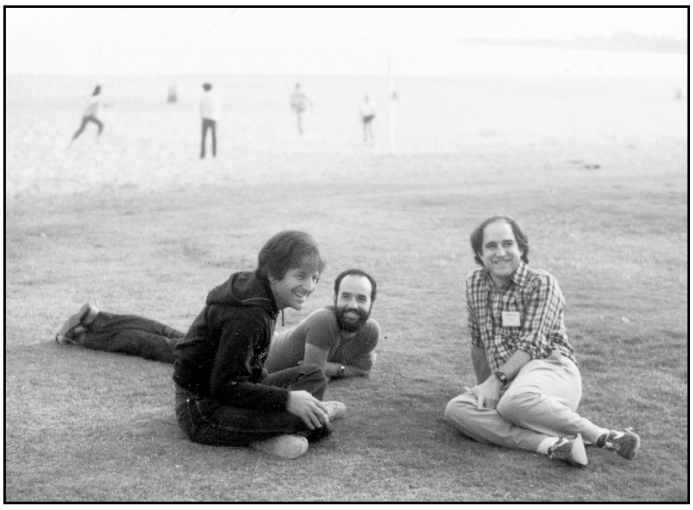
from *SIAM News*, Volume 36, Number 5, June 2003

Still Guarding Secrets after Years of Attacks, RSA Earns Accolades for its Founders

By Sara Robinson



In the late 1970s, computer scientists Ronald Rivest (right) and Adi Shamir (center), with number theorist Leonard Adleman (left), all then at MIT, devised the public key encryption code that bears their initials (and that has been in use ever since to secure electronic transactions). The trio is shown here at CRYPTO '82. Today, Rivest is still at MIT, where he is the Viterbi Professor of Computer Science; Shamir is the Borman Professor in the Applied Mathematics Department at the Weizmann Institute, and Adleman is the Distinguished Henry Salvatori Professor of Computer Science and a professor of molecular biology at the University of Southern California. They will receive the ACM's A.M. Turing Award this month.

In the fall of 1976, three friends, all young faculty members at the Massachusetts Institute of Technology, began working on a new type of cryptographic scheme. The group soon established a dynamic—Ron and Adi would come up with ideas, and Len would try to shoot them down. Len was consistently successful; late one night, though, Ron came up with an algorithm that Len couldn't crack. That algorithm—named RSA for the three developers, Ronald Rivest, Adi Shamir, and Leonard Adleman—remains unbroken to this day.

"It wasn't clear that it would endure as long as it has," Rivest says. "It has a surprising amount of vigor."

Like all practical cryptographic schemes, however, RSA stands on mathe-matically shaky ground. Because its security rests on unproven assumptions, RSA comes with no guarantee that the secrets it guards will remain safe. Still, for close to thirty years, although researchers have found weaknesses in implementations of the algorithm, its core has wea-thered every attack the best minds of cryptography have devised.

The algor-ithm's robustness, even in the absence of rigorous proof, provides a sense of security, researchers say. "We kind of chip at the sides, but no one has figured out how to get at the heart of it," says Dan Boneh, a professor of computer science at Stan-ford University.

Meanwhile, RSA has come to play a central role in electronic communications. As the first example of what is known as a *public key cryptosystem*, and the only one that has stood the test of nearly 30 years of attacks, RSA has become the algorithm of choice for encrypting Internet credit-card transactions, securing e-mail, and authenticating phone calls. This month, in recognition of the theoretical and practical contributions of RSA, Rivest, Shamir, and Adleman will receive the Association for Computing Machinery's 2002 Alan Turing Award, among the highest honors in computer science.

Not surprisingly, the security of RSA has remained a focus of cryptographic research, both theoretical and practical. Some researchers have emphasized the secure implementation of RSA, a fascinating topic in itself. Complexity theorists, on the other hand, have directed their efforts to the theoretical underpinnings of RSA, and the strength of the assumptions on which its security rests.

A Nucleus of an Idea

"We stand today on the brink of a revolution in cryptography."

This was the opening sentence in a landmark 1976 paper (published in the *IEEE Transactions on Information Theory*) by Whitfield Diffie and Martin Hellman, then of Stanford University. The development of computer networks, they wrote, "promises effortless and inexpensive contact between people or computers on opposite sides of the world." In this new world, sensitive information would need to be encrypted before being sent, possibly to strangers. Existing methods of key exchange, courier or registered mail, would no longer make sense.

The two computer scientists went on to propose a new method for secure key exchange, where all the necessary information for the exchange is publicly available. Their key mathematical idea, using ideas from the fledgling field of complexity theory, was to make use of a function that's easy to compute in one direction but computationally hard to invert.

In a finite field with a fixed generator g, it's easy to compute g raised to a power, but no known efficient algorithm, given an element of the field, can figure out what power of g it is. Diffie and Hellman used this property as follows: In a finite field of q elements with a fixed generator g, each user i generates an integer x_i chosen uniformly at random from 0 to $q - 1$. The user keeps x_i secret but then computes $y_i = g^{x_i} \bmod q$ and places it in a public database. Then, if i and j want to exchange information, they derive a cryptographic key from $k_{ij} = g^{x_i x_j} \bmod q$. Since $g^{x_i x_j} = y_i^{x_j} = y_j^{x_i}$, i and j can each compute k_{ij}, using his/her private key and the other person's public key, by exponentiating, which is com-putationally easy. But it seems that no other user can compute the key without taking a discrete logarithm, which, with current algorithms, is not computationally feasible if the numbers chosen are large enough.

Although Diffie and Hellman's meth-od worked only for key exchange, not for the actual encryption of messages, they discussed

the possibility that similar ideas might provide a secure public key encryption and authentication scheme. A public key cryptosystem, they said, should consist of a public encryption scheme E and a private decryption scheme D, where D and E are easy to compute, and such that for a message M, $D(E(M)) = M = E(D(M))$. It should also be the case that revealing E publicly does not give outsiders an easy way to compute D.

Such a scheme would be useful not only for encrypting messages but also for authenticating them. If D is applied to a message, Diffie and Hellman explained, anyone receiving the message can verify that it came from the holder of the private key by applying E to it. The key (so to speak) to finding such a scheme, they said, is something they called a *trap-door one-way function*, a function that is easy to compute but hard to invert unless you know the secret—the trap door. They left it open for their colleagues at MIT to find an example of such a function.

"Diffie and Hellman prompted a lot of people to ask themselves what they could do in this public key framework," says Dan Bernstein, a cryptographer at the University of Illinois, Chicago.

Congratulations, Ron!

One day in November, shortly after the Diffie–Hellman paper appeared, Rivest was reading it in his MIT office when Adleman dropped by. Rivest launched into an enthusiastic description of the results, but Adleman was not at all impressed: "It sounded like they'd solved some sort of engineering problem," he says.

"What resonated with RSA is that it didn't look like anything other than factoring would break it."

Adleman and Shamir had joined the MIT mathematics department earlier that year, and Rivest had become a member of the computer science department two years earlier. Although Rivest and Shamir were interested in cryptography and Adleman was primarily a number theorist, the three had a common interest in the budding field of computational complexity. Because MIT was (and still is) divided into interdepartmental laboratories, the three had offices near each other.

"We were both friends and colleagues and we were always walking into each other's offices, hanging out, and going on ski trips," Adleman remembers. "It was a very collegial atmosphere."

The Diffie–Hellman paper might have failed to captivate Adleman, but Rivest found an enthusiastic collaborator in Shamir. The two cryptographers started tossing around ideas for a system that would satisfy the Diffie–Hellman specifications. They quickly focused on number theoretic problems, Rivest says, following the ideas from the Diffie–Hellman paper, and with this they managed to capture Adleman's attention.

The trio soon settled into the dynamic that would be so productive. When Rivest or Shamir would come up with a new number theoretic scheme, Adleman, usually after only a few minutes' thought, would poke a hole in it.

Most of the proposals had flaws that were fairly obvious to a number theorist, Adleman recalls, although he singles out one, based on low-dimensional lattice reduction, that kept him up for a night.

The fun continued for several months; Shamir and Rivest would toy with the problem over dinners at local restaurants, and on group vacations to Vermont ski resorts. Public key cryptosystems were even the topic of conversation at a student's Passover seder in 1977.

Around midnight the evening of the seder, Rivest called Adleman with a new idea. It was immediately clear to Adleman that the idea was a good one. "My response was 'Congratulations, Ron, that should work,' " Adleman says. So far, it has.

"What resonated with RSA is that it didn't look like anything other than factoring would break it," Adleman says.

Rivest wrote up the paper and sent a copy to Adleman. The authors were listed in the usual alphabetical order: Adleman, Rivest, Shamir. Adleman immediately objected that he hadn't done enough to be included as an author, but Rivest disagreed. Remembering the sleepless night he spent thinking about the lattice-reduction proposal, Adleman reconsidered. "I thought that was genuine work—the rest was all observation," he recalls.

Adleman agreed to be listed on the paper but insisted that his name be last, out of alphabetical order, to reflect what he saw as a minimal contribution. "I remember thinking that this is probably the least interesting paper I will ever write and no one will read it and it will appear in some obscure journal," he says.

Adleman was wrong about two things: the interest the RSA paper would generate, and the nature of the journal that would publish it. RSA was first presented to the public by Martin Gardner, in an article in *Scientific American* in August 1977. The paper written by Rivest, Shamir, and Adleman appeared in *Communications of the ACM* later that year.

A slew of other proposals for public key cryptosystems followed publication of the RSA protocol. One of them, strangely enough, was based on the lattice-reduction idea that Adleman had grappled with. Naturally, Adleman shot it down.

How RSA Works

In the RSA public key cryptosystem, the recipient of a message has a public encryption key, e, made available to everyone, and a private key, d, that he keeps secret.

Suppose Alice wants to send Bob a secret message m. In RSA, Bob creates his public key e by choosing two large primes, P and Q, at random and multiplying them together to get N. T o find large primes, he chooses the numbers at random and, using one of several fast probabilistic methods, tests their primality. He then picks a number e that is relatively prime to $\Phi(N) = (P - 1)(Q - 1)$. Bob's private key d is the multiplicative inverse of e mod $\Phi(N)$.

The encryption–decryption protocols, then, are as follows. Bob publishes his public key e together with N. Alice encrypts the message m by raising it to the eth power mod N and sends it to Bob. To decrypt the message, Bob takes

m^e mod N and raises it to the dth power mod N. By a theorem of Euler, for any a relatively prime to N, $a^{\Phi(N)} \cong 1 \pmod{N}$. Thus, $m^{ed} \pmod{N} = m^{1(\text{mod } \Phi(N))} = m$.

The security of RSA rests on the assumption that it's infeasible to compute d, given only e and N when N is large enough. The only known way to break RSA, in general, is to compute P and Q by factoring N, and there are no efficient (polynomial-time) factoring algorithms.

Even if an implementation holds up under mathematical attacks, a wily eavesdropper can get information about RSA keys by other means.

Since the RSA encryption and decryp-tion algorithms are slow compared with *symmetric key* cryptosystems, in which the encryption and decryption keys are the same, RSA is not typically used to encrypt entire messages. Rather, a symmetric key algorithm, such as AES (the Advanced Encryption Standard), is used to encrypt the message; RSA is then used to transfer the key to the symmetric cipher.

The Complexity of RSA: What Would It Take to Break It?

In their paper, Rivest, Shamir, and Adleman give proofs that finding the decryption exponent d or $\Phi(N)$ is at least as hard as factoring N. What they were not able to show is that factoring N is the easiest way to break their encryption scheme.

"It may be possible to prove that any general method of breaking our scheme yields an efficient factoring algorithm," they wrote. "We have not been able to prove this conjecture, however."

Surprisingly, in the decades since RSA was introduced, no one else has been able to prove it, either. Nor has anyone been able to prove that factoring is easy—the efforts of scores of researchers over 30 years have produced merely a series of hints and partial results.

To be sure, the size of the numbers that can be factored has increased by leaps and bounds, due to algorithmic advances like the number field sieve and the inexorable advance of computer hardware technology. Even so, if N is chosen large enough (1024 bits is the current standard), breaking RSA via factoring remains out of reach.

The Diffie–Hellman paper came on the heels of work that launched the field of computational complexity theory, the study of the computational hardness of problems. In 1971, Stephen Cook, now a professor of computer science at the University of Toronto, defined a large class of problems that he called NP (for non-deterministic, polynomial-time); intuitively speaking, these are problems for which the correctness of a candidate answer can be quickly checked, but for which there are often exponentially many instances to check. NP contains P, the class of problems for which there are "fast," polynomial-time algorithms.

Cook's definition emerged from his proof that "satisfiability," a well-known computational problem, is at least as hard as any problem in NP. Specifically, he showed that any fast algorithm for the satisfiability problem can be converted into a fast algorithm for any problem in NP.

Richard Karp of the University of California at Berkeley then demonstrated that a number of other well-known problems, which he named NP-complete problems, also share this property.

Factoring, however, was and is not among the problems believed to be NP-complete. Until 1994, when Peter Shor of AT&T Labs devised a polynomial-time algorithm for factoring on a quantum computer (a machine that still doesn't exist at a practical scale), there was no known efficient algorithm for factoring. A year later, a group of computer scientists showed that solution of NP-complete problems on a quantum computer is unlikely. This and other results in complexity theory suggest that factoring is easier than the satisfiability problem. Breaking RSA by factoring could thus be feasible, even if P $\neq$ NP.

There are also indications that cracking RSA may be easier than factoring. In 1998, Dan Boneh, with Ramarathnam Venkatesan of Microsoft Research, showed that it's not possible, using only algebraic steps, to convert an efficient algorithm for computing eth roots mod N into an efficient algorithm for factoring N.

Since RSA is not necessarily as hard as factoring, which seems likely to be easier than the NP-complete problems, it seems as if it would be safer to build a cryptosystem on a problem that, like satisfiability, is known to be as hard as any problem in NP. But applying the theory of NP-completeness to cryptography is tricky.

Diffie and Hellman referred, in their paper, to the classes P and NP and noted that ideally, a public key cryptosystem should be based on a problem known to be NP-complete. But they cautioned that conventional complexity the-ory categorizes problems by their worst-case hardness, whereas for a public key cryptosystem, the underlying problem has to be hard in the typical case. Thus, a cryptosystem based on a typical NP-complete problem could have the very undesirable property that 90% of the keys are easy to break. It's not necessarily the case, then, that NP-complete problems yield secure cryptosystems.

Still, Diffie and Hellman pointed out that the knapsack problem (given a set S of n integers and an integer y, find a subset of S that sums to y), which Karp showed to be NP-complete, appears to be hard in the average case and suggested it as a candidate for a public key cryptosystem. Indeed, after RSA was published, several researchers came up with cryptosystems based on the knapsack problem. Unfortunately, none of them held up: The number theoretic glue used to convert the results into cryptosystems was too weak.

Complexity theorists have demonstrated that the existence of trap-door one-way functions is a stronger assumption than P $\neq$ NP, which means that the notion of secure public key cryptography could turn out to be nothing more than wishful thinking.

And yet provable security is not always a good thing. Michael Rabin of Harvard University designed a crypto-system that is provably equivalent to factoring. The system isn't practical, however, in that cracking a single encrypted message allows an eavesdropper to factor N, and thus to read all other messages encrypted with the same key.

Theory Versus Practice

As complexity theorists have pondered the hardness of RSA and factoring, cryptographers have been working on another aspect of RSA security: how it can be safely used in practice.

Boneh of Stanford calls RSA, as defined by its founders, "textbook RSA." "Textbook RSA isn't secure in practice," he points out, citing the following attack:

Suppose Bob has a public key (e, N), $N = PQ$, where N is 1024 bits, and imagine that Alice wants to send him a 64-bit message M. Alice computes $M^e \bmod N$ and sends it. Alice's message happens to be the product of two integers, M_1 and M_2, each less than 34 bits long (which is true about 20% of the time). A clever eavesdropper, let's call her Eve, can then guess M by trying far fewer than all 2^{64} possibilities.

Clever Eve, realizing that

$$C = M^e = M_1^e\, M_2^e \pmod N,$$

$$\frac{C}{M_1^e} = M_2^e \pmod N,$$

makes a table of all possible

$$\frac{C}{M_1^e}$$

for all possible M_1 between 1 and 2^{34}. She does this in time 2^{34}. She then computes all possible values of M_2^e (in time 2^{34}) and, by sorting, checks to see if any of those values are in her table (sorting time is proportional to $34 \cdot 2^{34}$). The total time is still less than 2^{40}, which is far less than 2^{64}, the time needed to test all possibilities for the message M.

To protect against such attacks, implementations of RSA do some preprocessing of the message before applying the "textbook" algorithm. One common method, called PKCS1, applied RSA to a block of bits consisting of the digits 02 (to identify it as an RSA-encrypted string), followed by a random pad, and then the message. By elongating the message, PKCS1 thwarted the above attack. But in 1998, Daniel Bleichen-bacher of Bell Labs found a weakness in this method as well.

The problem was the 02 identifier that appeared at the beginning of each RSA string. When the server decrypted the string and didn't see an 02, it would send back an error message, giving the sender information about the decrypted string. Bleichenbacher was able to show that by repeatedly querying the server (several million times) with random integer multiples of the string, it is possible to decrypt the entire RSA message.

This vulnerability could easily be fixed by limiting the number of such queries to the server. But what about similar attacks? Fortunately, in the early 1990s, Mihir Bellare of the University of California at San Diego and Phil Rogaway of the University of California at Davis had published an efficient method, known as OAEP, for preprocessing RSA messages; they showed that, under some assumptions, the method resists a large class of attacks, including that of Bleichen-bacher. (Another method, by theoretical computer scientists Danny Dolev of Hebrew University, Cynthia Dwork of Microsoft Research, and Moni Naor of the Weizmann Institute, is provably secure with no assumptions, although it is inefficient.) Following the 1998 attack, OAEP became the standard in practice.

Researchers have also found that the size of the encryption and decryption keys can affect RSA's security. RSA runs faster when the keys, particularly the decryption key, are small. But a series of results gives bounds for d below which RSA is insecure. Most recently, Boneh and Glenn Durfee of the Palo Alto Research Center showed that it's possible to break RSA if $d < N^{.292}$; researchers conjecture that any $d < N^{.5}$ is unsafe.

Even if an implementation holds up under mathematical attacks, a wily eavesdropper can get information about RSA keys by other means. In a series of results, Paul Kocher, who owns a San Francisco-based computer security firm, has shown that RSA can be broken by timing the decryption of a message or by measuring the power used by the device decrypting the message. Recent implementations of RSA protect against such attacks as well.

Afterword

RSA's success bears two important lessons for readers: If Ron Rivest comes to you with an engineering problem, listen, and if he wants to put your name first on the resulting paper, let him. "ARS sounds better and better to me now," Adleman quips.

Sara Robinson is a freelance writer based in Pasadena, California.