Chapter 9

Pioneering Public Key: Public Exchange of Secret Keys

Let's recap and lay the groundwork to see how a new twist on secret key distribution empowers a new form of cryptography.

Alice and Bob have developed secure secret keys. Alice encrypts her computer files and feels secure that no one can decrypt the files without her individual secret key. Alice and Bob's digital conversations use their shared secret key to authenticate each other, confidentially exchange files, and validate the integrity of the files (ensure that the files have not been altered during transit).

Review: Sharing and distributing secret keys is cumbersome.

But as you saw in Chapter 8, sharing secret keys is difficult and expensive. Alice must either personally deliver the shared secret key to Bob or unequivocally trust a courier. Trustworthy couriers are expensive. And if Bob forgets their shared secret key, Alice must repeat the same key delivery process.

The Search for an Innovative Key Delivery Solution

The secret key delivery problem has plagued cryptographers, governments, and kings for thousands of years. How do you securely deliver a secret key to a confidant using insecure public lines of communication? Although the key may pass through BlackHat's hands, BlackHat must not be able to ascertain the secret key. It's a tough problem. What advantage does Bob have over BlackHat that Bob can exploit?

Developing an Innovative Secret Key Delivery Solution

While a graduate student at Berkeley in the early 1970s, Ralph Merkle devised a system that enabled people like Alice and Bob to exchange secret keys over a public line, marking the beginning of public key cryptography. Even though

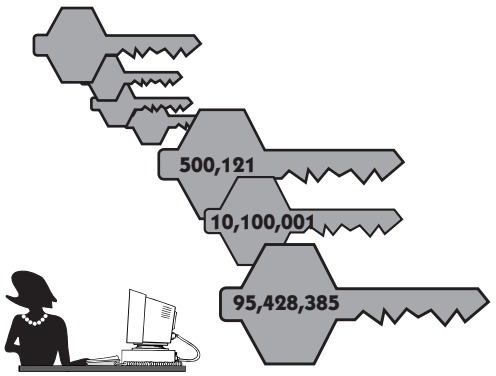
BlackHat is assumed to be listening to Alice and Bob's communications, Merkle envisioned a way to create a difficult, time-consuming problem for BlackHat. At the same time, Merkle's approach makes it easier for Alice and Bob to establish their shared secret key.

The goal was to create a problem that would take BlackHat a long time to solve even with the aid of a computer. Here's what Merkle devised.

First Attempt: A Database of Key/Serial Number Pairs

Suppose that Alice makes 1,000,000 new secret keys and stamps a unique serial number on each one (see Figure 9-1). Note that there's no reason to order the serial numbers. Alice keeps a database of each secret key and serial number.

If Alice then sends Bob a plaintext electronic copy of that database, he can easily pick a serial number (say, serial number 500,121) and its paired secret key (1yt8a42x35); then he calls Alice to tell her to use the secret key associated with serial number 500,121. But as Figure 9-2 shows, what's easy for Alice is also easy for the eavesdropping BlackHat. BlackHat has copied the database Alice sent to Bob—remember that it was sent over public lines—and quickly figures out the secret key. So this secret key exchange doesn't work for Alice and Bob.



Secret Key	Serial #
aiet187f45	1,287,341
9dsyh3701	77,783,902
	…
1yt8a42x35	500,121
	…
43879d323	10,100,001
	…
25s42fds70	95,428,385

Figure 9-1 Alice makes 1,000,000 secret keys.

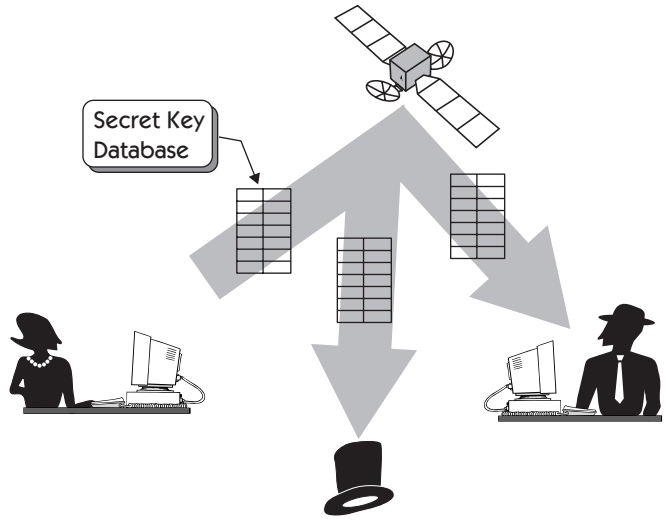


Figure 9-2 Alice sends Bob a file of secret keys and serial numbers, but BlackHat copies it and learns their secrets.

Second Attempt: An Encrypted Database of Key/Serial Number Pairs

Alice encrypts the entire database of serial number/ secret key pairs.

Now suppose that Alice encrypts her serial number/secret key database and then ships the encrypted database to Bob. Alice tells Bob the encryption method she used but not the encryption key. Because Bob doesn't know Alice's encryption key, he must try all possible keys. Say it takes Bob about one hour, using his desktop computer, to find Alice's encryption key.[1] After decrypting the entire database, Bob selects a secret key and tells Alice the matching serial number—say, serial number 500,121. As before, Alice knows to use secret key 1yt8a42x35 (see Figure 9-3). But, like Bob, BlackHat can also spend about an hour decrypting Alice's database, so BlackHat can also figure out that serial number 500,121 matches secret key 1yt8a42x35. We still need a method that will make BlackHat's job much tougher than Alice and Bob's job.

1. Obviously, Alice does not choose a "strong" cryptographic method to encrypt her database. Recall from Chapter 4 that a strong encryption method is one in which the most practical attack is to try each possible key and there are so many possible keys that it's infeasible to try even half of them.

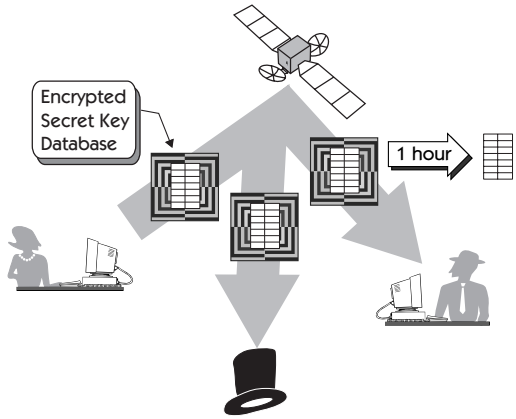


Figure 9-3 Alice sends an encrypted database of secret key and key pairs, but BlackHat isn't intimidated.

Merkle's Insight: Individually Encrypted Key/Serial Number Pairs

Third approach: Encrypt each individual serial number /secret key pair.

This brings us to Merkle's creative insight. As the result of this innovation, Bob's work doesn't change, but BlackHat's work increases dramatically. Let's see how.

Previously, Alice encrypted the entire database with one secret key. But that didn't make BlackHat work longer than Bob. Now Alice sends 1,000,000 encrypted secret key/serial number pairs (see Figure 9-4).

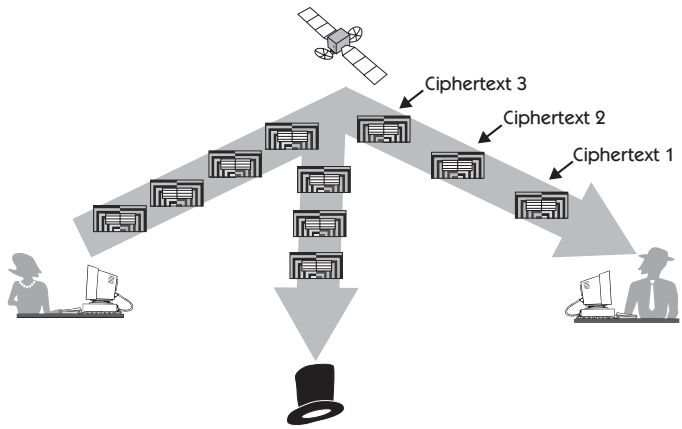


Figure 9-4 Alice sends Bob 1,000,000 encrypted secret key/serial number pairs. BlackHat eavesdrops and copies the key pairs sent to Bob.

Each secret key/serial number pair is encrypted with a different secret key.

Each secret key/serial number pair (second column, Table 9-1) is encrypted with a unique secret key (third column, Table 9-1) to make the encrypted pair (final column, Table 9-1). Alice uses a million *different* secret keys to encrypt the 1,000,000 individual secret key/serial number pairs. Table 9-1 shows each secret key/serial number pair encrypted with a separate key.

Bob gets 1,000,000 encrypted secret key/serial number pairs and picks one encrypted pair—say, Pair3. He spends an hour deciphering it and learns that Pair3 means secret key 1yt8a42x35 and serial number 500,121 (see Figure 9-5). As before, he tells Alice that he will encrypt with the secret key matching the serial number 500,121. Alice quickly matches the serial number to the corresponding secret key in her database.

As before, Alice and Bob assume that BlackHat is listening, has copied all 1,000,000 encrypted pairs Alice sent to Bob, and has heard Bob tell Alice to use the secret key associated with serial number 500,121.

Black Hat's Frustrating Problem

BlackHat must decrypt many more serial number/secret key pairs than does Bob.

How does BlackHat figure out the secret key Alice and Bob agreed to share? BlackHat doesn't know that Bob selected Pair 3; he knows only that one of the encrypted pairs Alice sent to Bob contains serial number 500,121. But here's the rub: BlackHat doesn't know which pair. It must be one of the million pairs, but he has no clue which one (see Figure 9-6).

Table 9-1 Alice's database of secret keys and serial numbers, encryption key, and encryption message sent to Bob (and snooped by BlackHat).

Alice Makes and Keeps			Alice Sends to Bob
Pair Number	**Plaintext of Secret Key / Serial Number (reproduced from Figure 9-1)**	**1,000,000 Different Secret Keys—One for Each Pair Number**	**Encrypted Text of Secret Key/Serial Number**
1	aiet187f45 / # *1,287,341*	1	**Ciphertext 1**
2	9dsyh3701 / # *77,183,902*	2	**Ciphertext 2**
3	1yt8a42x35 / ***# 500,121***	3	**Ciphertext 3**
. . .	. . .	. . .	. . .
900,000	43879d323 / # *10,100,001*	900,000	**Ciphertext 900,000**
. . .	. . .	. . .	. . .
1,000,000	25s42fds70 / # *95,428,385*	1,000,000	**Ciphertext 1,000,000**

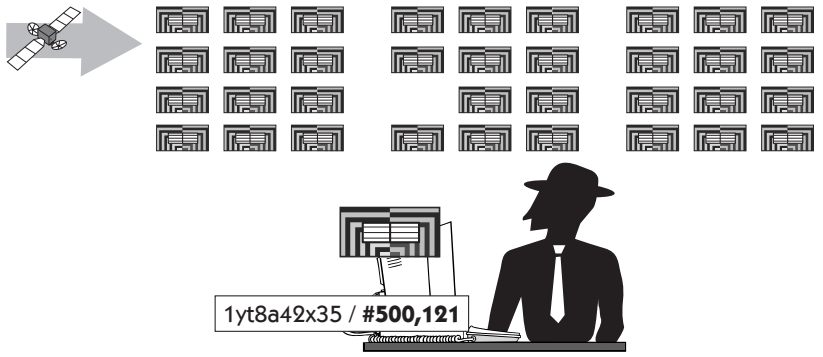


Figure 9-5 Bob picks one encrypted pair and decrypts it to learn the secret key and serial number.

Recall that Bob tells Alice only the serial number he learned. BlackHat has a much bigger problem than Bob: He must decrypt about half the encrypted pairs Alice sent to Bob until he stumbles onto the one pair that decrypts to 1yt8a42x35 / 500,121.

With this twist, Merkle turned a relatively simple problem for Bob into a time-consuming problem for BlackHat. If deciphering one encrypted pair takes about one hour and if BlackHat must try, on average, about 500,000 of them, BlackHat has a 500,000-hour problem. Bob has only a one-hour problem.

As a result, Alice and Bob can communicate confidentially with their shared secret key while BlackHat is busy trying to figure out which secret key they are using.

The Key to Public Key Technology

Easy and difficult puzzles

All public key cryptography works in the same way. You and your confidant share a quickly solved problem. Let's call it an easy puzzle. But if you withhold a critical puzzle piece from your adversary, the easy puzzle is transformed into

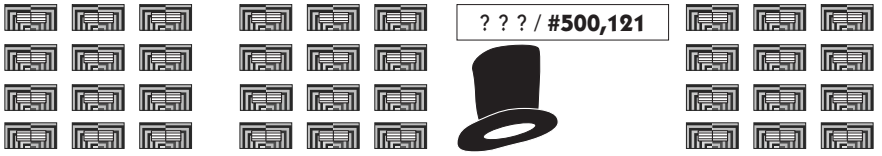


Figure 9-6 BlackHat has a big problem. He knows that one of the 1,000,000 pairs he copied contains the pair Alice and Bob are using; but because he doesn't know which one, he must try them one by one. According to probability theory, he'll have to try about half of them before he stumbles onto Bob's choice.

a difficult, time-consuming puzzle. In Merkle's innovative implementation, Bob solves one encrypted pair and finds a secret key to share with Alice. BlackHat doesn't know and isn't told which encrypted pair Bob solved. That critically withheld puzzle information forces BlackHat to solve, on average, 500,000 encrypted pairs—a much more time-consuming puzzle.

Technology reduces the perceived advantage.

A ratio of 500,000 to 1 may seem impressive, and certainly after 500,000 hours (or about 50 years) Alice and Bob's secrets have much less value. A delay of 50 years may seem sufficient to protect secrets, but more powerful computer hardware and more efficient software can dramatically reduce that number.

Bob may use a $1,000 computer to solve a puzzle sent to him by Alice, but an adversary—wanting to learn about, let's say, the next major West Coast land deal, might use a $10,000,000 computer. A computer 10,000 times more expensive and 10,000 times faster than Bob's desktop might solve the 50-year problem in 50 hours. With better software, it might take even less time.

Cryptographic history is replete with stories of technological advances eliminating cryptographic advantages. As a result, cryptographers prefer a much

Hiding in Plain Sight

Just as Merkle designed a way to keep an electronic adversary busy looking through a maze of possibilities, so, too, did spies design real-world mazes to hide the locations of *dead drops*: prearranged hidden places for leaving and retrieving messages and money. A good location was one that let you hide your message in plain sight, a hiding place that was hard to recognize unless you had secret information.

One of the more elaborate hiding places in the history of espionage helped the U. S. Central Intelligence Agency (CIA) retrieve valuable information about the Cuban missile crisis of the 1960s from an insider in Soviet military intelligence. The informant gave the CIA specific instructions about how to find the information hidden in a Moscow building—a task that would be hard for someone without the instructions or who had never directly observed the exact location.

His instructions were to go to a particular foyer near the number 28 telephone. Across the hallway was a dark green radiator held to the wall by a single metal hook. Between the wall and the radiator was a space 2 to 3 centimeters wide that was to be used as the drop. The selected hiding place made it easy for both parties to deposit and retrieve materials while in a standing position. In this way, someone observing the scene would be less likely to notice the drop.

Merkle envisioned his public distribution of secret keys in a similar way: Make it hard for BlackHat to find your secret key by giving him a much bigger search space than Bob's.

larger advantage: at least 500,000,000 to 1, which is about 1,000 years compared with 1 minute. One way to accomplish that is to give BlackHat a much bigger search space. For example, Alice could send a greater number of encrypted pairs, but that's an inefficient use of bandwidth. Only one of the 1,000,000 encrypted pairs is used; the other encrypted 999,999 pairs just cloak the selected one and are thrown away.

A New Solution: Diffie-Hellman-Merkle Key Agreement

Unlike Merkle's initial idea to solve the problem of key delivery, the first patented public key cryptographic system used math to achieve the overwhelming ratio needed to ensure that BlackHat would stay busy for a long time. During the 1970s, Merkle took his ideas to Martin Hellman of Stanford, where Merkle found a more sympathetic training ground for public key cryptography.[2] Hellman was joined by another cryptographic kindred spirit, Whitfield Diffie, who drove across the United States to meet him.

Like Thomas Edison, who tried thousands of materials before arriving at the combinations that made the light bulb possible, the Diffie-Hellman-Merkle trio tried all sorts of ways to solve the problem. Their persistence paid off. After two years of focusing on modular arithmetic and one-way functions[3]—problems that were harder to work one way than the other way—the men developed an open (that is, public) secret key agreement solution.

The first workable public key cryptographic system, the patented Diffie-Hellman-Merkle key agreement scheme (more commonly known as Diffie-Hellman, or DH), is still widely used in Internet browsers such as Secure Socket Layer (see Chapter 20) and Internet Protocol Security (see Chapter 21). Like Merkle's original approach, Diffie-Hellman confidants use insecure public communication lines to agree on a shared secret key.

Alice and Bob Openly Agree on a Secret Key

In Merkle's original approach, Alice sent Bob 1,000,000 potential secret keys and Bob randomly selected one. In Diffie-Hellman, as incredible as it sounds, Alice and Bob make the shared secret key, online, together, in full view of BlackHat—or any other eavesdropper. (The math, not important for this discussion, is shown in Appendix A.)

2. As the story goes, Merkle's Berkeley professor couldn't understand Merkle's ideas and Merkle dropped the course.
3. Modular math and one-way functions are discussed in Chapters 11 and 14, respectively.

The Diffie-Hellman method of secret key agreement makes secret key distribution much easier.

What's important is the breakthrough that allows Alice and Bob to openly exchange some Diffie-Hellman numbers. Then, in private, they use each other's DH numbers, their secret random numbers, and the DH method to agree on the same secret key.

Although BlackHat knows the DH method and copies Alice and Bob's conversation, he cannot figure out Alice and Bob's agreed-on secret key. Figure 9-7 shows how Alice and Bob create and exchange a secret key over a public line. BlackHat listens to every communication between Alice and Bob.

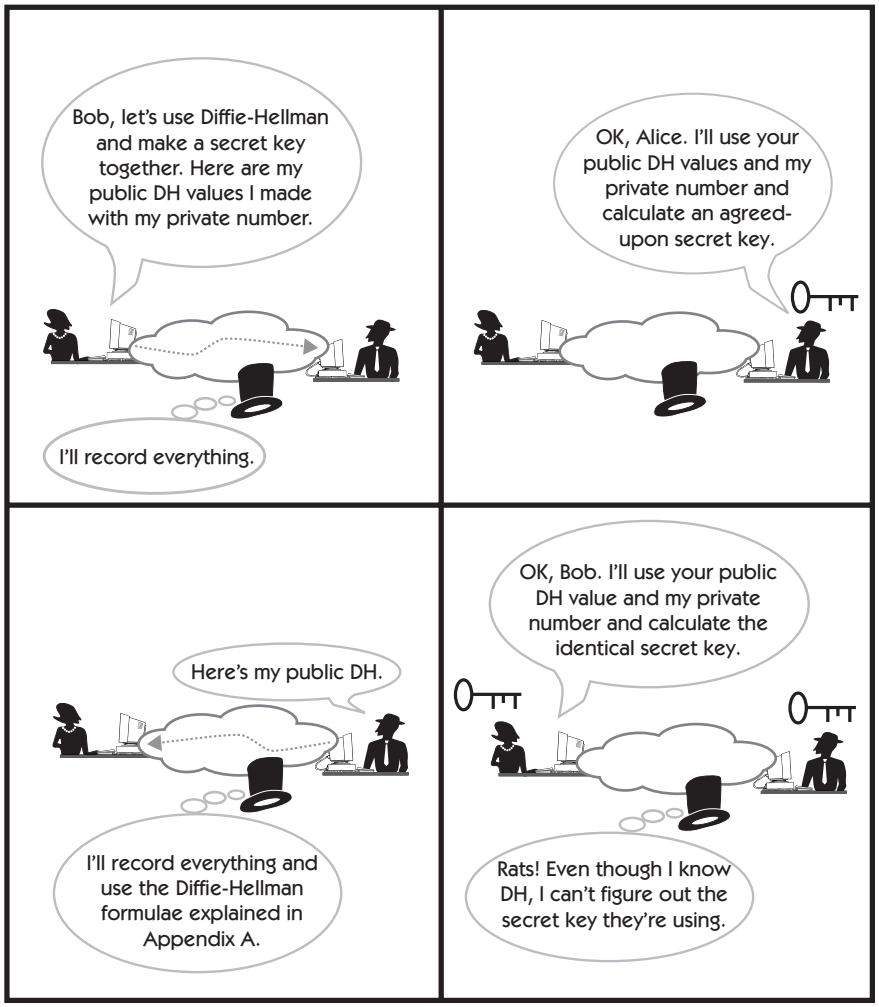


Figure 9-7 Alice and Bob using the Diffie-Hellman (-Merkle) secret key agreement. BlackHat records their conversation but cannot reproduce their shared secret key.

Problems with the Diffie-Hellman Method

Alice can't use DH to authenticate Bob (and vice versa).

But the DH method has two weaknesses. The first is the lack of authentication. DH, by itself, doesn't give Alice assurance that the person with whom she's agreeing on a secret key is Bob. Rather, it assures Alice that the secret key she calculates is known only to her and to the computer at the other end of the connection. It doesn't stop BlackHat from masquerading as Bob to Alice or masquerading as Alice to Bob.

Second, the DH method fails to provide a versatile means for secret key exchange. Recall that to agree on secret keys, Alice and Bob must exchange some DH values. Although their DH values can be openly exchanged, for some applications, such as e-mail, an online exchange is inconvenient.

Separate Encryption and Decryption Keys

Definitions: symmetric cipher, asymmetric cipher

So far in our discussion, Alice and Bob share the same key, and both of them use the identical secret key to encrypt and decrypt. Systems that use shared secret keys are called *symmetric* cipher methods. Diffie, Hellman, and Merkle also envisioned systems, called *asymmetric* ciphers, that have separate encryption and decryption keys. One key encrypts, and the other one, a different key, decrypts. Diffie's 1976 paper about this system marked the first time in 3,000 years of cryptographic study that anyone outside government institutions knew that such an idea had been conceived.

Diffie's paper ended up in the hands of Ronald Rivest at MIT. Rivest persuaded Adi Shamir and Leonard Adleman, also at MIT, to help him search for the math that would make this idea a reality.

Public Key: A Top Secret Discovery

In 1969, the British military asked British Intelligence at Government Communications Headquarters (GCHQ) to look into a problem with secure military communications. The military could see that miniaturization of radio equipment was going to allow every soldier to be in continual radio contact. But ensuring that such military communications were secure required the distribution of secret keys to all those soldiers, a daunting problem. GCHQ went to work trying to solve the dilemma.

The problem was given to James Ellis, one of Britain's prominent government cryptographers, known by the GCHQ as somewhat eccentric.

(Continued)

But Ellis was also known for his brilliance, which led him to read and collect a broad range of scientific materials. In those materials he found the seed of the idea that was proposed by the Stanford trio: an asymmetric cipher, in which one key encrypted and the other key decrypted. Ellis's brainstorm came after he read a Bell Telephone report written during WWII. To ensure the security of telephone speech, the report's unknown author proposed that the recipient mask the sender's message by adding noise to the line. Because the recipient had added the noise, theoretically the recipient should be able to remove it. It didn't work that way in reality because of the difficulty of removing noise from speech communications. But Ellis applied the noise principle to enciphering text. He suggested that it could be a way of achieving security without exchanging any secrets. Unfortunately, Ellis was not a mathematician. Although he knew he needed a special one-way function that only the receiver could reverse, he didn't have the mathematics background to implement his idea.

Ellis made his idea known to the higher-ups, and for several years GCHQ's brightest minds worked on a practical solution to the problem. The solution came to a novice cryptographer, Clifford Cocks, just six weeks after he joined GCHQ in 1973. Although Cocks knew very little about cryptography, he'd specialized in number theory at Cambridge University before joining British Intelligence.

According to cryptographic historian Simon Singh in *The Code Book,* Cocks claimed that it took him half an hour to solve the mathematical puzzle with prime numbers and factoring, the same solution that has become known as RSA (after Rivest, Shamir, and Adleman). At the time Cocks solved the problem, he had no idea that GCHQ had been working on its solution for years or that he'd discovered one of the most important cryptographic methods ever conceived. At that time, GCHQ couldn't put Cock's discovery into effect because there wasn't yet enough computing power available to make it practical.

NSA claimed it had public key technology before Diffie-Hellman.

In 1974 Malcolm Williamson, a mathematician and long-time friend of Cocks, joined GCHQ. When Williamson heard about Cocks's discovery, he set out to disprove it. Instead, he discovered what the world would soon know as Diffie-Hellman key exchange at about the same time it was being developed across the Atlantic.

By 1975, the British had discovered all the essential components of public key cryptography, but no one was talking. It was top secret. The credit went to the Americans, who commercialized and patented these ideas, which are key to the advancement of the digital revolution.

The RSA method solved the authentication and key exchange problems and thus enabled the assurances needed in our burgeoning digital age.

This powerful system can seem complex if you look at all its capabilities at once. Chapter 10 shows how public/private key pairs provide easy key exchange and confidentiality. Chapter 11 briefly examines the major math trick behind public key cryptography. After that, we look at how public key cryptography (and RSA in particular) goes beyond confidentiality to give us digital signatures needed for e-commerce.

Review

Ralph Merkle helped create the framework for public key cryptography. Despite the drawbacks to his key distribution approach, Merkle designed an ingenious way to securely distribute secret keys over an insecure public communication channel. Although Merkle's initial idea didn't give people the desired competitive advantage over their adversaries, he worked with Martin Hellman and Whitfield Diffie to create a mathematically feasible system. It's now accepted that British cryptographers developed public key cryptography before Diffie, Hellman, and Merkle, even though the Stanford trio were the first to patent a public key system.

The Diffie-Hellman-Merkle public key system (commonly referred to as Diffie-Hellman, or DH), implements the strategy of making a problem that has at least two solution paths: an easy one and a very difficult one. The idea is to give your friends the easy, less time-consuming path and force your adversaries to solve the difficult, more time-consuming version of the same problem. All public key cryptography uses this principle.

The Diffie-Hellman-Merkle key agreement method didn't provide all the versatility and assurances needed to fuel the digital age. Authentication was still a problem. The idea of asymmetric ciphers, in which one key is used to encrypt and another is used to decrypt, solves the problem of authentication. Rivest, Shamir, and Adleman built on the public key foundation built by Diffie, Hellman, and Merkle to create an asymmetric cipher known as RSA.