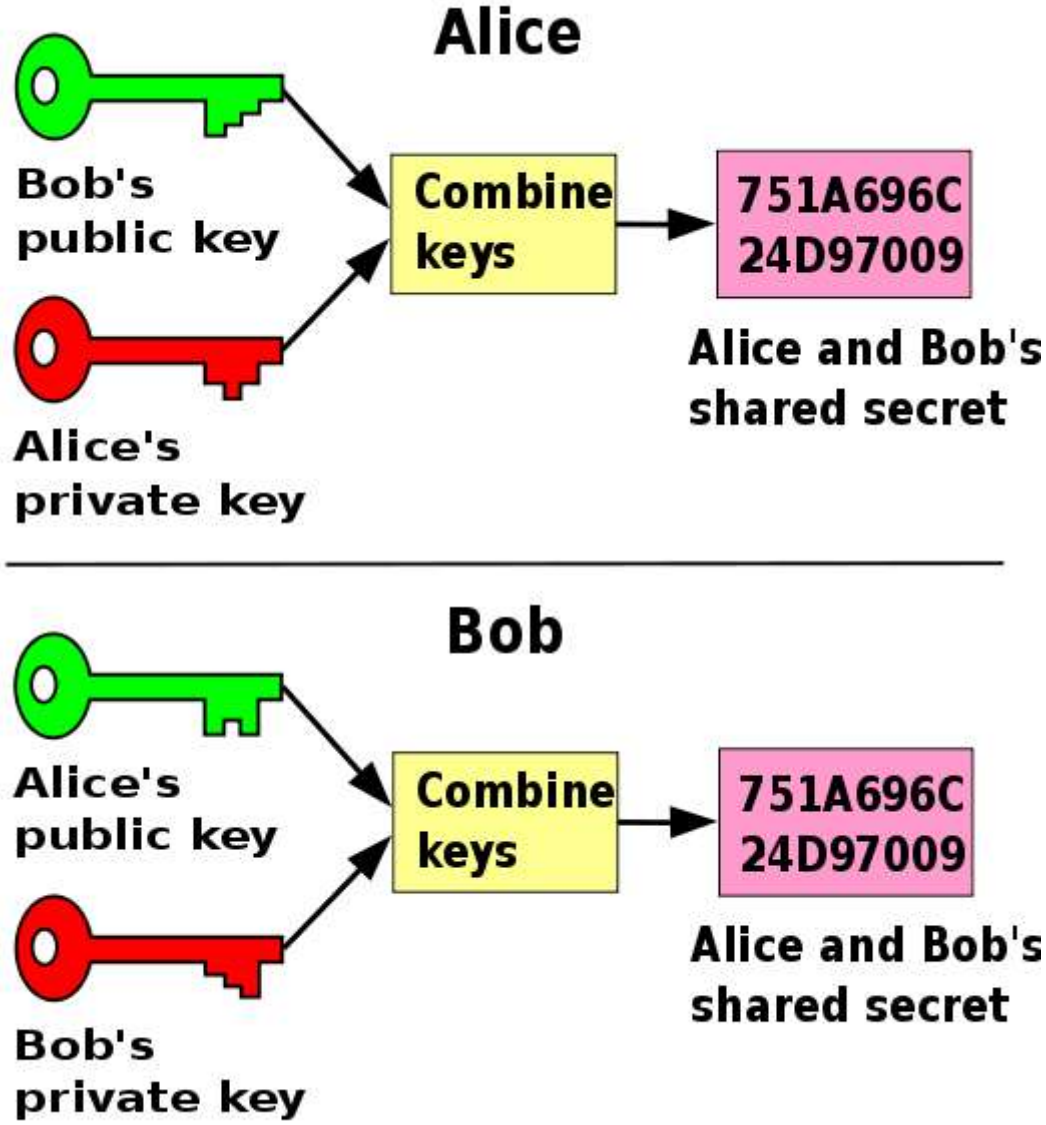
An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.



In 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 that private key.



In some related signature schemes, the private key is used to sign a message; but anyone can check the signature using the public key. Validity depends on private key security.



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 as the key for a symmetric cipher.

Public-key cryptography is a cryptographic approach which involves the use of asymmetric key algorithms instead of or in addition to symmetric key algorithms. Unlike symmetric key algorithms, it does not require a secure initial exchange of one or more secret keys to both sender and receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key. Use of these keys allows protection of the authenticity of a message by creating a digital signature of a message using the private key, which can be verified using the public key. It also allows protection of the confidentiality and integrity of a message, by public key

encryption, encrypting the message using the public key, which can only be decrypted using the private key.

Public key cryptography is a fundamental and widely used technology around the world. It is the approach which is employed by many cryptographic algorithms and cryptosystems. It underpins such Internet standards as Transport Layer Security (TLS) (successor to SSL), PGP, and GPG.

How it works

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**. The publicly available encrypting-key is widely distributed, while the private decrypting-key is known only to the recipient. Messages are encrypted with the recipient's public key and can *only* be decrypted with the corresponding private key. The keys are related mathematically, but the private key cannot feasibly (ie. in actual or projected practice) be derived from the public key. The discovery of algorithms that could produce public/private key pairs revolutionized the practice of cryptography beginning in the middle 1970s.

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 sender and 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, it is common to exchange a key using a key-exchange algorithm and transmit data using that key and a symmetric key algorithm. PGP, and the SSL/TLS family of schemes do this, for instance, and are thus called *hybrid cryptosystems*.

Description

The two main branches of public key cryptography are:

- Public key encryption: a message encrypted with a recipient's public key cannot be decrypted by anyone except a possessor of the matching private key—presumably, this will be the owner of that key and the person associated with the public key used. This is used for confidentiality.
- Digital signatures: a message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the sender had access to the private key (and therefore is likely to be the person associated with the public key used), and the part of the message that has not been tampered with. On the question of authenticity.

An analogy to public-key encryption is that of a locked mailbox 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 seal authenticates the sender.

A central problem for use of public-key cryptography is confidence (ideally proof) that a public key is correct, belongs to the person or entity claimed (i.e., is 'authentic'), 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. Another approach, used by PGP, is the "web of trust" method to ensure authenticity of key pairs.

History

During the early history of cryptography, two parties would agree upon a key using a secure, but non-cryptographic, method; for example, a face-to-face meeting or an exchange via a trusted courier. This key, which both parties kept absolutely secret, could then be used to exchange encrypted messages. A number of significant practical difficulties arise in 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 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*, written and published in the 1890s, William S. Jevons observed that 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, though he certainly did not invent the concept of public key cryptography.

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.

In 1997, it was publicly disclosed that asymmetric key algorithms were developed by James H. Ellis, Clifford Cocks, and Malcolm Williamson at the Government Communications Headquarters (GCHQ) in the UK in 1973. The researchers independently developed Diffie–Hellman key exchange and a special case of RSA. The GCHQ cryptographers referred to the technique as "non-secret encryption".

A generalization of Cocks' scheme was independently invented in 1977 by Rivest, Shamir and 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 large primes to encrypt and decrypt, performing both public key encryption and public key digital signature, and its security is connected to the presumed 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 as long as 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) difficulty of the discrete logarithm problem, as does the closely related DSA 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 equivalent estimated security.

Security

Some encryption schemes can be proven secure on the basis of the presumed hardness of a mathematical problem like 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.

In contrast to the one-time pad, no public-key encryption scheme has been shown to be secure against eavesdroppers with unlimited computational power. Proofs of security for asymmetric key cryptography therefore hold only with respect to computationally-limited adversaries, and can give guarantees (relative to particular mathematical assumptions) of

the form "the scheme cannot be broken using a desktop computer in 1000 years", or "this algorithm is secure if no improved method of (for instance, integer factoring) is found".

The most obvious application of a **public key encryption** system is confidentiality; a message which a sender encrypts using the recipient's public key can be decrypted only by the recipient's paired private key (assuming, 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 of 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 only be computed with the knowledge of a private key. To verify that a message has been signed by a user and has not been modified the receiver only needs to know the corresponding public key. In some cases (e.g. RSA) there exist digital signature schemes with many similarities to encryption schemes. In other cases (e.g. DSA) the algorithm does not resemble any encryption scheme.

To achieve authentication, *and* confidentiality, the sender could first sign the message using his private key, then encrypt the message and signature using the recipient's public key.

These characteristics can be used to construct many other, sometimes surprising, cryptographic protocols and applications, like digital cash, password-authenticated key agreement, multi-party key agreement, time stamping service, non-repudiation protocols, etc.

Practical considerations

A postal analogy

An analogy which can be used to understand the advantages of an asymmetric system is to imagine two people, Alice and Bob, 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 the example, a corrupt postal worker) from copying a key while it is in transit, allowing said 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.

In another kind of asymmetric key system, 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 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`. A simple `XOR` with the individual keys is such a commutative cipher. For example, let $E_1()$ and $E_2()$ be two encryption functions and let "`M`" be the message so 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 $E_1()$. She'll 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"), but the lock is such that it is impossible (so far as is known) to deduce from these instructions how to make a key which will open that lock ("private key"). Those wishing to send messages to Bob use the public key to encrypt the message; Bob uses his private key to decrypt it.

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. Unfortunately, there is no public-key scheme with this property, since all public-key schemes are susceptible to brute force key search attack. Such attacks are impractical if the amount of computation needed to succeed (termed 'work factor' by Claude Shannon) is out of reach of potential attackers. In many cases, the work factor can be increased by simply choosing a longer key. But other attacks may have much lower work factors, making resistance to brute force attack irrelevant, and some are known for some public key encryption algorithms. Both RSA and ElGamal encryption have known attacks which are much faster than the brute force approach. Such estimates have changed both with the decreasing cost of computer power, and new mathematical discoveries.

In practice, these insecurities can be generally avoided by choosing key sizes large enough that the best known attack would take so long that it is not worth any adversary's time and money to break the code. For example, if an estimate of how long it takes to break an encryption scheme is one thousand years, and it were used to encrypt your credit card details, they would be safe enough, since the time needed to decrypt the details will be rather longer than the useful life of those details, which expire after a few years. Typically, the key size needed is much longer for public key algorithms than for symmetric key algorithms.

Aside from the resistance to attack of a particular keypair, 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 when a new attack was found. 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. Thus, mere use of asymmetric key algorithms does not ensure security; it is an area of active research to discover and protect against new attacks.

Another potential security vulnerability in using asymmetric keys is the possibility of a man-in-the-middle attack, in which communication of public keys is intercepted by a third party and 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 to avoid suspicion. This attack may seem to be difficult to implement in practice, but it's not impossible when using insecure media (e.g. public networks such as the Internet or wireless communications). A malicious staff member at Alice or Bob's ISP might find it quite easy to carry out.

One approach to prevent such attacks is the use of a certificate authority, a trusted third party responsible for verifying the identity of a user of the system and issuing a tamper resistant and non-spoofable digital certificate for participants. Such certificates are signed data blocks stating that this public key belongs to that person, company or other entity. This approach also has weaknesses. For example, the certificate authority issuing the certificate must be trusted to have properly checked the identity of the key-holder, the correctness of the public key when it issues a certificate, and has made arrangements with all participants to check all certificates before protected communications can begin. Web browsers, for instance, are supplied with many self-signed identity certificates from PKI providers; these are used to check certificate's bonafides (issued properly by the claimed central PKI server?) and then, in a second step, the certificate of a potential communicant. An attacker who could subvert the certificate authority 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 problems, this approach is widely used; examples include SSL and its successor, TLS, which are commonly used to provide security in web browsers, for example, to securely send credit card details to an online store.

Computational cost

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 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, lest the algorithm function perfectly and yet be entirely insecure in practice. As with most cryptography, 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, either in the form of a hierarchical certificate authority (e.g., X.509), a local trust model (e.g., SPKI), or a web of trust scheme (e.g., 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 classification!

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 which 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 which 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 which include an expiry date are unsatisfactory in that the expiry 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. automatically by exceeding its use-before date, or less so, 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 essentially 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 revoke them in the hands of each user, and 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 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 obviously 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 both authority to revoke and to 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. It 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 two means of spreading information (e.g., a key revocation here) in a distributed system: either the information is pushed to users from a central point(s), or it is pulled from a central point(s) to 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 that all participants will actually receive the message, and if the number of participants is large and some of their physical or network distance great, the probability of complete success (which is, ideally, 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'. In other words, 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 those systems which 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 expiry date for each of the verification sources. How long this timeout should be is a decision which embodies a trade-off between availability and security that will have to be decided in advance, at system design time.

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 with T.

Such a compromise has two implications. 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 about who, where, when, etc of the events leading up to 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 and under what conditions to revoke a public key certificate, how to spread the revocation, but also, 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.

Such a recovery procedure can be quite complex, and while it is in progress the system will likely be vulnerable against Denial of Service attacks, among other things.

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

Examples of asymmetric key algorithms not widely adopted include:

- NTRUEncrypt cryptosystem
- McEliece cryptosystem

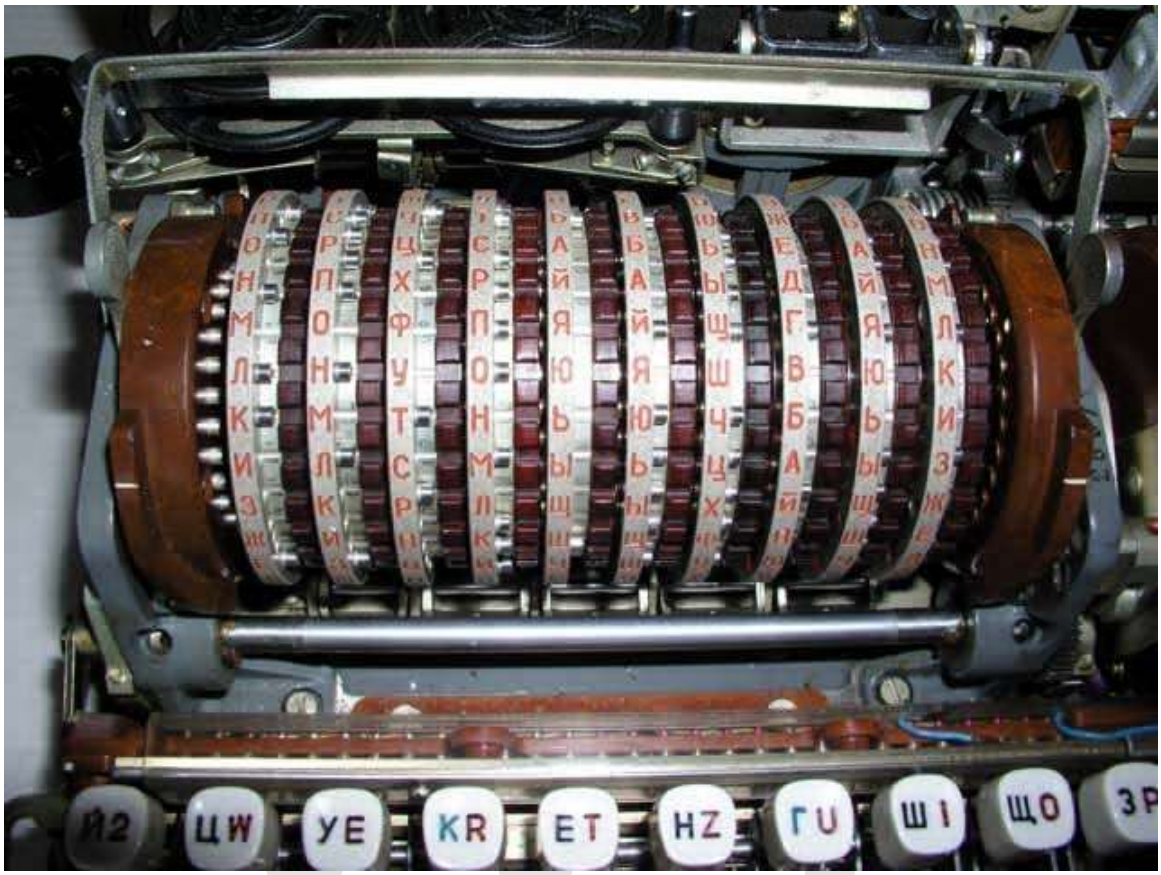
Examples of notable yet insecure asymmetric key algorithms include:

- 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 implemented as an IETF standard TLS
- SILC
- SSH.

Cryptanalysis



Close-up of the rotors in a Fialka cipher machine

Cryptanalysis (from the Greek *kryptós*, "hidden", and *analýein*, "to loosen" or "to untie") is the study of methods for obtaining the meaning of encrypted information, without access to the secret information that is normally required to do so. Typically, this involves knowing how the system works and finding a secret key. In non-technical language, this is the practice of **codebreaking** or **cracking the code**, although these phrases also have a specialised technical meaning.

"Cryptanalysis" is also used to refer to any attempt to circumvent the security of other types of cryptographic algorithms and protocols in general, and not just encryption. However, cryptanalysis usually excludes methods of attack that do not primarily target weaknesses in the actual cryptography, such as bribery, physical coercion, burglary, keystroke logging, and social engineering, although these types of attack are an important concern and are often more effective than traditional cryptanalysis.

Even though the goal has been the same, the methods and techniques of cryptanalysis have changed drastically through the history of cryptography, adapting to increasing cryptographic complexity, ranging from the pen-and-paper methods of the past, through

machines like Bombes and Colossus computers in World War II, to the computer-based schemes of the present. The results of cryptanalysis have also changed — it is no longer possible to have unlimited success in codebreaking, and there is a hierarchical classification of what constitutes an attack. In the mid-1970s, a new class of cryptography was introduced: asymmetric cryptography. Methods for breaking these cryptosystems are typically radically different from before, and usually involve solving carefully-constructed problems in pure mathematics, the best-known being integer factorization.

History of cryptanalysis

Cryptanalysis has coevolved together with cryptography, and the contest can be traced through the history of cryptography—new ciphers being designed to replace old broken designs, and new cryptanalytic techniques invented to crack the improved schemes. In practice, they are viewed as two sides of the same coin: in order to create secure cryptography, you have to design against possible cryptanalysis.

Classical cryptanalysis

First page of Al-Kindi's 9th century *Manuscript on Deciphering Cryptographic Messages*

Although the actual word "*cryptanalysis*" is relatively recent (it was coined by William Friedman in 1920), methods for breaking codes and ciphers are much older. The first known recorded explanation of cryptanalysis was given by 9th-century Arabian polymath, Al-Kindi (also known as "Alkindus" in Europe), in *A Manuscript on Deciphering Cryptographic Messages*. This treatise includes a description of the method of frequency analysis (Ibrahim Al-Kadi, 1992- ref-3). Italian scholar Giambattista della Porta was author of a seminal work on cryptanalysis *"De Furtivis Literarum Notis"*.

Frequency analysis is the basic tool for breaking most classical ciphers. In natural languages, certain letters of the alphabet appear more frequently than others; in English, "E" is likely to be the most common letter in any sample of plaintext. Similarly, the digraph "TH" is the most likely pair of letters in English, and so on. Frequency analysis relies on a cipher failing to hide these statistics. For example, in a simple substitution cipher (where each letter is simply replaced with another), the most frequent letter in the ciphertext would be a likely candidate for "E". Frequency analysis of such a cipher is therefore relatively easy, provided that the ciphertext is long enough to give a reasonably representative count of the letters of the alphabet that it contains.

In Europe during the 15th and 16th centuries, the idea of a polyalphabetic substitution cipher was developed, among others by the French diplomat Blaise de Vigenère (1523–96). For some three centuries, the Vigenère cipher, which uses a repeating key to select different encryption alphabets in rotation, was considered to be completely secure (*le chiffre indéchiffrable*—"the indecipherable cipher"). Nevertheless, Charles Babbage (1791–1871) and later, independently, Friedrich Kasiski (1805–81) succeeded in breaking this cipher. During World War I, inventors in several countries developed rotor cipher machines such as Arthur Scherbius' Enigma, in an attempt to minimise the repetition that had been exploited to break the Vigenère system.

In practice, frequency analysis relies as much on linguistic knowledge as it does on statistics, but as ciphers became more complex, mathematics became more important in cryptanalysis. This change was particularly evident before and during World War II, where efforts to crack Axis ciphers required new levels of mathematical sophistication. Moreover, automation was first applied to cryptanalysis in that era with the Polish Bomba device, the British Bombe development of it, the use of punched card equipment, and in the Colossus computers — the first electronic digital computers to be controlled by a program.

Depth

Sending two or more messages with the same key is an insecure process. To a cryptanalyst the messages are then said to be ***"in depth"***. This may be detected by the messages having the same *indicator* by which the sending operator informs the receiving operator, about the key for the message. In a symmetrical cipher the same key that was applied to the plaintext to produce the ciphertext, is applied to the ciphertext to recover the plaintext. An example of such a system is the Vernam cipher in which the key is combined with the plaintext or ciphertext using the "XOR" operator (symbolised by $\oplus$). So:

$$\text{Plaintext} \oplus \text{Key} = \text{Ciphertext}$$

and:

$$\text{Ciphertext} \oplus \text{Key} = \text{Plaintext}$$

This also means that when there is a depth, combining the two ciphertexts eliminates the common key leaving the combination of the two plaintexts.

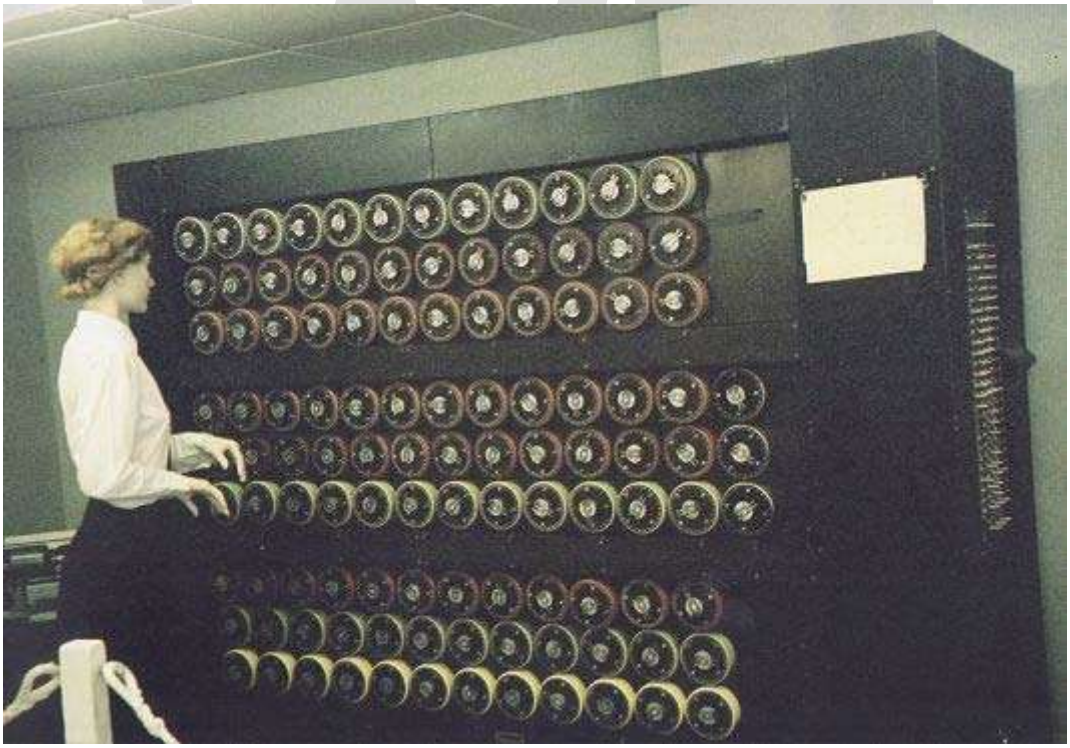
$$\text{Ciphertext1} \oplus \text{Ciphertext2} = \text{Plaintext1} \oplus \text{Plaintext2}$$

Conventional hand cryptographic methods can then be used to work out the individual plaintexts. When these are then combined with the ciphertext, the key is revealed.

$$\text{Plaintext} \oplus \text{Ciphertext} = \text{Key}$$

Knowledge of keys from a cipher may allow cryptanalysts to work out the system used for constructing them.

Modern cryptanalysis



The Bombe replicated the action of several Enigma machines wired together. Each of the rapidly rotating drums, pictured above in a Bletchley Park museum mockup, simulated the action of an Enigma rotor.

Even though computation was used to great effect in cryptanalysis of the Enigma and other systems during World War II, it also made possible new methods of cryptography orders of magnitude more complex than ever before. Taken as a whole, modern

cryptography has become much more impervious to cryptanalysis than the pen-and-paper systems of the past, and now seems to have the upper hand against pure cryptanalysis. The historian David Kahn notes,

"Many are the cryptosystems offered by the hundreds of commercial vendors today that cannot be broken by any known methods of cryptanalysis. Indeed, in such systems even a chosen plaintext attack, in which a selected plaintext is matched against its ciphertext, cannot yield the key that unlock[s] other messages. In a sense, then, cryptanalysis is dead. But that is not the end of the story. Cryptanalysis may be dead, but there is - to mix my metaphors - more than one way to skin a cat.".

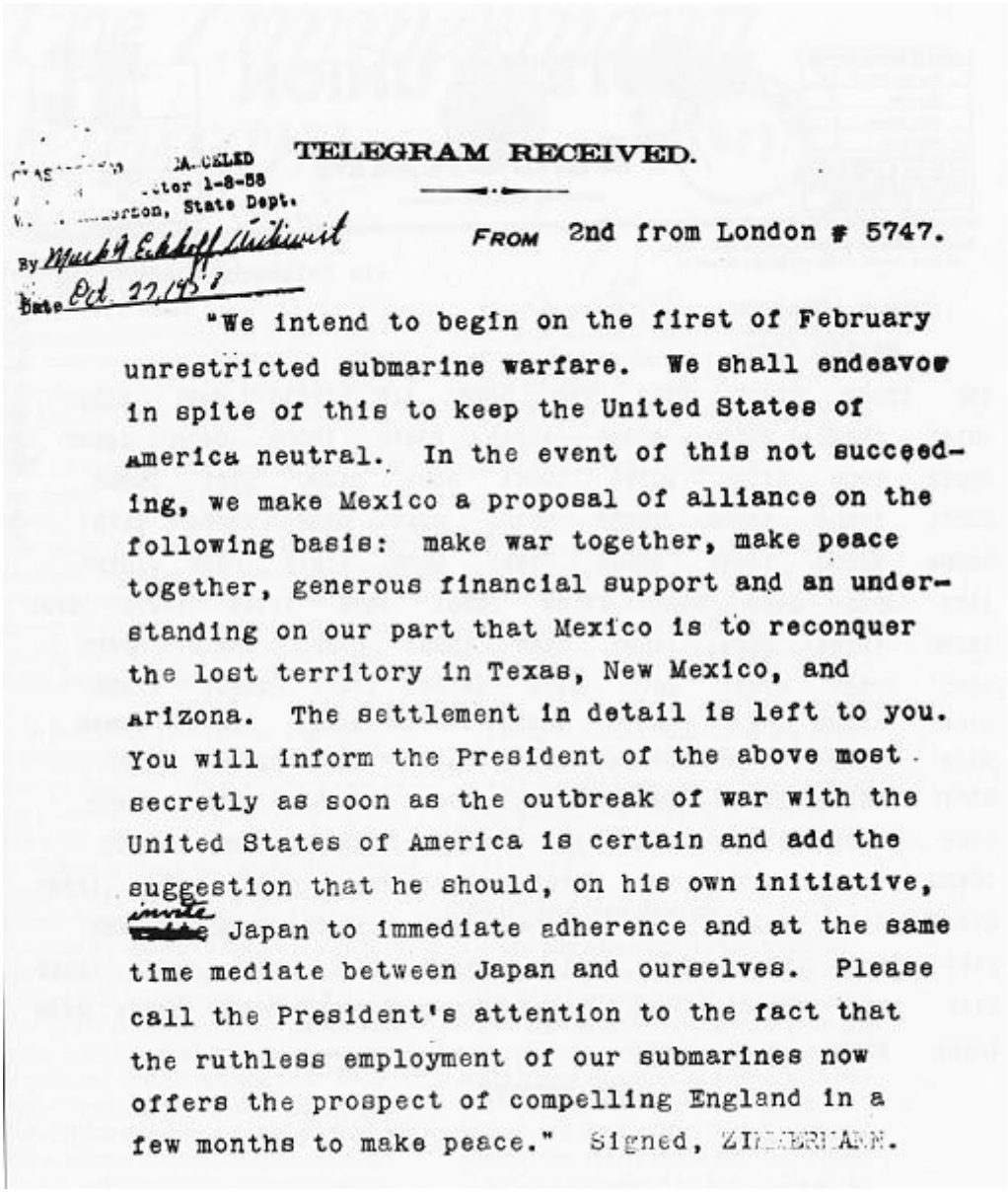
Kahn goes on to mention increased opportunities for interception, bugging, side channel attacks, and quantum computers as replacements for the traditional means of cryptanalysis. In 2010, former NSA technical director Brian Snow said that both academic and government cryptographers are "moving very slowly forward in a mature field."

However, any postmortems for cryptanalysis may be premature. While the effectiveness of cryptanalytic methods employed by intelligence agencies remains unknown, many serious attacks against both academic and practical cryptographic primitives have been published in the modern era of computer cryptography:

- The block cipher Madryga, proposed in 1984 but not widely used, was found to be susceptible to ciphertext-only attacks in 1998.
- FEAL-4, proposed as a replacement for the DES standard encryption algorithm but not widely used, was demolished by a spate of attacks from the academic community, many of which are entirely practical.
- The A5/1, A5/2, CMEA, and DECT systems used in mobile and wireless phone technology can all be broken in hours, minutes or even in real-time using widely-available computing equipment.
- Brute-force keyspace search has broken some real-world ciphers and applications, including single-DES, 40-bit "export-strength" cryptography, and the DVD Content Scrambling System.
- In 2001, Wired Equivalent Privacy (WEP), a protocol used to secure Wi-Fi wireless networks, was shown to be breakable in practice because of a weakness in the RC4 cipher and aspects of the WEP design that made related-key attacks practical. WEP was later replaced by Wi-Fi Protected Access.
- In 2008, researchers conducted a proof-of-concept break of SSL using weaknesses in the MD5 hash function and certificate issuer practices that made it possible to exploit collision attacks on hash functions. The certificate issuers involved changed their practices to prevent the attack from being repeated.

Thus, while the best modern ciphers may be far more resistant to cryptanalysis than the Enigma, cryptanalysis and the broader field of information security remain quite active.

The results of cryptanalysis



TELEGRAM RECEIVED.

FROM 2nd from London # 5747.

"We intend to begin on the first of February unrestricted submarine warfare. We shall endeavor in spite of this to keep the United States of America neutral. In the event of this not succeeding, we make Mexico a proposal of alliance on the following basis: make war together, make peace together, generous financial support and an understanding on our part that Mexico is to reconquer the lost territory in Texas, New Mexico, and Arizona. The settlement in detail is left to you. You will inform the President of the above most secretly as soon as the outbreak of war with the United States of America is certain and add the suggestion that he should, on his own initiative, invite Japan to immediate adherence and at the same time mediate between Japan and ourselves. Please call the President's attention to the fact that the ruthless employment of our submarines now offers the prospect of compelling England in a few months to make peace." Signed, ZIMMERMANN.

The decrypted Zimmermann Telegram.

Successful cryptanalysis has undoubtedly influenced history; the ability to read the presumed-secret thoughts and plans of others can be a decisive advantage. For example, in England in 1587, Mary, Queen of Scots was tried and executed for treason for her involvement in three plots to assassinate Elizabeth I of England which were known about because her coded correspondence with fellow conspirators had been deciphered by Thomas Phelippes; in World War I, the breaking of the Zimmermann Telegram was instrumental in bringing the United States into the war; in World War II, the cryptanalysis of the German ciphers — including the Enigma machine and the Lorenz cipher — has been credited with everything between shortening the end of the European

war by a few months to determining the eventual result. The United States also benefited from the cryptanalysis of the Japanese Purple code.

Governments have long recognized the potential benefits of cryptanalysis for intelligence, both military and diplomatic, and established dedicated organizations devoted to breaking the codes and ciphers of other nations, for example, GCHQ and the NSA, organizations which are still very active today. In 2004, it was reported that the United States had broken Iranian ciphers. (It is unknown, however, whether this was pure cryptanalysis, or whether other factors were involved:).

Types of cryptanalytic attack

Cryptanalytic attacks vary in potency and how much of a threat they pose to real-world cryptosystems. A *certificational weakness* is a theoretical attack that is unlikely to be applicable in any real-world situation; the majority of results found in modern cryptanalytic research are of this type. Essentially, the practical importance of an attack is dependent on the answers to the following four questions:

1. What knowledge and capabilities does the attacker need?
2. How much additional secret information is deduced?
3. How much computation is required? (What is the computational complexity?)
4. Does the attack break the full cryptosystem, or only a weakened version?

Access needed for the attack

Cryptanalysis can be performed under a number of assumptions about how much access the attacker has to the system under attack. As a basic starting point it is normally assumed that, for the purposes of analysis, the general algorithm is known; this is Kerckhoffs' principle of "the enemy knows the system". This is a reasonable assumption in practice — throughout history, there are countless examples of secret algorithms falling into wider knowledge, variously through espionage, betrayal and reverse engineering. (On occasion, ciphers have been reconstructed through pure deduction; for example, the German Lorenz cipher and the Japanese Purple code, and a variety of classical schemes).

Other assumptions include:

- *Ciphertext-only*: the cryptanalyst has access only to a collection of ciphertexts or codetexts.
- *Known-plaintext*: the attacker has a set of ciphertexts to which he knows the corresponding plaintext.
- *Chosen-plaintext* (*chosen-ciphertext*): the attacker can obtain the ciphertexts (plaintexts) corresponding to an arbitrary set of plaintexts (ciphertexts) of his own choosing.

- *Adaptive chosen-plaintext*: like a chosen-plaintext attack, except the attacker can choose subsequent plaintexts based on information learned from previous encryptions. Similarly *Adaptive chosen ciphertext attack*.
- *Related-key attack*: Like a chosen-plaintext attack, except the attacker can obtain ciphertexts encrypted under two different keys. The keys are unknown, but the relationship between them is known; for example, two keys that differ in the one bit.

These types of attack clearly differ in how plausible they would be to mount in practice. Although some are more likely than others, cryptographers will often take a conservative approach to security and assume the worst-case when designing algorithms, reasoning that if a scheme is secure even against unrealistic threats, then it should also resist real-world cryptanalysis as well.

The assumptions are often more realistic than they might seem upon first glance. For a known-plaintext attack, the cryptanalyst might well know or be able to guess at a likely part of the plaintext, such as an encrypted letter beginning with "Dear Sir", or a computer session starting with "`LOGIN:`". A chosen-plaintext attack is less likely, but it is sometimes plausible: for example, you could convince someone to forward a message you have given them, but in encrypted form. Related-key attacks are mostly theoretical, although they can be realistic in certain situations, for example, when constructing cryptographic hash functions using a block cipher.

Usefulness of attack results

The results of cryptanalysis can also vary in usefulness. For example, cryptographer Lars Knudsen (1998) classified various types of attack on block ciphers according to the amount and quality of secret information that was discovered:

- *Total break* — the attacker deduces the secret key.
- *Global deduction* — the attacker discovers a functionally equivalent algorithm for encryption and decryption, but without learning the key.
- *Instance (local) deduction* — the attacker discovers additional plaintexts (or ciphertexts) not previously known.
- *Information deduction* — the attacker gains some Shannon information about plaintexts (or ciphertexts) not previously known.
- *Distinguishing algorithm* — the attacker can distinguish the cipher from a random permutation.

Similar considerations apply to attacks on other types of cryptographic algorithm.

Computational resources required

Attacks can also be characterised by the resources they require. Those resources include:

- Time — the number of *computation steps* (like encryptions) which must be performed.
- Memory — the amount of *storage* required to perform the attack.
- Data — the quantity of *plaintexts and ciphertexts* required.

It's sometimes difficult to predict these quantities precisely, especially when the attack isn't practical to actually implement for testing. But academic cryptanalysts tend to provide at least the estimated *order of magnitude* of their attacks' difficulty, saying, for example, "SHA-1 collisions now 2^{52}"

Bruce Schneier notes that even computationally impractical attacks can be considered breaks: "Breaking a cipher simply means finding a weakness in the cipher that can be exploited with a complexity less than brute force. Never mind that brute-force might require 2^{128} encryptions; an attack requiring 2^{110} encryptions would be considered a break...simply put, a break can just be a certificational weakness: evidence that the cipher does not perform as advertised." (Schneier, 2000).

Partial breaks

Academic attacks are often against weakened versions of a cryptosystem, such as a block cipher or hash function with some rounds removed. Many, but not all, attacks become exponentially more difficult to execute as rounds are added to a cryptosystem, so it's possible for the full cryptosystem to be strong even through reduced-round variants are weak. Nonetheless, partial breaks that come close to breaking the original cryptosystem may mean that a full break will follow; the successful attacks on DES, MD5, and SHA-1 were all preceded by attacks on weakened versions.

Academic weakness versus practical weakness

In academic cryptography, a *weakness* or a *break* in a scheme is usually defined quite conservatively: it might require impractical amounts of time, memory, or known plaintexts. It also might require the attacker be able to do things many real-world attackers can't: for example, the attacker may need to choose particular plaintexts to be encrypted or even to ask for plaintexts to be encrypted using several keys related to the secret key. Furthermore, it might only reveal a small amount of information, enough to prove the cryptosystem imperfect but too little to be useful to real-world attackers. Finally, an attack might only apply to a weakened version of cryptographic tools, like a reduced-round block cipher, as a step towards breaking of the full system.

Cryptanalysis of asymmetric cryptography

Asymmetric cryptography (or public key cryptography) is cryptography that relies on using two keys; one private, and one public. Such ciphers invariably rely on "hard" mathematical problems as the basis of their security, so an obvious point of attack is to develop methods for solving the problem. The security of two-key cryptography depends

on mathematical questions in a way that single-key cryptography generally does not, and conversely links cryptanalysis to wider mathematical research in a new way.

Asymmetric schemes are designed around the (conjectured) difficulty of solving various mathematical problems. If an improved algorithm can be found to solve the problem, then the system is weakened. For example, the security of the Diffie-Hellman key exchange scheme depends on the difficulty of calculating the discrete logarithm. In 1983, Don Coppersmith found a faster way to find discrete logarithms (in certain groups), and thereby requiring cryptographers to use larger groups (or different types of groups). RSA's security depends (in part) upon the difficulty of integer factorization — a breakthrough in factoring would impact the security of RSA.

In 1980, one could factor a difficult 50-digit number at an expense of 10^{12} elementary computer operations. By 1984 the state of the art in factoring algorithms had advanced to a point where a 75-digit number could be factored in 10^{12} operations. Advances in computing technology also meant that the operations could be performed much faster, too. Moore's law predicts that computer speeds will continue to increase. Factoring techniques may continue do so as well, but will most likely depend on mathematical insight and creativity, neither of which has ever been successfully predictable. 150-digit numbers of the kind once used in RSA have been factored. The effort was greater than above, but was not unreasonable on fast modern computers. By the start of the 21st century, 150-digit numbers were no longer considered a large enough key size for RSA. Numbers with several hundred digits were still considered too hard to factor in 2005, though methods will probably continue to improve over time, requiring key size to keep pace or other methods such as elliptic curve cryptography to be used.

Another distinguishing feature of asymmetric schemes is that, unlike attacks on symmetric cryptosystems, any cryptanalysis has the opportunity to make use of knowledge gained from the public key.

Quantum computing applications for cryptanalysis

Quantum computers, which are still in the early phases of research, have potential use in cryptanalysis. For example, Shor's Algorithm could factor large numbers in polynomial time, in effect breaking some commonly used forms of public-key encryption.

By using Grover's algorithm on a quantum computer, brute-force key search can be made quadratically faster. However, this could be countered by doubling the key length.

Chapter 5

Cipher

In cryptography, a **cipher** (or **cypher**) is an algorithm for performing encryption or decryption — a series of well-defined steps that can be followed as a procedure. An alternative, less common term is **encipherment**. In non-technical usage, a "cipher" is the same thing as a "code"; however, the concepts are distinct in cryptography. In classical cryptography, ciphers were distinguished from codes. Codes operated by substituting according to a large codebook which linked a random string of characters or numbers to a word or phrase. For example, "UQJHSE" could be the code for "Proceed to the following coordinates". When using a cipher the original information is known as plaintext, and the encrypted form as **ciphertext**. The ciphertext message contains all the information of the plaintext message, but is not in a format readable by a human or computer without the proper mechanism to decrypt it; it should resemble random gibberish to those not intended to read it.

The operation of a cipher usually depends on a piece of auxiliary information, called a key or, in traditional NSA parlance, a **cryptovariable.** The encrypting procedure is varied depending on the key, which changes the detailed operation of the algorithm. A key must be selected before using a cipher to encrypt a message. Without knowledge of the key, it should be difficult, if not nearly impossible, to decrypt the resulting ciphertext into readable plaintext.

Most modern ciphers can be categorized in several ways

- By whether they work on blocks of symbols usually of a fixed size (block ciphers), or on a continuous stream of symbols (stream ciphers).
- By whether the same key is used for both encryption and decryption (symmetric key algorithms), or if a different key is used for each (asymmetric key algorithms). If the algorithm is symmetric, the key must be known to the recipient and sender and to no one else. If the algorithm is an asymmetric one, the enciphering key is different from, but closely related to, the deciphering key. If one key cannot be deduced from the other, the asymmetric key algorithm has the public/private key property and one of the keys may be made public without loss of confidentiality.

Etymology of "Cipher"

"Cipher" is alternatively spelled "cypher"; similarly "ciphertext" and "cyphertext", and so forth.

The word "cipher" in former times meant "zero" and had the same origin: Middle French as *cifre* and Medieval Latin as *cifra,* from the Arabic صِفْر *ṣifr* = zero. "Cipher" was later used for any decimal digit, even any number. There are many theories about how the word "cipher" may have come to mean "encoding":

- Encoding often involved numbers.
- The Roman number system was very cumbersome because there was no concept of zero (or empty space). The concept of zero (which was also called "cipher"), which we all now think of as natural, was very alien in medieval Europe, so confusing and ambiguous to common Europeans that in arguments people would say "talk clearly and not so far fetched as a cipher". Cipher came to mean concealment of clear messages or encryption.
    - The French formed the word "chiffre" and adopted the Italian word "zero".
    - The English used "zero" for "0", and "cipher" from the word "ciphering" as a means of computing.
    - The Germans used the words "Ziffer" (digit) and "Chiffre".
    - The Dutch still use the word "cijfer" to refer to a number.
    - The Italians also use the word "cifra" to refer to a number.

Dr. Al-Kadi concluded that the Arabic word *sifr*, for the digit zero, developed into the European technical term for encryption.

Code (cryptography)

In cryptography, a **code** is a method used to transform a message into an obscured form, preventing those who do not possess special information, or key, required to apply the transform from understanding what is actually transmitted. The usual method is to use a *codebook* with a list of common phrases or words matched with a *codeword*. Encoded messages are sometimes termed *codetext*, while the original message is usually referred to as plaintext.

Terms like *code* and *in code* are often used to refer to any form of encryption. However, there is an important distinction between codes and ciphers in technical work; it is, essentially, the scope of the transformation involved. Codes operate at the level of meaning; that is, words or phrases are converted into something else. Ciphers work at the level of individual letters, or small groups of letters, or even, in modern ciphers, with individual bits. While a code might transform "change" into "CVGDK" or "cocktail lounge", a cipher transforms elements below the semantic level, i.e., below the level of

meaning. The "a" in "attack" might be converted to "Q", the first "t" to "f", the second "t" to "3", and so on. Ciphers are more convenient than codes in some situations, there being no need for a codebook, with its inherently limited number of valid messages, and the possibility of fast automatic operation on computers.

Codes were long believed to be more secure than ciphers, since (if the compiler of the codebook did a good job) there is no pattern of transformation which can be discovered, whereas ciphers use a consistent transformation, which can potentially be identified and reversed (except in the case of the one-time pad).

One- and two-part codes

Codes are defined by "codebooks" (physical or notional), which are dictionaries of codegroups listed with their corresponding plaintext. Codes originally had the codegroups assigned in 'plaintext order' for convenience of the code designed, or the encoder. For example, in a code using numeric code groups, a plaintext word starting with "a" would have a low-value group, while one starting with "z" would have a high-value group. The same codebook could be used to "encode" a plaintext message into a coded message or "codetext", and "decode" a codetext back into plaintext message.

However, such "one-part" codes had a certain predictability that made it easier for others to notice patterns and "crack" or "break" the message, revealing the plaintext, or part of it. In order to make life more difficult for codebreakers, codemakers designed codes with no predictable relationship between the codegroups and the ordering of the matching plaintext. In practice, this meant that two codebooks were now required, one to find codegroups for encoding, the other to look up codegroups to find plaintext for decoding. Students of foreign languages work much the same way; for, say, a Frenchman studying English, there is need of both an English-French and a French-English dictionary. Such "two-part" codes required more effort to develop, and twice as much effort to distribute (and discard safely when replaced), but they were harder to break.

One-time code

A **one-time code** is a prearranged word, phrase or symbol that is intended to be used only once to convey a simple message, often the signal to execute or abort some plan or confirm that it has succeeded or failed. One time codes are often designed to be included in what would appear to be an innocent conversation. Done properly they are almost impossible to detect, though a trained analyst monitoring the communications of someone who has already aroused suspicion might be able to recognize a comment like "Aunt Bertha has gone into labor" as having an ominous meaning. Famous example of one time codes include:

- "One if by land; two if by sea" in "Paul Revere's Ride" made famous in the poem by Henry Wadsworth Longfellow
- "Climb Mount Niitaka" - the signal to Japanese planes to begin the attack on Pearl Harbor

- During World War II the British Broadcasting Corporation's overseas service frequently included "personal messages" as part of its regular broadcast schedule. The seemingly nonsensical stream of messages read out by announcers were actually one time codes intended for SOE agents operating behind enemy lines. An example might be "The princess wears red shoes" or "Mimi's cat is asleep under the table". Each code message was read out twice. By such means, the French Resistance were instructed to start sabotaging rail and other transport links the night before D-day.
- "Over all of Spain, the sky is clear" was a signal (broadcast on radio) to start the nationalist military revolt in Spain on July 17, 1936.

Sometimes messages are not prearranged and rely on shared knowledge hopefully known only to the recipients. An example is the telegram sent to U.S. President Harry Truman, then in Potsdam to meet with Stalin, informing Truman of the first successful test of an atomic bomb.

"Operated on this morning. Diagnosis not yet complete but results seem satisfactory and already exceed expectations. Local press release necessary as interest extends great distance. Dr. Groves pleased. He returns tomorrow. I will keep you posted."

Idiot code

An *idiot code* is a code that is created by the parties using it. This type of communication is akin to the hand signals used by armies in the field.

Example: Any sentence where 'day' and 'night' are used means 'attack'. The location mentioned in the following sentence specifies the location to be attacked.

- *Plaintext:* Attack Gotham.
- *Codetext:* We walked day and night through the streets but couldn't find it! Tomorrow we'll head into Gotham.

An early use of the term appears to be by George Perrault, a character in the science fiction book *Friday* by Robert A. Heinlein:

The simplest sort [of code] and thereby impossible to break. The first ad told the person or persons concerned to carry out number seven or expect number seven or it said something about something designated as seven. This one says the same with respect to code item number ten. But the meaning of the numbers cannot be deduced through statistical analysis because the code can be changed long before a useful statistical universe can be reached. It's an idiot code... and an idiot code can never be broken if the user has the good sense not to go too often to the well.

Richard Miniter, author of *Losing Bin Laden: How Bill Clinton's Failures Unleashed Global Terror*, was quoted in an interview by UPI Technology News:

Another way terrorists use the Internet to communicate is through conventional message boards. They simply go to common public places online, chat rooms and the like, and post messages using what intelligence operatives call an "idiot code", said Miniter.

Terrorism expert Magnus Ranstorp said that the men who carried out the September 11, 2001, attacks on the United States used basic e-mail and what he calls "idiot code" to discuss their plans.

Cryptanalysis of codes

While solving a monoalphabetic substitution cipher is easy, solving even a simple code is difficult. Decrypting a coded message is a little like trying to translate a document written in a foreign language, with the task basically amounting to building up a "dictionary" of the codegroups and the plaintext words they represent.

One fingerhold on a simple code is the fact that some words are more common than others, such as "the" or "a" in English. In telegraphic messages, the codegroup for "STOP" (i.e., end of sentence or paragraph) is usually very common. This helps define the structure of the message in terms of sentences, if not their meaning, and this is cryptanalytically useful.

Further progress can be made against a code by collecting many codetexts encrypted with the same code and then using information from other sources

- spies,
- newspapers,
- diplomatic cocktail party chat,
- the location from where a message was sent,
- where it was being sent to (i.e., traffic analysis)
- the time the message was sent,
- events occurring before and after the message was sent
- the normal habits of the people sending the coded messages
- etc.

For example, a particular codegroup found almost exclusively in messages from a particular army and nowhere else might very well indicate the commander of that army. A codegroup that appears in messages preceding an attack on a particular location may very well stand for that location.

Of course, cribs can be an immediate giveaway to the definitions of codegroups. As codegroups are determined, they can gradually build up a critical mass, with more and more codegroups revealed from context and educated guesswork. One-part codes are more vulnerable to such educated guesswork than two-part codes, since if the codenumber "26839" of a one-part code is determined to stand for "bulldozer", then the lower codenumber "17598" will likely stand for a plaintext word that starts with "a" or "b". At least, for simple one part codes.

Various tricks can be used to "plant" or "sow" information into a coded message, for example by executing a raid at a particular time and location against an enemy, and then examining code messages sent after the raid. Coding errors are a particularly useful fingerhold into a code; people reliably make errors, sometimes disastrous ones. Of course, planting data and exploiting errors works against ciphers as well.

- The most obvious and, in principle at least, simplest way of cracking a code is to steal the codebook through bribery, burglary, or raiding parties — procedures sometimes glorified by the phrase "practical cryptography" — and this is a weakness for both codes and ciphers, though codebooks are generally larger and used longer than cipher keys. While a good code may be harder to break than a cipher, the need to write and distribute codebooks is seriously troublesome.

Constructing a new code is like building a new language and writing a dictionary for it; it was an especially big job before computers. If a code is compromised, the entire task must be done all over again, and that means a lot of work for both cryptographers and the code users. In practice, when codes were in widespread use, they were usually changed on a periodic basis to frustrate codebreakers, and to limit the useful life of stolen or copied codebooks.

Once codes have been created, codebook distribution is logistically clumsy, and increases chances the code will be compromised. Codes can be thought reasonably secure if they are only used by a few careful people, but if whole armies use the same codebook, security becomes much more difficult.

In contrast, the security of ciphers is generally dependent on protecting the cipher keys. Cipher keys can be stolen and people can betray them, but they are much easier to change and distribute.

Superencipherment

In more recent practice, it became typical to encipher a message after first encoding it, so as to provide greater security by increasing the degree of difficulty for cryptanalysts. With a numerical code, this was commonly done with an "additive" - simply a long key number which was digit-by-digit added to the code groups, modulo 10. Unlike the codebooks, additives would be changed frequently. The famous Japanese Navy code, JN-25, was of this design, as were several of the (confusingly named) Royal Navy Cyphers used after WWI and into WWII.

One might wonder why a code would be used if it had to be enciphered to provide security. As well as providing security, a well designed code can also compress the message, and provide some degree of automatic error correction. For ciphers, the same degree of error correction has generally required use of computers.

Types of cipher

There are a variety of different types of encryption. Algorithms used earlier in the history of cryptography are substantially different from modern methods, and modern ciphers can be classified according to how they operate and whether they use one or two keys.

Historical ciphers

Historical pen and paper ciphers used in the past are sometimes known as classical ciphers. They include simple substitution ciphers and transposition ciphers. For example "GOOD DOG" can be encrypted as "PLLX XLP" where "L" substitutes for "O", "P" for "G", and "X" for "D" in the message. Transposition of the letters "GOOD DOG" can result in "DGOGDOO". These simple ciphers and examples are easy to crack, even without plaintext-ciphertext pairs.

Simple ciphers were replaced by polyalphabetic substitution ciphers which changed the substitution alphabet for every letter. For example "GOOD DOG" can be encrypted as "PLSX TWF" where "L", "S", and "W" substitute for "O". With even a small amount of known or estimated plaintext, simple polyalphabetic substitution ciphers and letter transposition ciphers designed for pen and paper encryption are easy to crack.

During the early twentieth century, electro-mechanical machines were invented to do encryption and decryption using transposition, polyalphabetic substitution, and a kind of "additive" substitution. In rotor machines, several rotor disks provided polyalphabetic substitution, while plug boards provided another substitution. Keys were easily changed by changing the rotor disks and the plugboard wires. Although these encryption methods were more complex than previous schemes and required machines to encrypt and decrypt, other machines such as the British Bombe were invented to crack these encryption methods.

Modern ciphers

Modern encryption methods can be divided by two criteria: by type of key used, and by type of input data.

By type of key used ciphers are divided into:

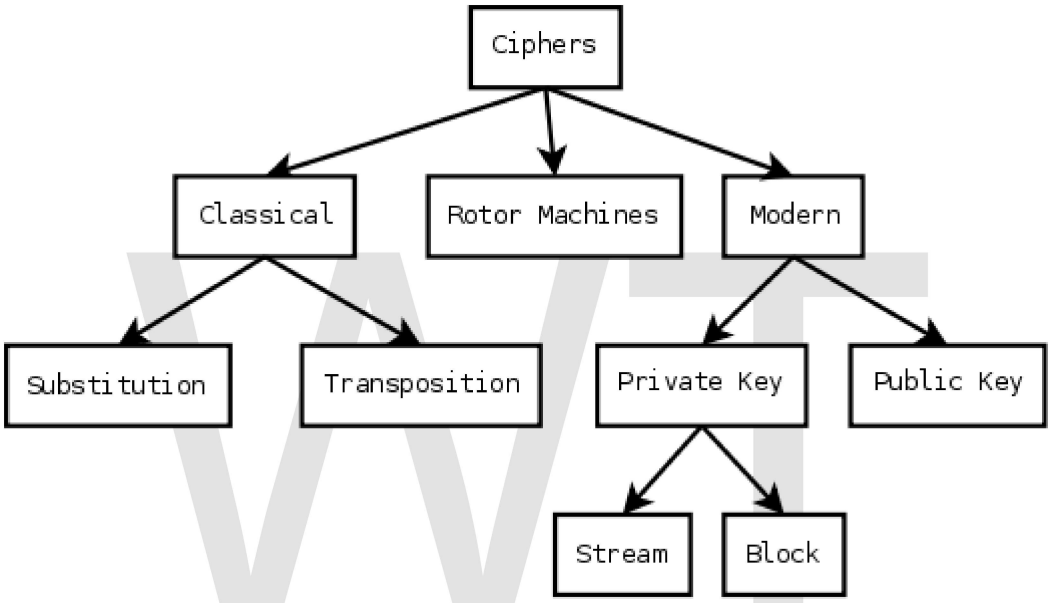
- symmetric key algorithms (Private-key cryptography), where the same key is used for encryption and decryption, and
- asymmetric key algorithms (Public-key cryptography), where two different keys are used for encryption and decryption.

In a symmetric key algorithm (e.g., DES and AES), the sender and receiver must have a shared key set up in advance and kept secret from all other parties; the sender uses this key for encryption, and the receiver uses the same key for decryption. The Feistel cipher uses a combination of substitution and transposition techniques. Most block cipher

algorithms are based on this structure. In an asymmetric key algorithm (e.g., RSA), there are two separate keys: a *public key* is published and enables any sender to perform encryption, while a *private key* is kept secret by the receiver and enables only him to perform correct decryption.

Type of input ciphers data can be distinguished into two types:

- block ciphers, which encrypt block of data of fixed size, and
- stream ciphers, which encrypt continuous streams of data



Key Size and Vulnerability

In a pure mathematical attack (i.e., lacking any other information to help break a cipher), three factors above all, count:

- Mathematical advances that allow new attacks or weaknesses to be discovered and exploited.
- Computational power available, i.e., the computing power which can be brought to bear on the problem. It is important to note that average performance/capacity of a single computer is not the only factor to consider. An adversary can use multiple computers at once, for instance, to increase the speed of exhaustive search for a key (i.e., "brute force" attack) substantially.
- Key size, i.e., the size of key used to encrypt a message. As the key size increases, so does the complexity of exhaustive search to the point where it becomes infeasible to crack encryption directly.

Since the desired effect is computational difficulty, in theory one would choose an algorithm and desired difficulty level, thus decide the key length accordingly.

An example of this process can be found at Key Length which uses multiple reports to suggest that a symmetric cipher with 128 bits, an asymmetric cipher with 3072 bit keys, and an elliptic curve cipher with 512 bits, all have similar difficulty at present.

Claude Shannon proved, using information theory considerations, that any theoretically unbreakable cipher must have keys which are at least as long as the plaintext, and used only once: one-time pad.

Chapter 6

Commitment Scheme

In cryptography, a **commitment scheme** allows one to commit to a value while keeping it hidden, with the ability to reveal the committed value later. Commitments are used to bind a party to a value so that they cannot adapt to other messages in order to gain some kind of inappropriate advantage. They are important to a variety of cryptographic protocols including secure coin flipping, zero-knowledge proofs, and secure computation.

Interactions in a commitment scheme take place in two phases:

1. the *commit phase* during which a value is chosen and specified
2. the *reveal phase* during which the value is revealed and checked

In simple protocols, the commit phase consists of a single message from the sender to the receiver. This message is called *the commitment*. It is essential that the specific value chosen cannot be known by the receiver at that time (this is called the *hiding* property). A simple reveal phase would consist of a single message, *the opening*, from the sender to the receiver, followed by a check performed by the receiver. The value chosen during the commit phase must be the only one that the sender can compute and that validates during the reveal phase (this is called the *binding* property).

The concept of commitment schemes was first formalized by Gilles Brassard, David Chaum, and Claude Crepeau in 1988, but the concept was used without being treated formally prior to that. The notion of commitments appeared earliest in works by Manuel Blum, Shimon Even, and Shamir et al. The terminology seems to have been originated by Blum, although commitment schemes can be interchangeably called **bit commitment schemes**—sometimes reserved for the special case where the committed value is a binary bit.

Applications

Coin flipping

Suppose Alice and Bob want to resolve some dilemma via coin flipping. If they are physically in the same place, a typical procedure might be:

1. Alice "calls" the coin flip

2. Bob flips the coin
3. If Alice's call is correct, she wins, otherwise Bob wins

If they are not in the same place, this procedure is faulty. Alice would have to trust Bob's report of how the coin flip turned out, whereas Bob knows what result is more desirable for him. Using commitments, a similar procedure is:

1. Alice "calls" the coin flip and tells Bob only a *commitment* to her call,
2. Bob flips the coin and reports the result,
3. Alice reveals what she committed to
4. If Alice's revelation matches the coin result Bob reported, Alice wins

For Bob to be able to skew the results to his favor, he must be able to understand the call hidden in Alice's commitment. If the commitment scheme is a good one, Bob cannot skew the results. Similarly, Alice cannot affect the result if she cannot change the value she commits to.

A way to visualize a commitment scheme is to think of the sender as putting the value in a locked box, and giving the box to the receiver. The value in the box is hidden from the receiver, who cannot open the lock themselves. Since the receiver has the box, the value inside cannot be changed—merely revealed if the sender chooses to give them the key at some later time.

Zero-knowledge proofs

One particular motivating example is the use of commitment schemes in zero-knowledge proofs. Commitments are used in zero-knowledge proofs for two main purposes: first, to allow the prover to participate in "cut and choose" proofs where the verifier will be presented with a choice of what to learn, and the prover will reveal only what corresponds to the verifier's choice. Commitment schemes allow the prover to specify all the information in advance in a commitment, and only reveal what should be revealed later in the proof. Commitments are also used in zero-knowledge proofs by the verifier, who will often specify their choices ahead of time in a commitment. This allows zero-knowledge proofs to be composed in parallel without revealing additional information.

Verifiable secret sharing

Another important application of commitments is in verifiable secret sharing, a critical building block of secure multiparty computation. In a secret sharing scheme, each of several parties receive "shares" of a value that is meant to be hidden from everyone. If enough parties get together, their shares can be used to reconstruct the secret, but even a malicious cabal of insufficient size should learn nothing. Secret sharing is at the root of many protocols for secure computation: in order to securely compute a function of some shared input, the secret shares are manipulated instead. However, if shares are to be generated by malicious parties, it may be important that those shares can be checked for correctness. In a verifiable secret sharing scheme, the distribution of a secret is

accompanied by commitments to the individual shares. The commitments reveal nothing that can help a dishonest cabal, but the shares allow each individual party to check to see if their shares are correct.

Defining the security of commitment schemes

Formal definitions of commitment schemes vary strongly in notation and in flavour. The first such flavour is whether the commitment scheme provides perfect or computational security with respect to the hiding or binding properties. Another such flavour is whether the commitment is interactive, i.e. whether both the commit phase and the reveal phase can be seen as being executed by a Cryptographic protocol or whether they are non-interactive, consisting of two algorithms *Commit* and *CheckReveal*. In the latter case *CheckReveal* can often be seen as a derandomised version of *Commit*, with the randomness used by *Commit* constituting the opening information.

If the commitment C to a value x is computed as $C:=Commit(x,open)$ with *open* the randomness used for computing the commitment, then $CheckReveal(C,x,open)$ simply consists in verifying the equation $C=Commit(x,open)$.

Using this notation and some knowledge about mathematical functions and probability theory we formalise different versions of the binding and hiding properties of commitments. The two most important combinations of these properties are perfectly binding and computationally hiding commitment schemes and computationally binding and perfectly hiding commitment schemes. Note that no commitment scheme can be at the same time perfectly binding and perfectly hiding.

Computational binding

Let *open* be chosen from a set of size 2^k, i.e., it can represented as a k bit string, and let $Commit_k$ be the corresponding commitment scheme. As the size of k determines the security of the commitment scheme it is called the security parameter.

Then for all non-uniform probabilistic polynomial time algorithms that output x,x' and *open*,*open*' of increasing length k, the probability that $x \neq x'$ and $Commit_k(x,open) = Commit_k(x',open')$ is a negligible function in k.

This is a form of Asymptotic analysis. It is also possible to state the same requirement using Concrete security: A commitment scheme *Commit* is (t,ε) secure, if for all algorithms that run in time t and output x,x',*open*,*open*' the probability that $x \neq x'$ and $Commit(x,open) = Commit(x',open')$ is at most ε.

Perfect, statistical and computational hiding

Let U_k be the uniform distribution over the 2^k opening values for security parameter k. A commitment scheme is perfect, statistical, computational hiding, if for all $x \neq x'$ the

probability ensembles $\{Commit_k(x, U_k)\}_{k\in\mathbb{N}}$ and $\{Commit_k(x', U_k)\}_{k\in\mathbb{N}}$ are equal, statistically close, or computationally indistinguishable.

Constructing commitment schemes

A commitment scheme can either be perfectly binding (it is impossible for Alice to alter her commitment after she has made it, even if she has unbounded computational resources) or perfectly concealing (it is impossible for Bob to find out the commitment without Alice revealing it, even if he has unbounded computational resources) but not both.

Bit-commitment from any one-way permutation (or injective one way function)

One can create a bit-commitment scheme from any one-way function that is injective. The scheme relies on the fact that every one-way function can be modified (via the Goldreich-Levin theorem) to possess a computationally hard-core predicate (while retaining the injective property). Let *f* be an injective one-way function, with *h* a hard-core predicate. Then to commit to a bit *b* Alice picks a random input *x* and sends the triple

$$(h, f(x), b \oplus h(x))$$

to Bob, where $\oplus$ denotes XOR, i.e. addition modulo 2. To decommit Alice simply sends *x* to Bob. Bob verifies by computing *f(x)* and comparing to the committed value. This scheme is concealing because for Bob to recover *b* he must recover *h(x)*. Since *h* is a computationally hard-core predicate, recovering *h(x)* from *f(x)* with probability greater than one-half is as hard as inverting *f*. Perfect binding follows from the fact that *f* is injective and thus *f(x)* has exactly one preimage.

Bit-commitment from a pseudo-random generator

Note that since we do not know how to construct a one-way permutation from any one-way function, this section reduces the strength of the cryptographic assumption necessary to construct a bit-commitment protocol.

In 1991 Moni Naor showed how to create a bit-commitment scheme from a cryptographically secure pseudorandom number generator. The construction is as follows. If *G* is pseudo-random generator such that *G* takes *n* bits to *3n* bits, then if Alice wants to commit to a bit *b*

- Bob selects a random *3n*-bit vector *R* and sends *R* to Alice.
- Alice selects a random *n*-bit vector *Y* and computes the *3n*-bit vector *G(Y)*.
- If *b*=1 Alice sends *G(Y)* to Bob, otherwise she sends the bitwise exclusive-or of *G(Y)* and *R* to Bob.

To decommit Alice sends *Y* to Bob, who can then check whether he initially received *G(Y)* or $G(Y) \oplus R$.

This scheme is statistically binding, meaning that even if Alice is computationally unbounded she cannot cheat with probability greater than 2^{-n}. For Alice to cheat, she would need to find a *Y′*, such that $G(Y') = G(Y) \oplus R$. If she could find such a value, she could decommit by sending the truth and Y, or send the opposite answer and Y'. However, G(Y) and G(Y') are only able to produce 2^n possible values while R and the commitment are both picked out of 2^{3n} values in the set of possible *3n*-bit values. She does not pick R, so there is a strong probability that a Y' satisfying the equation required to cheat will not exist.

The concealing property follows from a standard reduction, if Bob can tell whether Alice committed to a zero or one, he can also distinguish the output of the pseudo-random generator *G* from true-random, which contradicts the cryptographic security of *G*.

A perfectly binding scheme based on the discrete log problem

Alice chooses a group of prime order *p*, with generator *g*.

Alice randomly picks a secret value *x* from *0* to $p - 1$ to commit to and calculates $c = g^x$ and publishes *c*. The discrete logarithm problem dictates that from *c*, it is computationally infeasible to compute *x*, so under this assumption, Bob cannot compute *x*. On the other hand, Alice cannot compute a $x' <> x$, such that $g^{x'} = c$, so the scheme is binding.

This scheme isn't perfectly concealing as someone could find the commitment if he manages to solve the discrete logarithm problem. (In fact, this scheme isn't hiding at all with respect to the standard hiding game, where an adversary should be unable to guess which of two messages he chose were committed to - similar to the IND-CPA game. A better example of a perfectly binding commitment scheme is one where the commitment is the encryption of *x* under a semantically secure, public-key encryption scheme, and the decommitment is the string of random bits used to encrypt *x*.)

Quantum bit commitment

It is an interesting question in quantum cryptography if there exist *unconditionally secure* bit commitment protocols on the quantum level, that is protocols which are (at least asymptotically) binding and concealing even if there are no restrictions on the computational resources. One could hope that there might be a way to exploit the intrinsic properties of quantum mechanics, as in the protocols for unconditionally secure key distribution.

However, Dominic Mayers showed in 1996, that this is impossible. Any such protocol can be reduced to a protocol where the system is in one of two pure states after the commitment phase, depending on the bit Alice wants to commit. If the protocol is

unconditionally concealing, then Alice can unitarily transform these states into each other using the properties of the Schmidt decomposition, effectively defeating the bindingness property.

One subtle assumption of the proof is that the commit phase must be finished at some point in time. This leaves room for protocols that require a continuing information flow until the bit is unveiled or the protocol is cancelled, in which case it is not binding anymore.

Chapter 7

Digital Signature and Data Encryption Standard

Digital signature

A **digital signature** or **digital signature scheme** 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, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery and tampering.

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, but not all electronic signatures use digital signatures. In some countries, including the United States, India, and members of the European Union, electronic signatures have legal significance. However, laws concerning electronic signatures do not always make clear whether they are digital cryptographic signatures in the sense used here, leaving the legal definition, and so their importance, somewhat confused.

Digital signatures employ a type of asymmetric cryptography. For messages sent through an insecure 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; 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 nonetheless. 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

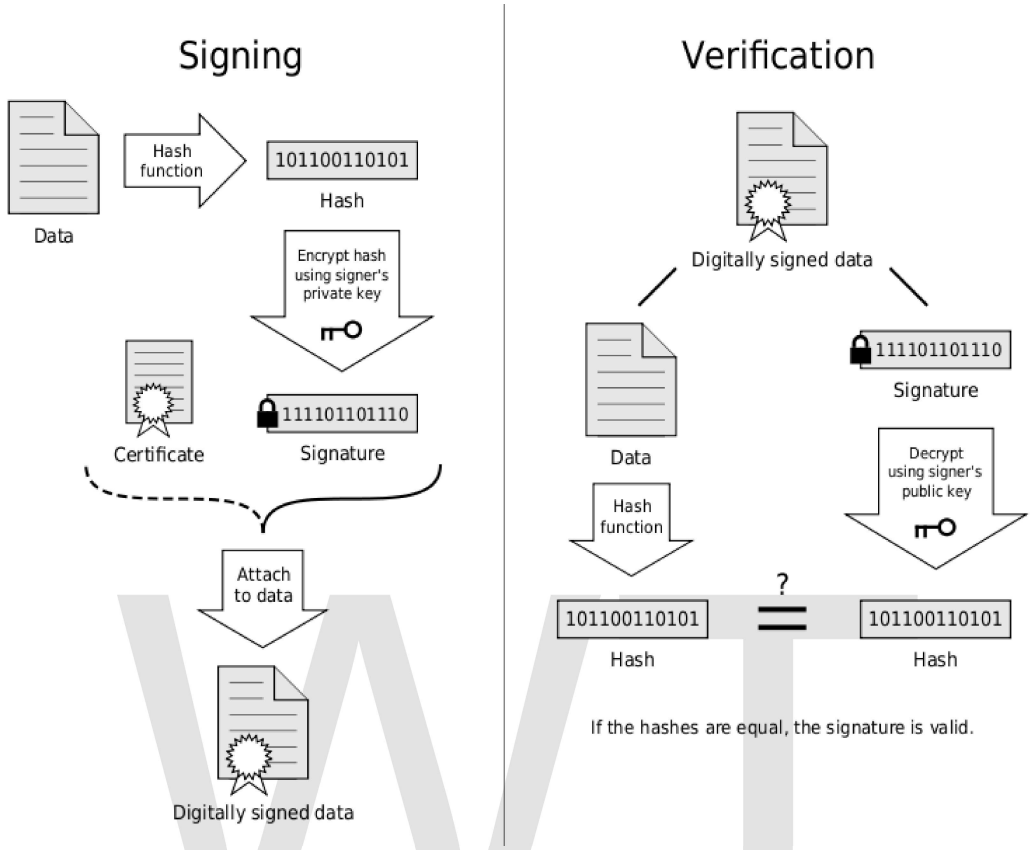


Diagram showing how a simple digital signature is applied and then verified

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, a signature generated from a fixed message and fixed private key should verify the authenticity of that message by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party who does not possess the 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. 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). The first widely marketed software package to offer digital signature was Lotus Notes 1.0, released in 1989, which used the RSA algorithm.

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 model.

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

In 1988, Shafi Goldwasser, Silvio Micali, and Ronald Rivest became the first to rigorously define the security requirements of digital signature schemes. 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.

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. 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-and-decrypt form of signature is existentially unforgeable, even against a chosen-message attack.

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.

Uses 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 will invalidate 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.

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. This is in contrast to symmetric systems,

where both sender and receiver share the same secret key, and thus in a dispute a third party cannot determine which entity was the true source of the information.

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 encrypts the hash using the stored private key of the user, and then returns the encrypted 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 encrypted 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 app and signing app to verify each other's integrity. For example, the signing application may require all requests to come from digitally-signed binaries.

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) means that the semantic interpretation of a signed message cannot be changed. In particular this also means that a message cannot contain hidden info that the signer is unaware of, and that can be revealed after the signature has been applied. WYSIWYS is a desirable property of digital signatures that is difficult to guarantee because of the increasing complexity of modern computer systems.

Digital signatures vs. ink on paper signatures

An ink signature can be easily replicated from one document to another by copying the image manually or digitally. Digital signatures cryptographically bind an electronic identity to an electronic document and the digital signature cannot be copied to another document. Paper contracts often 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.

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.

The current state of use — legal and practical

Digital signature schemes share basic prerequisites that— regardless of cryptographic theory or legal provision— they need to have meaning:

1. Quality algorithms
   Some public-key algorithms are known to be insecure, practicable attacks against them having been discovered.
2. Quality implementations
   An implementation of a good algorithm (or protocol) with mistake(s) will not work.
3. The private key must remain private
   if it becomes known to any other party, that party can produce *perfect* digital signatures of anything whatsoever.
4. 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 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 non trivial. 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.
5. 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.

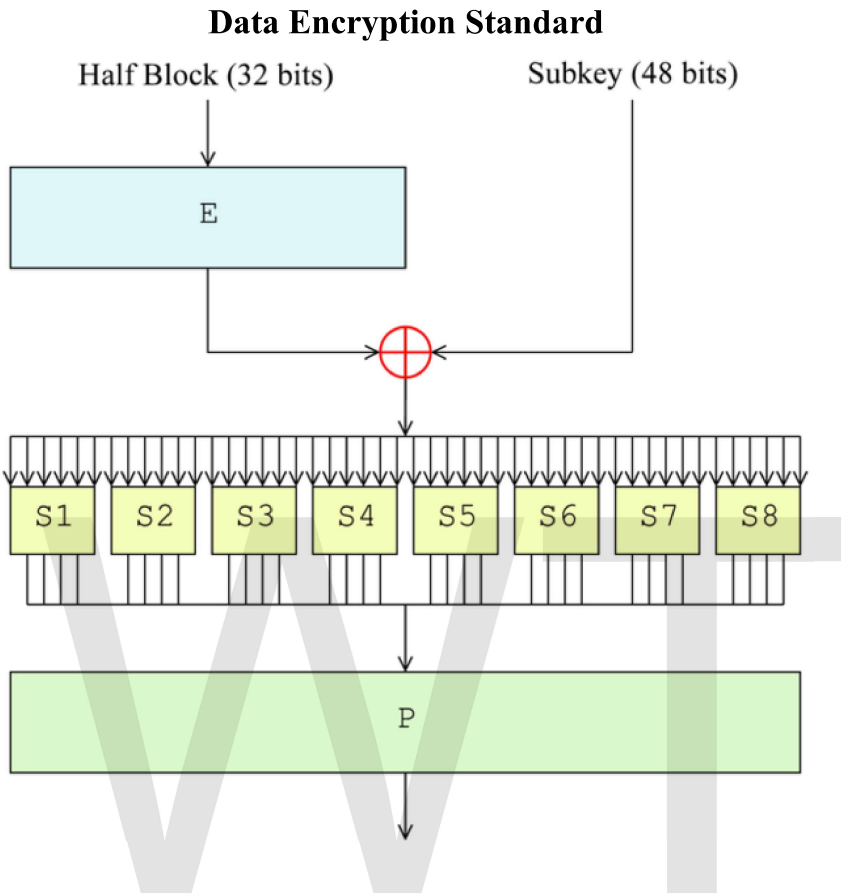
Industry standards

Some industries have established common interoperabiltity 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. 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.

Data Encryption Standard



The Feistel function (F function) of DES

General

Designers	IBM
First published	1977 (standardized on January 1979)
Derived from	Lucifer
Successors	Triple DES, G-DES, DES-X, LOKI89, ICE

Cipher detail

Key sizes	56 bits
Block sizes	64 bits
Structure	Balanced Feistel network
Rounds	16

Best public cryptanalysis

DES is now considered insecure because a brute force attack is possible. As of 2008, the best analytical attack is linear cryptanalysis, which requires 2^{43} known plaintexts and has a time complexity of $2^{39–43}$ (Junod, 2001); under a chosen-plaintext assumption, the data complexity can be reduced by a factor of four (Knudsen and Mathiassen, 2000).

The **Data Encryption Standard** (**DES**) is a block cipher that uses shared secret encryption. It was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally. It is based on a symmetric-key algorithm that uses a 56-bit key. The algorithm was initially controversial with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis.

DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes. There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES). Furthermore, DES has been withdrawn as a standard by the National Institute of Standards and Technology (formerly the National Bureau of Standards).

In some documentation, a distinction is made between DES as a standard and DES the algorithm which is referred to as the **DEA** (the **Data Encryption Algorithm**).

History of DES

The origins of DES go back to the early 1970s. In 1972, after concluding a study on the US government's computer security needs, the US standards body NBS (National Bureau of Standards) — now named NIST (National Institute of Standards and Technology) — identified a need for a government-wide standard for encrypting unclassified, sensitive information. Accordingly, on 15 May 1973, after consulting with the NSA, NBS solicited proposals for a cipher that would meet rigorous design criteria. None of the submissions, however, turned out to be suitable. A second request was issued on 27 August 1974. This time, IBM submitted a candidate which was deemed acceptable — a cipher developed during the period 1973–1974 based on an earlier algorithm, Horst Feistel's Lucifer cipher. The team at IBM involved in cipher design and analysis included Feistel, Walter Tuchman, Don Coppersmith, Alan Konheim, Carl Meyer, Mike Matyas, Roy Adler, Edna Grossman, Bill Notz, Lynn Smith, and Bryant Tuckerman.

NSA's involvement in the design

On 17 March 1975, the proposed DES was published in the *Federal Register*. Public comments were requested, and in the following year two open workshops were held to discuss the proposed standard. There was some criticism from various parties, including from public-key cryptography pioneers Martin Hellman and Whitfield Diffie, citing a shortened key length and the mysterious "S-boxes" as evidence of improper interference from the NSA. The suspicion was that the algorithm had been covertly weakened by the intelligence agency so that they — but no-one else — could easily read encrypted messages. Alan Konheim (one of the designers of DES) commented, "We sent the S-boxes off to Washington. They came back and were all different." The United States Senate Select Committee on Intelligence reviewed the NSA's actions to determine whether there had been any improper involvement. In the unclassified summary of their findings, published in 1978, the Committee wrote:

In the development of DES, NSA convinced IBM that a reduced key size was sufficient; indirectly assisted in the development of the S-box structures; and certified that the final DES algorithm was, to the best of their knowledge, free from any statistical or mathematical weakness.

However, it also found that

NSA did not tamper with the design of the algorithm in any way. IBM invented and designed the algorithm, made all pertinent decisions regarding it, and concurred that the agreed upon key size was more than adequate for all commercial applications for which the DES was intended.

Another member of the DES team, Walter Tuchman, stated "We developed the DES algorithm entirely within IBM using IBMers. The NSA did not dictate a single wire!" In contrast, a declassified NSA book on cryptologic history states:

In 1973 NBS solicited private industry for a data encryption standard (DES). The first offerings were disappointing, so NSA began working on its own algorithm. Then Howard Rosenblum, deputy director for research and engineering, discovered that Walter Tuchman of IBM was working on a modification to Lucifer for general use. NSA gave Tuchman a clearance and brought him in to work jointly with the Agency on his Lucifer modification."

and

NSA worked closely with IBM to strengthen the algorithm against all except brute force attacks and to strengthen substitution tables, called S-boxes. Conversely, NSA tried to convince IBM to reduce the length of the key from 64 to 48 bits. Ultimately they compromised on a 56-bit key.

Some of the suspicions about hidden weaknesses in the S-boxes were allayed in 1990, with the independent discovery and open publication by Eli Biham and Adi Shamir of differential cryptanalysis, a general method for breaking block ciphers. The S-boxes of DES were much more resistant to the attack than if they had been chosen at random, strongly suggesting that IBM knew about the technique in the 1970s. This was indeed the case; in 1994, Don Coppersmith published some of the original design criteria for the S-boxes. According to Steven Levy, IBM Watson researchers discovered differential cryptanalytic attacks in 1974 and were asked by the NSA to keep the technique secret. Coppersmith explains IBM's secrecy decision by saying, "that was because [differential cryptanalysis] can be a very powerful tool, used against many schemes, and there was concern that such information in the public domain could adversely affect national security." Levy quotes Walter Tuchman: "[t]hey asked us to stamp all our documents confidential... We actually put a number on each one and locked them up in safes, because they were considered U.S. government classified. They said do it. So I did it". Bruce Schneier observed that "It took the academic community two decades to figure out that the NSA 'tweaks' actually improved the security of DES."

The algorithm as a standard

Despite the criticisms, DES was approved as a federal standard in November 1976, and published on 15 January 1977 as **FIPS PUB 46**, authorized for use on all unclassified data. It was subsequently reaffirmed as the standard in 1983, 1988 (revised as **FIPS-46-1**), 1993 (**FIPS-46-2**), and again in 1999 (**FIPS-46-3**), the latter prescribing "Triple DES" (see below). On 26 May 2002, DES was finally superseded by the Advanced Encryption Standard (AES), following a public competition. On 19 May 2005, FIPS 46-3 was officially withdrawn, but NIST has approved Triple DES through the year 2030 for sensitive government information.

The algorithm is also specified in ANSI X3.92, NIST SP 800-67 and ISO/IEC 18033-3 (as a component of TDEA).

Another theoretical attack, linear cryptanalysis, was published in 1994, but it was a brute force attack in 1998 that demonstrated that DES could be attacked very practically, and highlighted the need for a replacement algorithm.

The introduction of DES is considered to have been a catalyst for the academic study of cryptography, particularly of methods to crack block ciphers. According to a NIST retrospective about DES,

The DES can be said to have "jump started" the nonmilitary study and development of encryption algorithms. In the 1970s there were very few cryptographers, except for those in military or intelligence organizations, and little academic study of cryptography. There are now many active academic cryptologists, mathematics departments with strong programs in cryptography, and commercial information security companies and consultants. A generation of cryptanalysts has cut its teeth analyzing (that is trying to "crack") the DES algorithm. In the words of cryptographer Bruce Schneier, "DES did

more to galvanize the field of cryptanalysis than anything else. Now there was an algorithm to study." An astonishing share of the open literature in cryptography in the 1970s and 1980s dealt with the DES, and the DES is the standard against which every symmetric key algorithm since has been compared.

Chronology

Date	Year	Event
15 May	1973	NBS publishes a first request for a standard encryption algorithm
27 August	1974	NBS publishes a second request for encryption algorithms
17 March	1975	DES is published in the *Federal Register* for comment
August	1976	First workshop on DES
September	1976	Second workshop, discussing mathematical foundation of DES
November	1976	DES is approved as a standard
15 January	1977	DES is published as a FIPS standard FIPS PUB 46
	1983	DES is reaffirmed for the first time
	1986	Videocipher II, a TV satellite scrambling system based upon DES begins use by HBO
22 January	1988	DES is reaffirmed for the second time as FIPS 46-1, superseding FIPS PUB 46
July	1990	Biham and Shamir rediscover differential cryptanalysis, and apply it to a 15-round DES-like cryptosystem.
	1992	Biham and Shamir report the first theoretical attack with less complexity than brute force: differential cryptanalysis. However, it requires an unrealistic 2^{47} chosen plaintexts.
30 December	1993	DES is reaffirmed for the third time as FIPS 46-2
	1994	The first experimental cryptanalysis of DES is performed using linear cryptanalysis (Matsui, 1994).
June	1997	The DESCHALL Project breaks a message encrypted with DES for the first time in public.
July	1998	The EFF's DES cracker (Deep Crack) breaks a DES key in 56 hours.
January	1999	Together, Deep Crack and distributed.net break a DES key in 22 hours and 15 minutes.
25 October	1999	DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the preferred use of Triple DES, with single DES permitted only in legacy systems.
26 November	2001	The Advanced Encryption Standard is published in FIPS 197
26 May	2002	The AES standard becomes effective
26 July	2004	The withdrawal of FIPS 46-3 (and a couple of related standards) is proposed in the *Federal Register*
19 May	2005	NIST withdraws FIPS 46-3
April	2006	The FPGA based parallel machine COPACOBANA of the Universities of Bochum and Kiel, Germany, breaks DES in 9 days at $10,000 hardware cost. Within a year software improvements reduced the average time to 6.4 days.
Nov.	2008	The successor of COPACOBANA, the RIVYERA machine reduced the average time to less than one single day.

Replacement algorithms

Concerns about security and the relatively slow operation of DES in software motivated researchers to propose a variety of alternative block cipher designs, which started to appear in the late 1980s and early 1990s: examples include RC5, Blowfish, IDEA,

NewDES, SAFER, CAST5 and FEAL. Most of these designs kept the 64-bit block size of DES, and could act as a "drop-in" replacement, although they typically used a 64-bit or 128-bit key. In the USSR the GOST 28147-89 algorithm was introduced, with a 64-bit block size and a 256-bit key, which was also used in Russia later.

DES itself can be adapted and reused in a more secure scheme. Many former DES users now use Triple DES (TDES) which was described and analysed by one of DES's patentees; it involves applying DES three times with two (2TDES) or three (3TDES) different keys. TDES is regarded as adequately secure, although it is quite slow. A less computationally expensive alternative is DES-X, which increases the key size by XORing extra key material before and after DES. GDES was a DES variant proposed as a way to speed up encryption, but it was shown to be susceptible to differential cryptanalysis.

In 2001, after an international competition, NIST selected a new cipher, the Advanced Encryption Standard (AES), as a replacement. The algorithm which was selected as the AES was submitted by its designers under the name Rijndael. Other finalists in the NIST AES competition included RC6, Serpent, MARS and Twofish.

Description

DES is the archetypal block cipher — an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is usually quoted as such. Every 8th bit of the selected key is discarded, i.e. positions 8, 16, 24, 32, 40, 48, 56, 64 are removed from the 64 bit key leaving behind only the 56 bit key.

Like other block ciphers, DES by itself is not a secure means of encryption but must instead be used in a mode of operation. FIPS-81 specifies several modes for use with DES. Further comments on the usage of DES are contained in FIPS-74.

Overall structure

There is also an initial and final permutation, termed *IP* and *FP*, which are inverses (IP "undoes" the action of FP, and vice versa). IP and FP have almost no cryptographic significance, but were apparently included in order to facilitate loading blocks in and out of mid-1970s hardware.

Before the main rounds, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes — the only difference

is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms.

The ⊕ symbol denotes the exclusive-OR (XOR) operation. The *F-function* scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

The Feistel (F) function

1. *Expansion* — the 32-bit half-block is expanded to 48 bits using the *expansion permutation*. The output consists of eight 6-bit(8*6=48bits) pieces, each containing a copy of 4 corresponding input bits, plus a copy of the immediately adjacent bit from each of the input pieces to either side.
2. *Key mixing* — the result is combined with a *subkey* using an XOR operation. Sixteen 48-bit subkeys — one for each round — are derived from the main key using the *key schedule* (described below).
3. *Substitution* — after mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the *S-boxes*, or *substitution boxes*. Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, provided in the form of a lookup table. The S-boxes provide the core of the security of DES — without them, the cipher would be linear, and trivially breakable.
4. *Permutation* — finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation, the *P-box*. This is designed so that, after expansion, each S-box's output bits are spread across 6 different S boxes in the next round.

The alternation of substitution from the S-boxes, and permutation of bits from the P-box and E-expansion provides so-called "confusion and diffusion" respectively, a concept identified by Claude Shannon in the 1940s as a necessary condition for a secure yet practical cipher.

Key schedule

The 56 bits are then divided into two 28-bit halves; each half is thereafter treated separately. In successive rounds, both halves are rotated left by one or two bits (specified for each round), and then 48 subkey bits are selected by *Permuted Choice 2* (*PC-2*) — 24 bits from the left half, and 24 from the right. The rotations (denoted by "<<<" in the diagram) mean that a different set of bits is used in each subkey; each bit is used in approximately 14 out of the 16 subkeys.

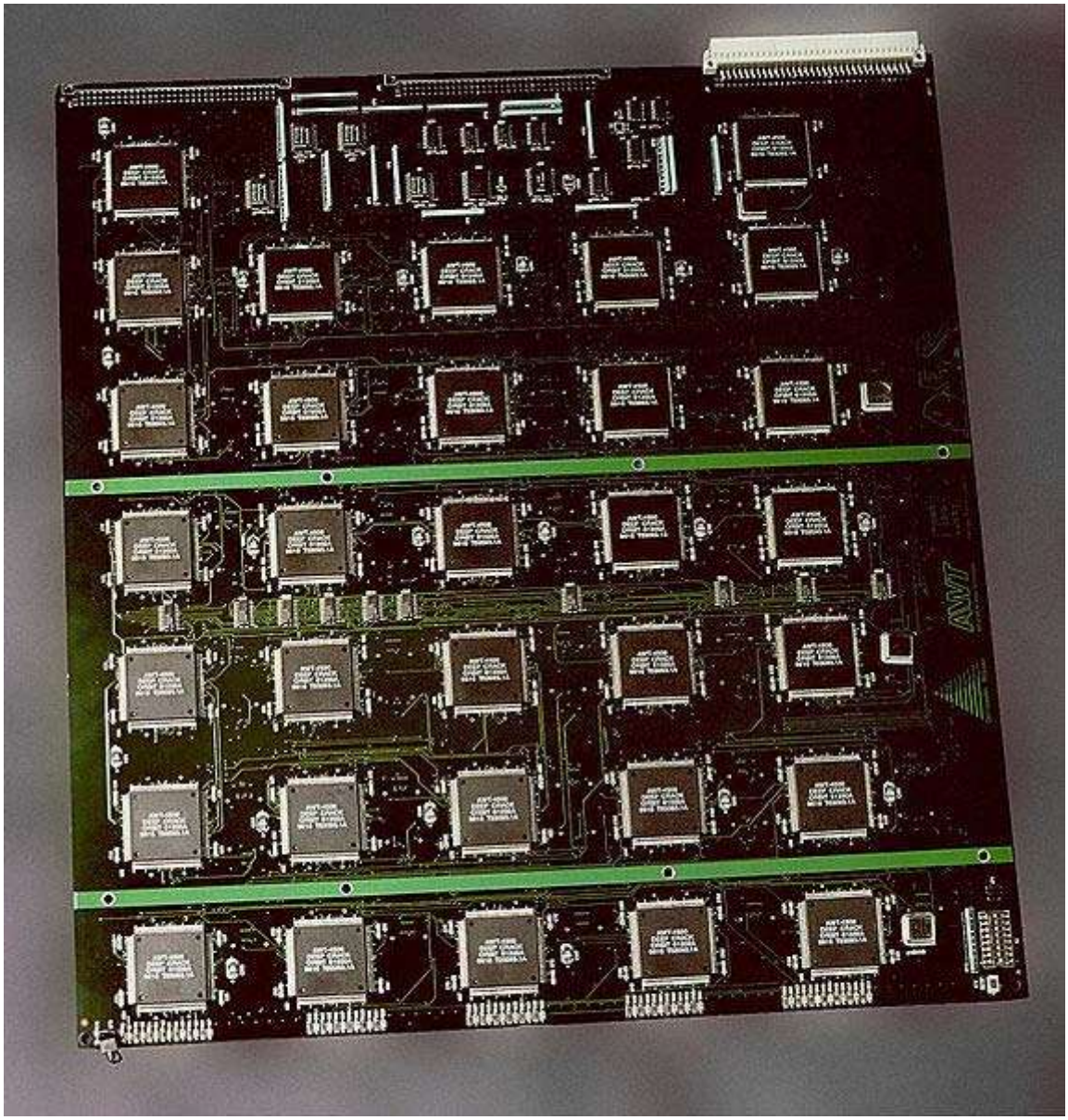
The key schedule for decryption is similar — the subkeys are in reverse order compared to encryption. Apart from that change, the process is the same as for encryption. The same 28 bits are passed to all rotation boxes.

Security and cryptanalysis

Although more information has been published on the cryptanalysis of DES than any other block cipher, the most practical attack to date is still a brute force approach. Various minor cryptanalytic properties are known, and three theoretical attacks are possible which, while having a theoretical complexity less than a brute force attack, require an unrealistic number of known or chosen plaintexts to carry out, and are not a concern in practice.

Brute force attack

For any cipher, the most basic method of attack is brute force — trying every possible key in turn. The length of the key determines the number of possible keys, and hence the feasibility of this approach. For DES, questions were raised about the adequacy of its key size early on, even before it was adopted as a standard, and it was the small key size, rather than theoretical cryptanalysis, which dictated a need for a replacement algorithm. As a result of discussions involving external consultants including the NSA, the key size was reduced from 128 bits to 56 bits to fit on a single chip.



The EFF's US$250,000 DES cracking machine contained 1,856 custom chips and could brute force a DES key in a matter of days — the photo shows a DES Cracker circuit board fitted with several Deep Crack chips.

In academia, various proposals for a DES-cracking machine were advanced. In 1977, Diffie and Hellman proposed a machine costing an estimated US$20 million which could find a DES key in a single day. By 1993, Wiener had proposed a key-search machine costing US$1 million which would find a key within 7 hours. However, none of these early proposals were ever implemented—or, at least, no implementations were publicly acknowledged. The vulnerability of DES was practically demonstrated in the late 1990s. In 1997, RSA Security sponsored a series of contests, offering a $10,000 prize to the first team that broke a message encrypted with DES for the contest. That contest was won by the DESCHALL Project, led by Rocke Verser, Matt Curtin, and Justin Dolske, using idle

cycles of thousands of computers across the Internet. The feasibility of cracking DES quickly was demonstrated in 1998 when a custom DES-cracker was built by the Electronic Frontier Foundation (EFF), a cyberspace civil rights group, at the cost of approximately US$250,000. Their motivation was to show that DES was breakable in practice as well as in theory: "*There are many people who will not believe a truth until they can see it with their own eyes. Showing them a physical machine that can crack DES in a few days is the only way to convince some people that they really cannot trust their security to DES.*" The machine brute-forced a key in a little more than 2 days search.

The next confirmed DES cracker was the COPACOBANA machine built in 2006 by teams of the Universities of Bochum and Kiel, both in Germany. Unlike the EFF machine, COPACOBANA consists of commercially available, reconfigurable integrated circuits. 120 of these Field-programmable gate arrays (FPGAs) of type XILINX Spartan3-1000 run in parallel. They are grouped in 20 DIMM modules, each containing 6 FPGAs. The use of reconfigurable hardware makes the machine applicable to other code breaking tasks as well. One of the more interesting aspects of COPACOBANA is its cost factor. One machine can be built for approximately $10,000. The cost decrease by roughly a factor of 25 over the EFF machine is an impressive example for the continuous improvement of digital hardware. Adjusting for inflation over 8 years yields an even higher improvement of about 30x. Since 2007, SciEngines GmbH, a spin-off company of the two project partners of COPACOBANA has enhanced and developed successors of COPACOBANA. In 2008 their COPACOBANA RIVYERA reduced the time to break DES to less than one day, using 128 Spartan-3 5000's. Currently SciEngines RIVYERA holds the record in brute-force breaking DES utilizing 128 Spartan-3 5000 FPGAs.

Attacks faster than brute-force

There are three attacks known that can break the full sixteen rounds of DES with less complexity than a brute-force search: differential cryptanalysis (DC), linear cryptanalysis (LC), and Davies' attack. However, the attacks are theoretical and are unfeasible to mount in practice; these types of attack are sometimes termed certificational weaknesses.

- Differential cryptanalysis was rediscovered in the late 1980s by Eli Biham and Adi Shamir; it was known earlier to both IBM and the NSA and kept secret. To break the full 16 rounds, differential cryptanalysis requires 2^{47} chosen plaintexts. DES was designed to be resistant to DC.
- Linear cryptanalysis was discovered by Mitsuru Matsui, and needs 2^{43} known plaintexts (Matsui, 1993); the method was implemented (Matsui, 1994), and was the first experimental cryptanalysis of DES to be reported. There is no evidence that DES was tailored to be resistant to this type of attack. A generalisation of LC — *multiple linear cryptanalysis* — was suggested in 1994 (Kaliski and Robshaw), and was further refined by Biryukov et al. (2004); their analysis suggests that multiple linear approximations could be used to reduce the data requirements of the attack by at least a factor of 4 (i.e. 2^{41} instead of 2^{43}). A similar reduction in data complexity can be obtained in a chosen-plaintext variant of linear cryptanalysis (Knudsen and Mathiassen, 2000). Junod (2001) performed

several experiments to determine the actual time complexity of linear cryptanalysis, and reported that it was somewhat faster than predicted, requiring time equivalent to 2^{39}–2^{41} DES evaluations.

- *Improved Davies' attack*: while linear and differential cryptanalysis are general techniques and can be applied to a number of schemes, Davies' attack is a specialised technique for DES, first suggested by Donald Davies in the eighties, and improved by Biham and Biryukov (1997). The most powerful form of the attack requires 2^{50} known plaintexts, has a computational complexity of 2^{50}, and has a 51% success rate.

There have also been attacks proposed against reduced-round versions of the cipher, i.e. versions of DES with fewer than sixteen rounds. Such analysis gives an insight into how many rounds are needed for safety, and how much of a "security margin" the full version retains. Differential-linear cryptanalysis was proposed by Langford and Hellman in 1994, and combines differential and linear cryptanalysis into a single attack. An enhanced version of the attack can break 9-round DES with $2^{15.8}$ known plaintexts and has a $2^{29.2}$ time complexity (Biham et al., 2002).

Minor cryptanalytic properties

DES exhibits the complementation property, namely that

$$E_K(P) = C \Leftrightarrow E_{\overline{K}}(\overline{P}) = \overline{C}$$

where $\overline{x}$ is the bitwise complement of x. E_K denotes encryption with key K. P and C denote plaintext and ciphertext blocks respectively. The complementation property means that the work for a brute force attack could be reduced by a factor of 2 (or a single bit) under a chosen-plaintext assumption.

DES also has four so-called *weak keys*. Encryption (E) and decryption (D) under a weak key have the same effect:

$E_K(E_K(P)) = P$ or equivalently, $E_K = D_K$

There are also six pairs of *semi-weak keys*. Encryption with one of the pair of semiweak keys, K_1, operates identically to decryption with the other, K_2:

$$E_{K_1}(E_{K_2}(P)) = P \text{ or equivalently, } E_{K_2} = D_{K_1}.$$

It is easy enough to avoid the weak and semiweak keys in an implementation, either by testing for them explicitly, or simply by choosing keys randomly; the odds of picking a weak or semiweak key by chance are negligible. The keys are not really any weaker than any other keys anyway, as they do not give an attack any advantage.

DES has also been proved not to be a group, or more precisely, the set $\{E_K\}$ (for all possible keys K) under functional composition is not a group, nor "close" to being a group (Campbell and Wiener, 1992). This was an open question for some time, and if it had been the case, it would have been possible to break DES, and multiple encryption modes such as Triple DES would not increase the security.

It is known that the maximum cryptographic security of DES is limited to about 64 bits, even when independently choosing all round subkeys instead of deriving them from a key, which would otherwise permit a security of 768 bits.