Code Making & CODE BREAKING

PART 3

In this last part, Mike Bedford takes a look at some of the sophisticated ciphers which are in use today, such as the banks and corporations world-wide, and which will pave the way to the widespread adoption of e-commerce.

So far in our investigation of code making - or cryptography to use the proper terminology - we've seen a progression from ciphers which can be used by hand to those which required machines to automate the encryption and decryption processes. We've also seen a trend towards automation in code breaking or cryptanalysis. Specifically, we saw how the first ever electronic computers were designed for the sole purpose of cracking German wartime ciphers - and these two trends continue hand-in-hand. As improving technology permits ever more sophisticated ciphers to be used, so the code breakers also turn to increasingly advanced technology. As cryptanalysis goes hi tech, then so the code making has to become cleverer still. But, whereas 50 years ago the hardware used for cryptography was rare, expensive, and protected by the Official Secrets Act, today it is commonplace. Any PC can be used to emulate the wartime cipher machines like Enigma or perform cryptanalysis in just the same way as the Bombe electro-mechanical or the Colossus electronic computers. And so there's been another trend in the world of cryptography. Since the tools of the trade are now easy to get hold of, ciphers are no longer the sole domain of spies and the military. Encryption is now an important commercial tool, especially as the potentially insecure Internet is being used to transmit sensitive information such as credit card details. This month, to bring our investigation of cryptography to a close, we'll take a look at some of the sophisticated ciphers which are in use today. These are the ciphers which are used by banks and corporations world-wide and which will pave the way to the widespread adoption of e-commerce.

Data Encryption Standard – DES

The Data Encryption Standard, or DES, was developed during the 70s in response to a US government initiative for the provision of a cheap, secure encryption technology for use in the protection of non-classified information, which would be available to all and appropriate for use in a wide variety of applications.

DES is described as a secret key, block product cipher. We've already come across the term 'product cipher' - one which uses a combination of substitution and transposition - but, if your knowledge of ciphers is limited to what you've read in the first two articles in this series, the other terms will be new to you. All the ciphers we've seen so far are stream ciphers, that is ciphers which operate on a single letter at a time. In a block cipher, the algorithm operates on a block of data at a time and in DES that block is 64 bits or eight ASCII characters long. The final term, secret key, is an intriguing one. It means exactly what it sounds like - that the sender and the receiver use a key which is kept secret from all other parties in order to maintain secure communication. Surprisingly, though, there are other types of cipher where the key is not kept secret but we'll have to leave an explanation of this until later in the article.

The DES algorithm isn't especially difficult to understand - it can readily be implemented in hardware, and a high level language program runs only to around 300 lines - but it is a rather long and protracted process. In our description of DES, therefore, I shall rely heavily on flow diagrams. I suggest that you refer to the diagrams as you work your way through the textual description. Figure 13 is the overall flow diagram of the DES encryption process and Figure 14 is a more detailed view of what happens in each of the boxes in Figure 13 which are labelled 'Iteration 1' to 'Iteration 16.' Note that in each of these diagrams, blue boxes represent the data as it progresses through the process, changing from plain text to cipher text en route, yellow boxes represent keys, and green boxes represent operations. To give an example from Figure 13, the operation labelled 'Iteration 1' takes data L0, R0 and key K1 as its inputs and generates data L1 and R1 as its outputs. You'll notice that the flow diagrams show 16 keys labelled K1 to K16, each of which is used in one of the iterations. In fact, DES requires just the one 56-bit key but this is used to generate the sixteen 48-bit keys used in each of the iterations. We'll take a look at how these sixteen keys are generated once we've worked through the main encryption process.

Initial Permutation

As we've already seen, DES works on blocks of 64-bits. The first stage in the encryption process is a transposition - this is designated by the box labelled 'Initial Permutation'. This involves jumbling up the 64 bits according to a fixed transposition table. The 64-bit result is then divided into 32-bit left and right parts, L0 and R0 which are processed in the first iteration to give another pair of 32-bit parts, L1 and R1. This continues until, after the 16th iteration is

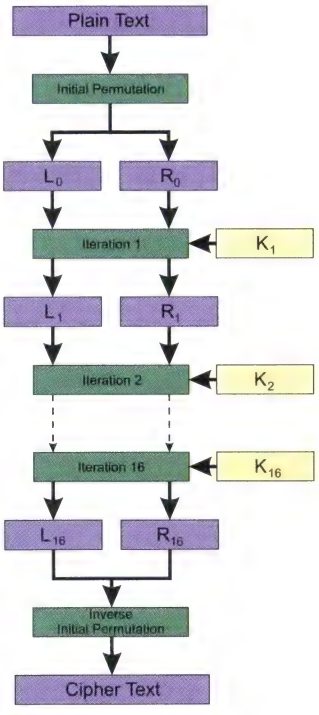

Figure 13.

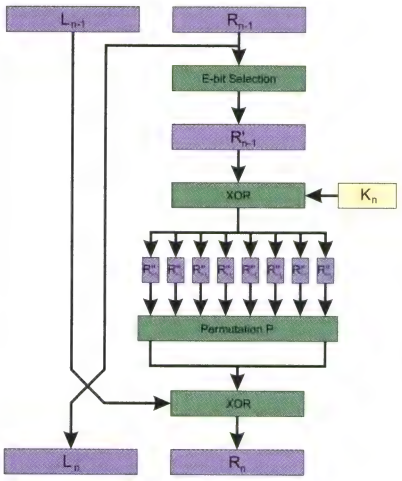

Figure 14.

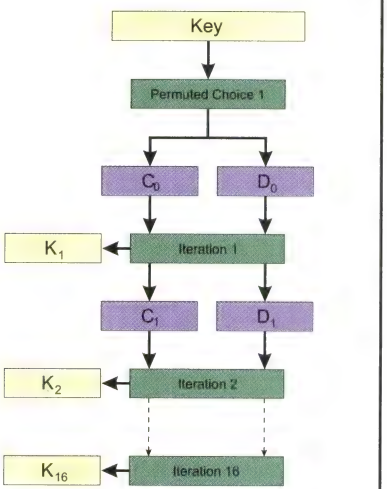

Figure 15.

complete, L16 and R16 are re-combined and finally subjected to the Inverse Initial Permutation - another transposition, in fact the reverse of the first transposition - to give the final cipher text.

So, now let's take a look at what goes on in each of those 16 iterations. The right-hand part of the input data to iteration n, Rn-1, is transposed using another fixed table call the E-bit selection table. In fact, 16 of the bits are also duplicated as part of this process, thereby increasing the length to 48 bits. The result is now XORed with the 48-bit key for this particular iteration, Kn. The 48-bit result is now split into eight groups of six bits each. The box labelled Permutation P is, in fact, a fixed substitution so each of these 6-bit values is substituted for a different value, each now 4-bits in length. The eight 4-bit values are now re-combined to give a 32-bit result, and this is now XORed with the left hand part of the input data to iteration n, Ln-1, to give the right hand part of the output data from the iteration, Rn. The left-hand part of the output data from iteration n, Ln, is simply the right hand part of the input data, Rn-1.Turning now to the generation of the Keys for each iteration, K1 to K16, we'll refer to Figures 15 and 16. As before, the first figure is an overall flow diagram and the second shows the detail of what goes on in each iteration. I won't go through these in detail – I'll just point out that the boxes labelled 'Permuted Choice 1' and 'Permuted Choice 2' involve a transposition of bits and that the second of these also discards some of the bits to produce Kn, the 48-bit key for iteration n.

We haven't presented all this because we expect that many of you will feel inclined to write your own DES software - and you'd need the details of each of the transposition and substitution tables if you were to try. The reason for showing you the flow diagrams is to illustrate that, despite being somewhat convoluted, the process is really just a block-oriented extension of techniques we've already see. This contrasts with the RSA cipher which we'll look at next. But at this stage it's appropriate to address the question of whether this is a monoalphabetic or a polyalphabetic cipher. After all, we saw in the first part of this series that any monoalphabetic cipher - one which always causes a particular plain text letter to become the same cipher text letter - is not very secure. Specifically, it can be cracked by measuring the frequency of occurrence of cipher text letters, letter pairs and triplets and comparing them to the frequency of occurrence in the English language. At first sight, DES does appear to be monoalphabetic – with the same key, one block of input data will always produce the same block of output data. However, the phrase monoalphabetic doesn't apply since the algorithm operates on blocks of 64-bits or eight characters at a time rather than on a single letter. Whereas a block will always encode the same way, if that block was composed of eight letter as, for example, those eight letters wouldn't all end up represented by the same eight bits in the final cipher text. Any attempt to crack a code by looking at the frequency of occurrence of blocks of eight characters - the equivalent of the method we used to crack monoalphabetic ciphers - just isn't an option.

Public Key Ciphers

DES is described as a private key cipher as, for that matter, are all the ciphers we've seen so far. And in a private key cipher, only those parties authorised to encrypt or decrypt messages should have knowledge of that key. This seems so obvious and so much a part of all we've seen so far that it will probably come as a surprise to learn that there is a different class of cipher called the public key cipher. But before we look at this in more detail, let's give some thought to the problems with private key ciphers. Let's assume that I want to communicate with you over a communication channel which I know isn't secure. We've never communicated before so we'll need to agree on some method of encryption. OK, perhaps I take the initiative here so I decide on the DES encryption standard and I pick a key. But now, of course, there's a problem. How do I let you know the key I've picked - certainly I can't transmit it to you over that insecure communication channel. In practice, the only really secure method would be for me to hand the key to you in person. But for someone who needs to communicate with a number of parties world-wide, communicating the keys would be a lengthy and expensive process. This problem of key distribution is the one which public key encryption is designed to overcome.

Another name for a private key cipher is a symmetrical key cipher. What this means is that the same key is used for encryption and decryption as shown in Figure 17. A public key cipher, on the other hand, is asymmetrical in that one key is used for encryption and a different key is used for decryption. This scheme is illustrated in Figure 18. The two keys are, obviously, related, but the cipher is designed such that someone can't work out the one key from the other. Strictly speaking, of course, it is possible to determine one key from the other but it would take a phenomenal amount of computing time. When I talk of something being impossible, therefore, what I really mean is that it's computationally impractical. So let's now look at how this helps overcome the problem of key distribution. All parties intending to communicate using public key encryption generate a pair of keys. Note that, although it's not possible to work out one key from the other, it is, of course, possible to generate a pair of keys with this strange relationship. One of the keys is kept secret which is called the private key and is never divulged to anyone else, and the other key is published. It would be possible, for example, for people to publish their public key in a directory of e-mail addresses. Now, if I want to send a message to you, I encrypt it using your public key. Only you will now be able to decrypt that message since the decryption process requires your private key. Similarly, to reply, you would encrypt the message using my public key and I would decrypt it using my private key. Only the public keys have to be distributed and there's no need to keep these secret, so the problem of key distribution has been solved.

RSA Public Key Cipher

The most common public key cipher is called RSA in recognition of its three developers, Rivest, Shamir and Adleman. I'm not going to describe the inner workings of the RSA cipher at all. Unlike DES, which is

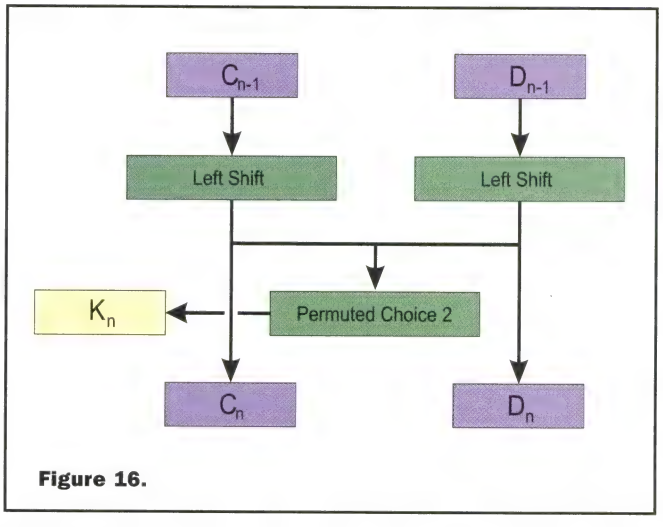

Figure 16.

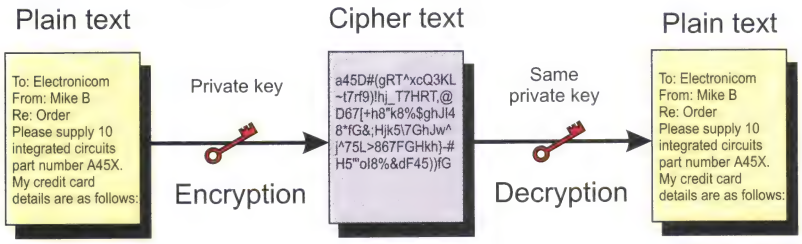

Figure 17.

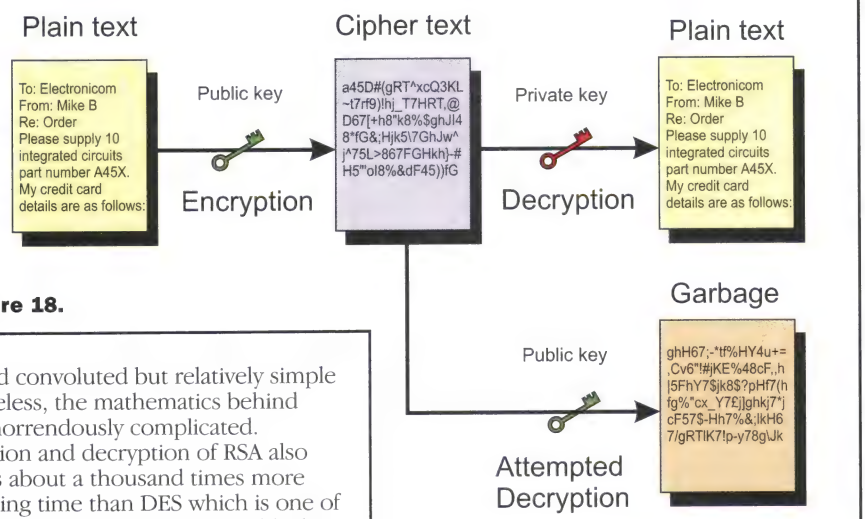

Figure 18.

long and convoluted but relatively simple nonetheless, the mathematics behind RSA is horrendously complicated. Encryption and decryption of RSA also involves about a thousand times more computing time than DES which is one of the major disadvantages of the public key approach. For this reason, a common technique is to use a public key scheme for the distribution of a key which is then used with a separate private key cipher. The private key would be used for just the one message and then discarded. Figure 19 shows this hybrid method in use.

At first sight, it would seem that another disadvantage of a public key cipher is that it doesn't allow authentication of the sender. With a private key cipher, if you receive a message, purportedly from a known party, the fact that you're able to decode it using a key known only to you and that party would indicated that it had indeed been sent by that party. As it stands, of course, no such authentication is provided by a public key cipher. Anyone could encode a message to you using your public key and you'd have no way of telling whether the message actually came from whom it claims to have been sent by. However, a well designed public key cipher, such as RSA, does allow a sender to electronically 'sign' a message in a unique way. Let's see how this works by referring to Figure 20. First of all, though, we need to take a look at the concept of a one-way hash function. This is a function which operates on a block of data to generate a unique value but for which there is no reverse function which would allow the original data to be recovered. An example is the checksum which is appended to the end of a block of data to allow the receiver to verify whether or not the data had been corrupted during transmission. The receiving party calculates the checksum from the received data and compares it with the checksum calculated by the sender and appended to the message. A discrepancy indicated data corruption. So, to return to the digital signature, the sending party encrypts the message using the intended recipient's public key using the method we've already seen. The sender also performs a one-way hash function on the unencrypted text and encrypts the result of the hash function using his own private key. Now, RSA allows messages either to be encrypted using the public key and decrypted using the private key or vice versa. Encrypting using the private key is not normally very useful since anyone could then decrypt that message using the public key but it is useful in this instance. The recipient decrypts the message using his own private key and then performs the one-way hash function on the decrypted message. The recipient now decrypts the hash function result sent by the other party using that party's public key. If the result is the same as the locally-generated hash function result, this proves that the message had been sent by the party whose public key had been used to decrypt the hash function result. The sender is, therefore, authenticated.

Political Wranglings

In the section on DES, we spoke about a key length of 56 bits. In fact, there has been quite some controversy over the years about the key length of this and other encryption algorithms. The standard which formed the basis of DES, IBM's Lucifer, had a 128-bit key. Before authorising it for public release, though, the National Security Agency – the USA's equivalent of MI5 – insisted on a reduction to 56 bits. As you could well imagine, this made the cipher substantially less secure, in fact, a brute force attack would manage to crack the 56-bit DES a few trillion times quicker than the full 128-bit variant. However, as you'll know if you've followed the political tos-and-fros reported in the computer magazines, the US government imposed even more stringent limitations on what could be exported. Encryption software is considered to fall into the same category as munitions for export purposes. Accordingly, such software can only be shipped from the USA with an appropriate export license. Licenses were, initially, only granted for products with a 40-bit keys - another 64,000 times less secure than the 56-bit version.

The fact that something is illegal, doesn't always prevent it from happening, though, as is evidenced by the Zimmermann saga. American mathematician Phil Zimmermann, developed a system of cryptography called PGP or Pretty Good Privacy which had a 128-bit key and could not, therefore, be exported legally. This didn't deter Zimmermann, though, and, in common with much of the early Internet community, felt that a stand had to be made in the interests of free speech. That stand involved making the source code of PGP freely available over the Internet and it came very close to gaining Zimmermann a four year spell in a Federal Penitentiary. Another line of attack against apparently unjust export restrictions was totally legal. Academics and hackers everywhere started a consorted attack on 40-bit versions of DES and similar ciphers. Some of the cryptanalysts were flying the 'free speech' flag by showing how insecure a 40-bit key was, for others the motivation was the pure technical challenge of cracking a cipher, but it's also significant that for many others the motivation was financial – cash prizes were on offer. And who was putting up this money? Well, if I tell you that the export restrictions were causing American software houses to loose out to foreign competition, you'll probably guess the answer – these companies had a vested interest in proving to the NSA that a 40-bit cipher is virtually useless. So how did these amateur cryptanalysts fare? In 1995, French student Damien Doligez cracked a 40-bit cipher in eight days using a combination of 120 workstations and a few supercomputers. The next year, a group of cryptographers estimated that a 40-bit key could be cracked in 12 minutes and a 56-bit key in 18 months using a $10,000 machine consisting of 25 FPGAs. For $10 million, a machine with 25,000 FPGA chips could crack a 56-bit DES key in 13 hours; one with 250,000 ASICs could do it in 6 minutes. In 1997, following RSA Data Security's announcement of cash prizes for the first person breaking each cipher of varying key lengths, Ian Goldberg, a student at Berkeley, walked away with a $1,000 prize of cracking the 40-bit RC5 cipher in three and a half hours using a network of 250 computers that tested 100 billion keys per hour. A few weeks later, Germano Caronni of the Swiss Federal Institute of Technology won the $5,000 48-bit prize. Caronni used more than 3,500 computers networked over the Internet to search 1.5 trillion keys per hour. The key was found after 13 days.

Well, to cut a long story short, the US government eventually relented about a year ago and encryption systems with key lengths up to 56-bits can now be exported. However, it's questionable how much more secure 56-bit key is today than a 40-bit key was when DES was first introduced. More recently, a team assembled by the Electronic Frontier Foundation cracked the 56-bit DES in less than 23 hours. In fact, a bill which will abandon all export controls on encryption products is in the process of

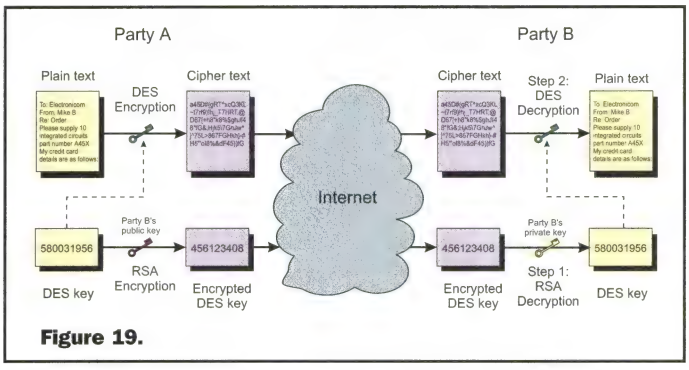

Figure 19.

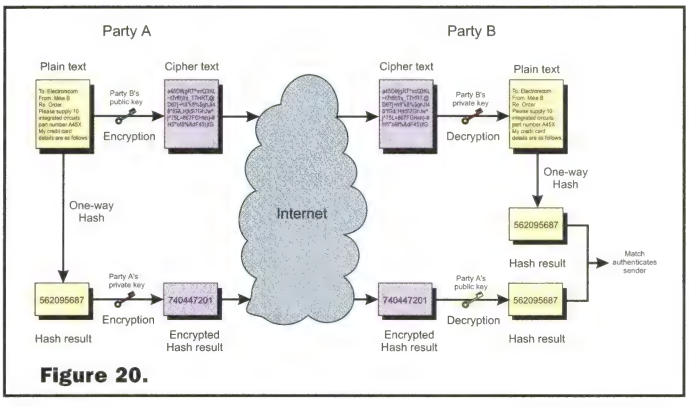

Figure 20.

being introduced. Until recently, though, it looked as if this bill only stood a chance of being approved if it required suppliers of products to build in a key recovery mechanism. Also referred to as 'the back door', this is a method, in theory known only to the supplier and to certain government departments including law enforcement agencies, which would permit a message to be decrypted without use of the key. Needless to say, civil liberty groups voiced opposition to this, as did other parties concerned that the back door could well jeopardise security. So, if you're concerned about civil liberties you'll be dismayed to hear, no doubt, that the British government, in its proposed forthcoming legislation, looks set to require something similar. But instead of a back door, the proposed solution is to use the services of a so-called Trusted Third Party. Users of encryption products would be required to lodge their key with this trusted third party. In the event of an investigation relating to criminal or terrorist activity, the police and other government departments would be ably to apply to obtain the key from the trusted third party.

Hands-on Encryption

To conclude our investigation of the world of cryptography, I thought it would be appropriate to give pointers on where you can get hold of cryptographic software. I'm not talking here of software which could be described as a curiosity - software to illustrate the use of the various historical ciphers we looked at earlier in this series - I'm talking about software which can be used for real world applications. However, if you do want to play around with some of the historical ciphers, search engines will list a wealth of information. Bear in mind, though, that if you just look for words such as cipher, encryption, cryptography and the like you'll get just too many references and most of them will be to current day cryptography. So try looking for information on specific historical ciphers such as Playfair or Enigma.

But to return to my main emphasis, practical cryptographic software will, almost certainly, be based on the ciphers described in this article, that is DES or RSA. And commercial products are available from companies like Sophos (www.sophos.com) who specialise in computer security. In addition to encryption for the purpose of secure electronic transmission of data, packages which will encrypt data as it's written to your hard disk and decrypt it as it's read back are also available. This, of course, provides protection from unauthorised access to your PC and secures your data in the event of the PC being stolen. However, there is also plenty of public domain software and shareware available for downloading from the Web. If you look around I'm sure you'll find plenty but here are a couple of examples of free software which you may like to take a look at. Do bear in mind, though, that you should properly virus-check any software you download from the Web before you use it. Scramdisk, available from http://www.scramdisk.clara.net, encrypts the data on your hard disk to protect it from prying eyes. The software decrypts data on the fly but only to a user who can enter the appropriate password (i.e. key). And a number of packages for file encryption and secure e-mailing can be downloaded from http://abi.hypermart.net.

Finally, as a parting shot, I thought you might also be interested in taking a look at http://web.mit.edu/network/pgp.html which is the Web site from which PGP - the software which got Phil Zimmermann into so much trouble - can be downloaded. However, you won't actually be able to obtain the software from that site, not unless you make some false declarations, that is, since it is still governed by US export restrictions. Specifically, in order to be provided with complete details of how to download the software, you have to electronically sign a declaration stating either that you're a US citizen living in the USA or that you're a Canadian citizen living in Canada. Personally, I wouldn't risk incurring the wrath of the CIA by making a false declaration.

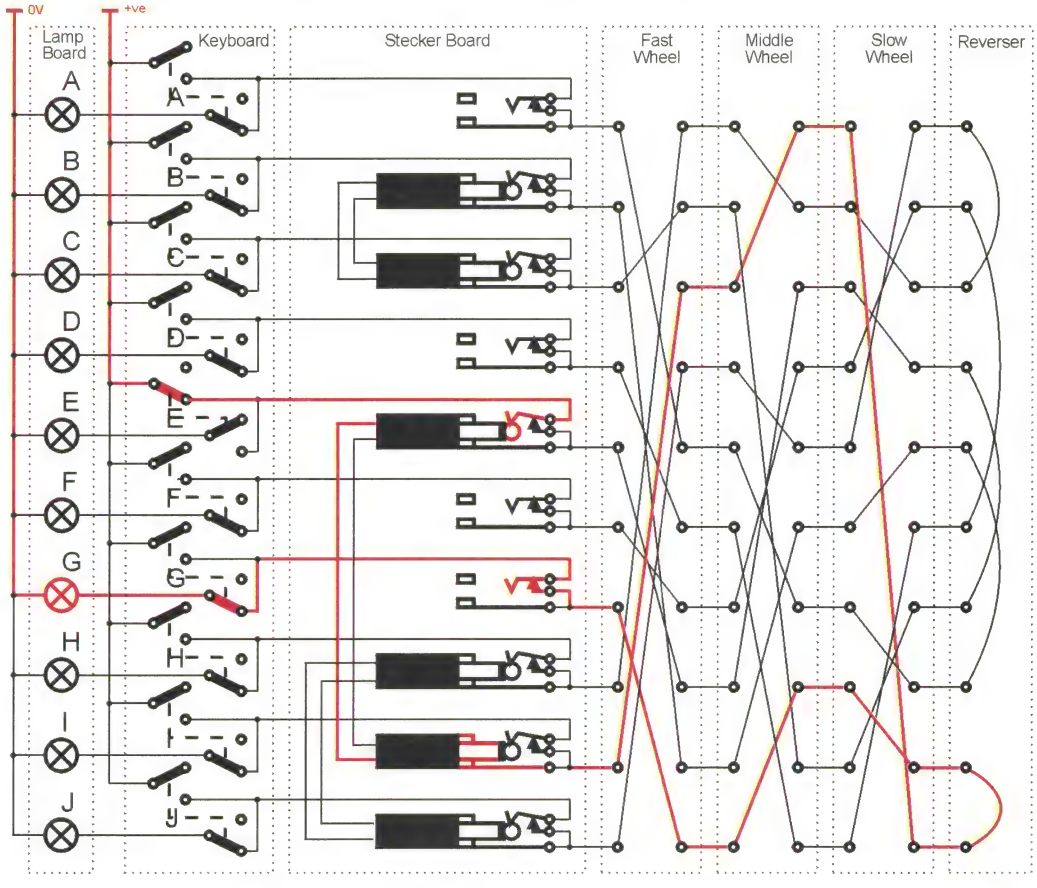

Figure 8. Logical wiring diagram of the German wartime Enigma machine. This is reprinted in colour and applies to the text in Part 2 from last month.