After he had showered and shared a meal with the others, Danny Tenacce was about to return to the room to check his work when he spied Salito heading toward him.

“It’s my shift. You’re stuck with me again.”

Danny entered the room and motioned her in.

“What’s wrong?” she said in a low voice.

“Nothing. I just want to know what you think about Father McCleary?”

“Well, he’s big, strong, smart, incredibly even-tempered. A little too perfect, if ya ask me.”

“Exactly! Do you think he’s really a priest?”

“Suspicious, eh? You’d make a good cop. I was wonderin’ that myself.”

“Well?”

“I’ve had a couple o’ thoughts. I always consider the worst scenario first.”

“What’s that?”

“That he and DeAngelo are Knights posing as Brothers, waitin’ for us to find and decipher the key so they can steal it and decipher the book. Or that they’re both Brothers and The Brothers really are the bad guys. I talked ta Joe about it.”

“What did he say?”

“I said maybe DeAngelo is the one that tipped the Knights as to the location of the safe house. He said that he never told DeAngelo where the house was. I said maybe he put the Hulk on your tail. After all, there’s no question the man’s got skills.”

Danny laughed and nodded. “And what did he say to that?”

“He said, no way. That he and DeAngelo went way back and if DeAngelo was a rat, then his life was over anyway. I thought, strong words from a strong man with good instincts. I believe him.”

“He convinced me similarly. That’s why I didn’t balk when he said he thought the book was safe with Father Frank. But the Hulk, as you aptly called him, could still be a double agent working for The Knights. Or some other entity.”

“Joe considered that. His argument against it was that DeAngelo had learned ta keep his guard up from his military experience, and from his years as a priest, had acquired a deep understanding of human character. He didn’t think he could be duped.”

“And if he was?”

“Then when McCleary tries to take the book, Joe plans ta take ‘im out.”

“That might be easier said than done.”

“I agree, but you know Joe. He said another thing that made me think he’s right, though.

“What’s that?”

“The Knights don’t seem too interested in keepin’ us around ta decipher the book. I mean, they were tryin’ ta have us arrested, and before that, at Mary’s Bookstore, they frankly tried ta kill us. More likely they just wanna destroy it. Either, contrary to legend, they know what’s in the book and don’t want its contents known or they don’t wanna take any chances that the book says what they don’t want it ta say. Either way, if McCleary was a Knight, he woulda made a play for the book on Staten Island. And if Father DeAngelo was a Knight. Well …”

“He’s got a point.”

“I’ve got another idea.”

“Which is?”

“That McCleary works for the United States government. DeAngelo said that he was with the Delta Force. Maybe he still works for the Delta Force, or the CIA, or the government in some capacity, and is just posin’ as a priest. Think what would happen if, God forbid, the book reveals that Jesus was just part of a scheme ta deceive the public into thinkin’ he was divine; deceive ‘em so they would believe in an afterlife where there was reward for good behavior and punishment for evil; that the whole thing was a ruse ta keep people’s behavior in check. If the ruse were exposed, social chaos would ensue.”

“Anarchy.”

“Huh?”

“Anarchy would be a more appropriate term. Chaos has a very specific meaning in science and mathematics. It refers to a deterministic system in which the state of a system at some time is sensitive to the system’s initial conditions; to quote Lorenz, ‘a system in which the present determines the future, but the approximate present does not approximately determine the future.’”

“Genesis says it’s the waste and void before God said ‘Let there be light.’ I mean it as defined in common usage: disorder. Does the word really matter?”

“It might.”

“Ok. Anarchy then. No way the government wants ta deal with that. I think McCleary may be here ta snatch the book if it says what I hope and pray it doesn’t.”

“You think there’ll be anarchy if the truth is exposed? I disagree. I think people are rational beings who would recognize that behaving in a civilized manner is in their best interest, whether or not there’s eternal bliss or damnation hanging over their heads.”

“With all due respect, Doctor, I contend that your judgement is impaired by your exposure ta only a limited segment of society. As cops, your father and I see the whole spectrum of humanity, especially the unsavory element with which you have little experience, and I assure you, if Christianity was shown ta be a hoax, lawless pandemonium would result.”

“I must admit, I find myself wondering—after this debate and our previous discussion about consciousness—whether the woman who passes herself off as a jockette/cheerleader from Brooklyn isn’t really a professor from Harvard. No Standuvard—that school’s a little more conservative. At any rate, I think we’ll have to settle both of these issues later. I hear the rest of them coming.”

“Yes we will,” Salito replied.

The door opened. Tenacce made his customary intrusive entrance with McCleary in tow. “Did I interrupt somethin’?”

“No, we were just talking.”

“About you, actually.”

“Oh,” said Tenacce, his detective’s suspicion aroused. “Any progress?”

“Not a bit but I haven’t checked the computer in a while.”

Danny Tenacce sprang hopefully from his chair and rattled the mouse to bring the digital screen to life. When he saw the numbers being added at warp speed to the bottom of the window, he slammed his hand down on the desk.

“I understand your frustration,” said McCleary.

“How could you?”

“Because I’ve worked on this problem myself. For seven years. With the same results.”

“You’ve worked on this problem?”

“Why are you surprised?”

“Because we just found the key to the key.”

“It’s 2048 bits long, if that will help.”

“I know that but how do you know that?”

“Word of mouth.”

“Ah, yes. The Brotherhood. So then you know that solution of this problem is tantamount to proving P = NP.”

“It would have far-reaching consequences, to be sure.”

“I was considering the analogy more as an illustration of the problem’s difficulty but you’re right.”

“You’ve already developed an algorithm with far-reaching consequences in your decryption of the last papyrus. What do you plan to do with it?”

“I’m uncertain.”

Tenacce scowled. “Sounds interesting. Maybe you two could translate inta English.”

Danny was surprised that his father had been listening. “We’re talking about the difficulty of decoding the key.”

He remained at the desk, a place at which he had been for the better part of the last three days, save the few hours when his brain and lagging lids would not allow him to be there—sat there, regathering his pencil and pad, about to readdress the problem that had befuddled him—when he noticed Salito and his father staring.

“And,” Salito said finally.

“And what?”

“Why is decoding the key so difficult?”

“It’s a complicated issue,” said Danny, doing his best to deflect the inquiry.

“Like everything else in this case,” Salito responded.

“Let’s have it,” demanded Tenacce.

“It’s gonna take some time.” Danny looked at his father.

“We got plenty o’ that,” his father said.

Danny rose from his seat and moved a whiteboard to the center of the room before his audience. He uncapped a marker and began.

“In computer science, the complexity of solving a problem is classified into categories. This complexity depends on the time it takes to solve the problem, which, in turn, depends on the number of steps it takes to accomplish the task. P, one of those categories, stands for polynomial time. This refers to the fact that the mathematical expression for the number of steps needed to solve the problem is a polynomial. It means that, in the worst-case scenario, given you have n things to test, it would take no more than nk steps to solve the problem—solve the problem, that is, on what’s called a deterministic Turing machine, a theoretical machine like a computer that can execute only one task at a time. Let me explain.

“Say you want to compare each of 1000 cards with each other to see which card has the highest number printed on it. Then n = 1000. You would compare each card to the other 999. Each comparison is a step. That’s (n-1) = 999 steps per card, times n = 1000 cards; (n - 1)n = n2 - n = 10002 - 1000 = 1,000,000 - 1000 = 999,000 steps. A computer can do that kind of calculation in less than a second. Notice that as n becomes large, the n2 term gives you most of the value of the expression; n is much smaller. In the case of the cards, the n2 term gives you 1,000,000 and the n term only contributes minus 1000, small compared to a million, so you ignore the n term. That’s why they say the expression for the number of steps in polynomial time is nk. In the above example, n = 1000 and k = 2. The characteristic of P-type problems is that they are easy for a computer to solve and also easy to verify. In the problem of the cards, I showed that you can solve the problem in polynomial time. You can also verify, in polynomial time, whether or not the card you picked as having the highest number actually has the highest number.

“Next consider the problem of trying to figure out what numbers you have to multiply together to get another number. First, let me give you a couple of definitions. The numbers that we’re going to be dealing with in this problem are natural numbers: positive whole numbers like you’re used to dealing with: 1, 2, 3, 4 …. Not fractions, decimals, square roots, and so forth. Second, the numbers you multiply together to get another number are called factors. A prime number is one in which the only factors you can multiply together to get it are 1 and the number itself. A composite number is a number that has factors other than 1 and itself. So 29 is a prime number because the only numbers you can multiply to get it are 1 and 29. On the other hand, 6 is a composite number because you can multiple 2 x 3 to get 6, as well as 1 x 6. Euclid proved about 2300 years ago that every composite number can be expressed as multiple prime numbers, multiplied together. The string of prime factors multiplied together to make an integer are unique. For example, the number 12 has prime factors 2 x 2 x 3. No other number has this combination of prime factors.

“Now for the specific problem I want to examine. I want to talk about multiplying two prime numbers together to get a composite number. Let's say that I give you the number 203. To determine its prime factors, you’d start with 2 and divide it into 203. It doesn’t go evenly so you go to the next prime integer 3. It doesn’t divide 203 evenly either. Neither does the next prime integer, 5. How about 7. 7 into 203 goes 29 times. When you get two prime factors that, when multiplied together, give you the number, you’re done. So the prime factorization of 203 is 7 times 29.

“That’s easy. Now suppose I ask you to factor 8,162,821.”

“Good luck,” blurted Tenacce. “Ya’d have ta start at 3. Ya don’t need ta bother with 2 ‘cause ya know it won’t divide an odd number without somethin’ left over. Ya’d take 3 and divide that and every odd number inta your number—correction, every odd prime number …”

“All prime numbers are odd other than 2,” Danny pointed out.

Tenacce thought about it for a few seconds. “Right, every prime number until ya find somethin’ that goes in even. If you’re lucky, ya find the answer before ya get ta 8,162,820.”

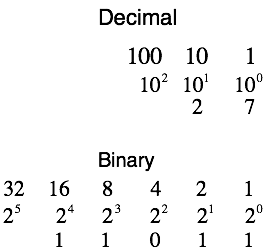
“That’s about right. Actually, you can shorten the process a little more. Say you’re going to factor 49. You start with 2 and you try to divide it into 49. It doesn’t work. Same with 3, and 5. You don’t need to try 4 or 6 because they’re not prime. 7 x 7 works. That’s all you need to do. Going any further is redundant. In the worst case, the furthest you’ll ever have to go is to an integer that, when multiplied by itself, equals the number. Such a number is called a square root. If you want to find prime factors for 119, the furthest you would possibly need to go would be to 10 because the square root of 119 is 10.9. Although, in this case, you wouldn’t even need to go that far because 7 x 17 = 119. The square root of 8,162,821 is 2,857 plus a decimal. So you might need to try dividing up to 2,857 numbers into 8,162,821 to figure out if it has prime factors. Of course, the only numbers you would need to divide into 8,162,821 would be prime numbers, so if you knew which numbers less than 2,857 are prime, you would only have to try those numbers and the number of steps to solve the problem would be considerably less than 2,857.

“The prime factors that happen to give 8,162,821 are 3011 and 2711. It would take a lot more time to do the 2711 steps to figure this out than to *verify* that 3011 and 2711 are the two prime factors that are the correct ones. All you would have to do to verify that 3011 and 2711 are the answer is multiply them together. That wouldn’t take much time at all. A computer can still handle checking 2711 numbers easily but what if the number you’re trying to factor is 617 digits long, like the 2048 bit key that we—er, I—have got to decode to find out where the key to the book is?

“What’s this bit thing that ya keep talking’ about?”

“A bit is the basic information unit with which a computer works. A computer is basically a long string of logic gates hooked together. Each gate can give an answer of yes or no, represented numerically by a 1 or a 0, respectively. Numbers are also represented by 0’s or 1’s. They’re called binary or base 2 numbers. The usual numbers we’re used to are decimal or base ten numbers. Basically, that means that each digit in the number can be one of ten possibilities, 0-9. With binary numbers, there are only 2 possibilities for each number: 0 or 1. You can represent the same number in either binary or decimal form.”

Danny Tenacce erased what he had and scrawled a new diagram on the whiteboard:



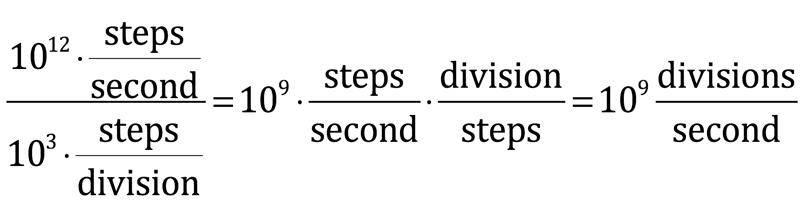
“Take the decimal number 27. You remember exponents. That’s the number of times you have to multiply the base number (in this case, 10) together. So, in the diagram, , ,  (any number to the zero power is 1). 27 in decimal form is 2 groups of tens and 7 groups of ones so you put a 2 in the tens column and 7 in the ones column. There are no hundreds or higher digits so you leave them blank.

“You separate binary numbers into columns in a similar fashion. . To express the decimal number 27 in binary form, start by taking the column with the largest value that can be divided into 27, in this case, 16. Divide 16 into 27. It goes once so you have 1 group of 16 in 27. Therefore, put a 1 in the sixteens column. When you divide 16 into 27, you get a remainder of 11. Go to the next column, 8. 8 goes into 11 once so put a 1 in the eights column. You’re left with 3. Go to the next column, 4. 4 doesn’t go into 3 so put a 0 in the fours column. You still have 3 left over. Go to the twos column. 2 goes into 3 once so put a 1 in the twos column. You’ve got 1 left over. Go to the ones column. 1 goes into 1 once. Put a 1 into the ones column. You’ve got no more numbers left so you’re done. So 27 in binary form is 11011. Each column is a bit so 27 is a 5-bit number. It’s stored in a computer with five little logical units. Notice that it takes more digits to express a given number in binary form than in decimal form: two digits in decimal form and five digits in binary form in the case of 27. A 2048-bit key like the one that guards directions to the key is a 2048-digit binary number. The decimal equivalent is 617 digits long.

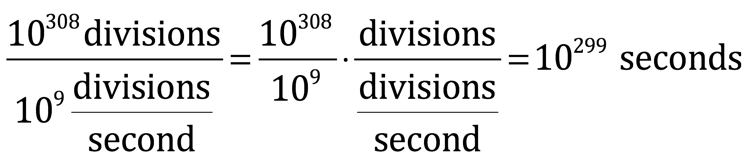
“To factor a number this big (call it n), as I described previously, you might need to carry out as many as the square root of n calculations. The square root sign is . Taking the square root of a number is the same as the number raised to half its exponent. So

 . For simplicity, let’s make it .

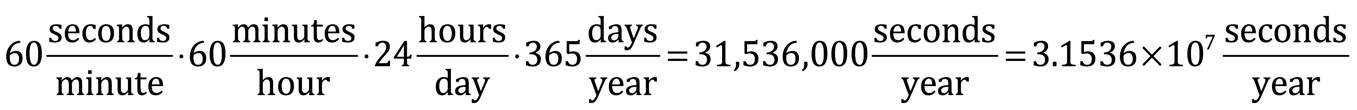
So you might need to carry out as many as  calculations, i.e. divide as many as  numbers into n to find the answer. Each of these divisions probably takes—I don’t know—maybe 1000 (i.e. ) computer steps or FLOPs (floating point operations) as they’re called. That’s probably a conservative estimate, but as you’ll see, it really won’t matter. A decent computer can carry out about a teraflop each second. That’s  FLOPs, or computer steps, per second. So a decent computer will be able to carry out  divisions per second. I got that by doing the following:



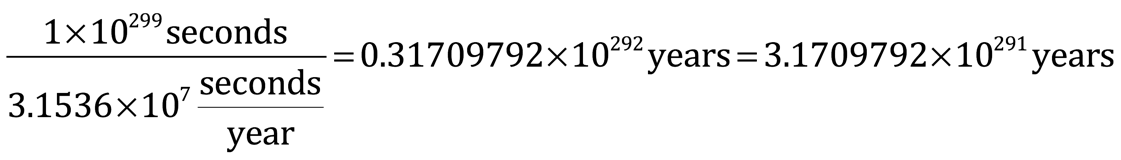
To figure out how many seconds it’ll take to carry out  divisions, you divide that by the number of divisions per second your computer can carry out:



There are 60 seconds per minute, 60 minutes per hour, 24 hours per day and 365 days per year, so there are 31,536,000 seconds per year:



To figure out how long it would take—in years—to carry out all  division steps necessary to find the secret key, you divide the length of time to do this—in seconds—by the number of seconds in a year:



Even if you had 100 computers, it would still take  years. The universe is about 13.7 billion years old. A billion is .

“I think you’re starting to get the picture. Factoring large numbers with two large prime factors is an extremely hard problem to solve but if you have the two factors, multiplying them together on a computer to verify that you have the correct answer is easy.

“A problem that is hard to solve but easy to verify is called an NP problem. NP stands for nondeterministic polynomial. A nondeterministic Turing machine is a theoretical machine that can carry out multiple possible actions simultaneously. A deterministic Turing machine, as I described a little while ago, can execute only one action at a time and takes a long time to solve NP problems. The time that a deterministic Turing machine would require to solve the prime factorization problem I described is called exponential time because it can be expressed mathematically by the general formula kn. For factoring prime numbers, k would be the base of the number to be factored and n is the square root of the number; in our example  steps. As you can see, that number gets very big, very fast as n gets big. On the other hand, the finding of the card with the highest number in a deck of 1000 that I described before could be solved in polynomial time. The number of steps to the solve the problem is given by a polynomial expression, specifically, . Obviously,  is a lot less than .

“So it would take a deterministic Turing machine exponential time to solve an NP problem like finding prime factors of a large number. In contrast, a nondeterministic Turing machine could carry out the same NP problem in much less time; specifically, in polynomial time. Thus the name. NP - nondeterministic polynomial.

“As I’ve said, modern computers work like deterministic Turing machines. No NP problem has ever been solved by modern computers in polynomial time. Computer scientists aren’t sure if that’s because it’s impossible or because no one has been smart enough to do it yet. If it is possible, that would mean that P equals NP. If it’s impossible, then it would mean that P does not equal NP. The Clay Mathematics Institute in Oxford, England has seven unsolved problems for which it will pay anyone who solves them $1,000,000 per problem. A formal proof that P equals NP, or that P does not equal NP, is among those problems. It would be important to know since, if P equals NP, programmers will work harder to solve NP problems, many of which have important practical implications. If it can be proven that P does not equal NP, then computer scientists will stop wasting their time working on NP problems.

“Decrypting the key would essentially provide an example that P=NP which would be tantamount to proving P = NP, a task that many have tried but at which no one has yet succeeded. That gives you some idea what we’re up against.”

“We have reason to be optimistic, though” said McCleary.

“Why is that?”

“At least it’s an RSA key you’re dealing with. If it were a 2048-bit symmetric key … or even 256.”

Salito shifted uncomfortably in her seat. “Then the far reachin’ consequences that you mentioned are that, usin’ this algorithm, current encryption methods could be cracked. Computer security would be out the window.”

“Those would be the consequences,” confirmed McCleary. “I can assure you, there are many people who would love to gain possession of the factoring algorithm Dr. Tenacce has already designed. I wouldn’t be surprised if a couple of agents from the US government pay you a visit when the smoke clears.”

“If the smoke clears,” Danny Tenacce said. “And I’m sure you’ll be the one sending them.”

“Better than the Russian mob. So what are you going to do with it?”

“I haven’t decided yet.”

“What does factoring a number have ta do with encryption anyway?” asked Salito.

“Everything,” Danny replied. “At least where RSA encryption is concerned.”

“Can you elaborate.”

“Do you want the long version or the short version?”

“We’ll take the short version,” Tenacce volunteered quickly.

“Are you sure? Even the short version is going to take some math.”

“We endured P versus NP, didn’t we?”

Danny turned his eyes to Salito.

“Go for it,” she said.

“Ok, then. Say Mary wants to send a confidential message to Danny and keep it from John, who’s always trying to intercept it. Danny sends an open padlock to Mary, a padlock a copy of which John or anyone else who wants one can have. Mary puts her message in a box and locks it with the padlock. The nature of the box and lock are such that no one can cut off the lock, cut a hole in the box or otherwise access the message except by using the key to open it, a key that only Danny has. Mary sends the locked message back to Danny, in plain view of John, who, without the key, can’t open it. When Danny receives the box, with the key, he easily opens the box and reads the message. This is an example of a one-way function. The padlock is easy to lock but hard to unlock. Unless, of course, you have the key.

“In RSA encryption, the functions of the physical lock and key are accomplished with mathematics. The padlock function is performed by a thing called the public key while the function of the key that Danny uses to unlock the padlock is referred to as the private key.

“Our situation is a little different in that Mary made up Danny’s public key, encrypted the message with the public key and sent the encrypted message to Danny. Now all Danny has to do is use the secret key to unlock the message. The problem is, Danny doesn’t have the secret key. He has to find it. By factoring a 2048-bit number.”

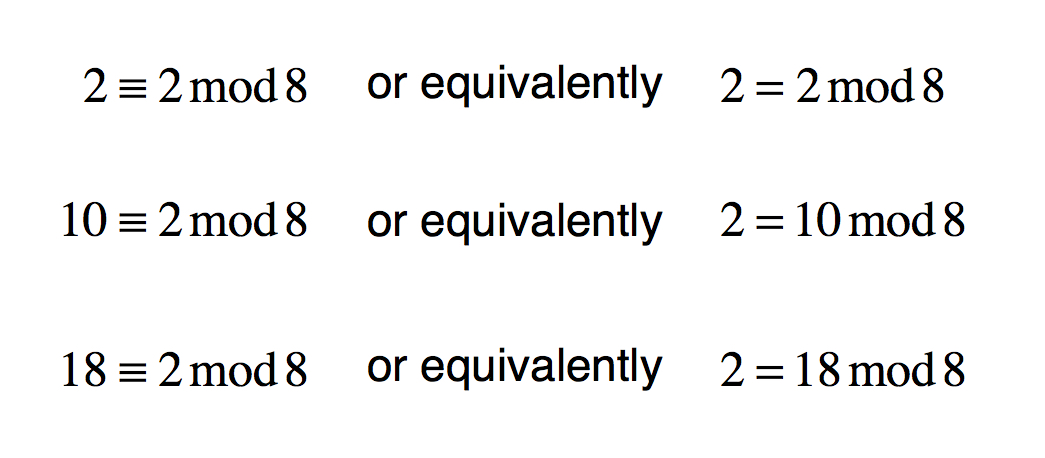
“Kinda like an NP problem,” said Tenacce.

“Exactly like an NP problem,” chimed McCleary.

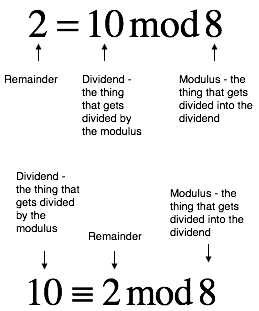
“That’s the basic concept. To practically implement this strategy, computers use a one-way mathematical function called a trap door function—the mathematical analogue of the padlock—easy to encrypt but difficult to decrypt, unless you have the key. This function makes use of modular arithmetic, a technique to which you’ve already been exposed. Peterson used it to construct his message in blood. It’s crucial to the process so I need to refresh your memories.

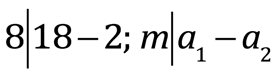
“In principle, it’s simple. It’s just clock arithmetic. Let me give you an example. Take the equation . That means if you divide 20 by 12 you get a remainder of 8. An alternative way to think about it is to picture a clock like the ones you hang on the wall. It has 12 numbers on its face. Ignore the minute markers. Each number signifying an hour is a tick. If you start at 12 and go 20 ticks around the clock you end up on 8. The number of ticks on the clock is called the modulus. In this case, it’s 12 but it could be 3 or 8 or 4892. Anything. Let’s take a clock with 8 numbers—or ticks—on its face. Let’s use this to figure out the meaning of this equation: . The clock in this case has 8 numbers—or ticks—on its face. To solve this equation, you’d start at the top, at 8 (which is the same as 0, by the way), go 10 ticks around the clock and end up at 2, which is the answer. Alternatively, we could take 10 and divide it by the modulus 8. We’d end up with a remainder of 2, which is, again, the correct answer.

“One other thing we need to address regarding modular arithmetic is the concept of congruence. The symbol for congruence in modular arithmetic is ‘.’ This refers to a group of entities that are equivalent. For example, 2, 10 and 18 are all congruent to . That means use a clock with 8 hour marks, or ticks. Start at the top of the clock, at 8 (or 0, to which it is equivalent). If you go around the clock 2 ticks, 10 ticks or 18 ticks, in each case, you wind up in the same place: at 2. The mathematical expressions for these relationships would be:



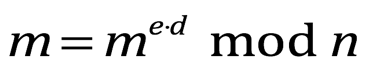
“Another way to think about this is as follows: divide 8 into 2 and you get 0 with a remainder of 2; divide 8 into 10 and you get 1 with a remainder of 2; divide 8 into 18 and you get 2 with a remainder of 2. The remainder is the thing of interest here. If you divide the modulus into the number on either side of the equation, you wind up with the same remainder. From this, we have two ways of saying the same thing:



“Also note that if we subtract the two numbers other than the modulus in the above equations, the modulus will divide that difference evenly. The answer will be an integer and there will be no remainder. (10 - 2)/8 = 1, (2 -10)/8 = -1, (18 - 2)/8 = 2, and so on. This idea is expressed in equations as , etc.

“So that’s a crash course in modular arithmetic. Now on to the specific equation on which modern cryptography, the so-called RSA algorithm, is based.”

Danny erased the contents of the whiteboard and replaced it with a single line of letters and numbers:



“How’d ya come up with that?” Tenacce groaned.

“I took a cryptanalysis course and slogged through a bunch of convoluted mathematical proofs.”

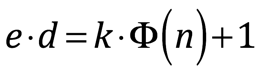
Tenacce and Salito frowned in protest.

“You’re the ones that asked for the short version.”

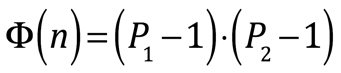
“Yeah, yeah. So tell us what it means.”

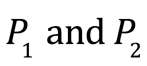
“It means this:

 is the message encoded in numeric form

 where

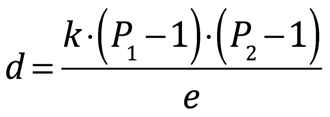
 is the Euler phi function which is a count of the numbers that share no common factors with *n*

 Where

are two prime numbers, that, when multiplied together, equal *n*

*n* is the large number we need to factor

Rearranging and substituting, that means that



“We picked  and . That’s how we came up with *n*. We also chose *k*. So we have all the information needed to determine *d*. *d*, it turns out, is the secret key needed to decode a message sent to us. Let me show you how it all works.

“If we were using computers, in the modern era, and Danny wanted to receive a message from Mary, he would send his public key, which consist of the numbers *e* and *n*, to Mary. John and anyone else who wants it has access to that key. Since computers only understands numbers, Mary’s message consists of a string of numbers. Each number stands for a character like you can type on a keyboard. There is a standardized system of translation from characters to numbers that computers use, called ASCII, which stands for American Standard Code for Information Interchange. Let’s say Mary’s message is just the letter J. The ASCII code for J is 74. 74 is the value of *m*. Mary uses the values of *m*, *e* and *n* to generate an encrypted message, let’s call it *c*, according to the following equation:



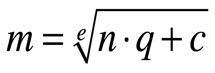
“Mary sends the encrypted message, *c*, to Danny. The actual message is a long string of characters which translates into a number with a long string of digits. John can see the encrypted message but can’t decrypt it.”

“Why can’t he decrypt it?” Salito was animated as she spoke. “John knows *c*, *e* and *n*. He oughtta be able to figure out *m*, shouldn’t he?”

“No, because we’re using modular arithmetic. Remember,



therefore,



As you said, John knows *c*, *e* and *n*—they’re just numbers that anyone can see. However, he has no idea what *m* and *q* are. What you have, then, is one equation with two unknowns. There is no unique solution to such an equation; *m* and *q*, literally, could be anything. I suppose John could try putting in values for *m* and *q* by trial and error, then check to see if the message encoded by the string of numbers that is *m*, makes sense. However, because  and  are so large, the range of values that *q* could be would astronomical. The time it would take to test values of *q* and find the correct answer would be so long as to make this method impractical, on a par with trying to factor *n* and guess the private key.

Danny, however, has the secret key, *d*, which he can use to recover the message, *m*, by using the equation



“John knows *e* and could decrypt the message if he could factor *n* since the factors of *n* are and , and and  determine *d*, but we’ve already seen how long factoring *n* could take if  and  are large.”

“So where are we in all this?” asked Salito.

“The papyrus contains a bunch of encrypted text and two numbers. I assume those numbers are *n* and *e*. Of course, that’s assuming we’re dealing with RSA encryption. There are other algorithms out there.”

“That’s a lot of assumin’. You know what that gets ya,” said Tenacce.

“I think those are pretty good assumptions,” said McCleary.

“Does tradition tell you that?” Danny Tenacce asked, with some bite.

“Yes.”

Salito regarded the equations on the whiteboard then at the one who had written them. “So it looks like we’re in about the same position as John.”

“That’s about right.”

“I’ll pray for you,” said McCleary.

“You said that already. It hasn’t helped much,” said Danny Tenacce.

“Consider the words of the poem again, then.

But Danny did not answer. Instead, he capped his marker, returned to the desk and began work on a new algorithm.