Zero-Knowledge Proofs

A new heuristic method lets you prove your identity without revealing a password or other information

Peter Wayner

RECOGNIZING THE DIFFERENCE between an authorized user and a fake is a difficult problems for computers. Traditional password systems and other heuristics for controlling access can never be made perfectly secure because a computer can judge only the signals it receives, not whether the binary bits are the product of a sincere, authorized user or of an impostor providing the same input.

Now, new mathematical techniques known as zero-knowledge proofs can strengthen these heuristic approaches by providing complex *interactive passwords* for users, passwords that cannot be faked by anyone who happens to intercept the message—or even by the host computer itself. The chief developers of the new methods are Oded Goldreich of Haifa University, Silvio Micali and Shafi Goldwasser of MIT, Manuel Blum of Berkeley, Charles Rackoff of the University of Toronto, and others (see reference 1).

Zero-knowledge proofs differ from regular mathematical proofs in two epistemologically curious ways: They hide the truth while defending its validity, and they are played out much like a card game. In a zero-knowledge-based exchange, the prover first makes an assertion. The skeptic verifies the assertion and specifies the next fact he or she would like to hear. The prover responds with another assertion. The exchange continues until the skeptic is satisfied.

What makes this ordinary-sounding interrogation process unique is that the individual assertions taken together reveal no privileged information *except* the fact that the prover isn't lying; the skeptic can safely conclude that the prover is indeed the person he or she claims to be.

A Preliminary Example

Arms-control treaties, because they are plagued by mistrust, are a good preliminary example for understanding how an interaction can hide information while providing some kind of validation.

Suppose a nation wants to prove it does not have nuclear warheads at a storage plant without revealing exactly how many conventional warheads it has stockpiled. In one zero-knowledge approach, the representative from the proving nation randomly divides the warheads between two locked rooms. The examiner from the skeptical nation flips a coin to choose one of the rooms. The prover hands the skeptic the corresponding key so he can check the contents of the selected room. If the skeptic doesn't find a nuclear warhead in the room, he can conclude there is only a one-in-two chance that the treaty is being violated.

The prover nation locks both doors, rearranges the warheads, and again lets the skeptic nation randomly select a room for viewing. After this examination, if no nuclear weapons are found, the skeptic concludes there is only a one-in-four chance that the treaty is being violated.

After 20 or 30 such iterations, the skeptic can be satisfied that even though he lacks absolute proof, the chance that he has chosen the wrong door every time is practically nil. Meanwhile, the prover can be content that the exact number of missiles hasn't been revealed.

This isn't a true zero-knowledge proof because it gives the skeptic a statistically converging estimate of the number of missiles. Nevertheless, it does illustrate several important facets of the method.

First, the techniques never prove something perfectly and incontrovertibly, but they always come as close as the two parties' patience will allow. This is a drawback for anyone who needs literal certification, but it should make no difference to practical people who realize how quickly 2^n shrinks. Second, zero-knowledge proofs keep the prover honest by letting the skeptic demand any particular fact, while hiding the entire truth from the skeptic by letting him or her choose only a fraction at most. Third, these proofs rely upon one-way functions to protect the information.

One-way functions are an important part of cryptography, enabling a person to encrypt information and place it in the open, secure in the knowledge that no inverse function can be found to decipher the information. In the warheads example, the one-way function is the random division of warheads between the two rooms. It is impossible to infer the total number of warheads from the number found in just one of the rooms.

In mathematics, one-way functions are operations that have no inverse, or at least no readily discoverable inverse. For in-

continued

Peter Wayner is a graduate student in computer science at Cornell University. He can be reached at the Department of Computer Science, Cornell University, Ithaca, NY 14853.

The security can be further strengthened by requiring that each prover be able to handle any of 1000 different x,y pairs.

stance, given a list of prime numbers, you can easily generate a product. However, the inverse operation—factoring—can be so time-consuming as to be impractical when the number is large—say, 100 or more digits. The public-key-cryptography system (see reference 2) relies on the difficulty of factoring large numbers.

Quadratic residuosity is another number-theoretic property that gives a good one-way function (see reference 3). I will use it in a program that demonstrates the operation of a zero-knowledge proof.

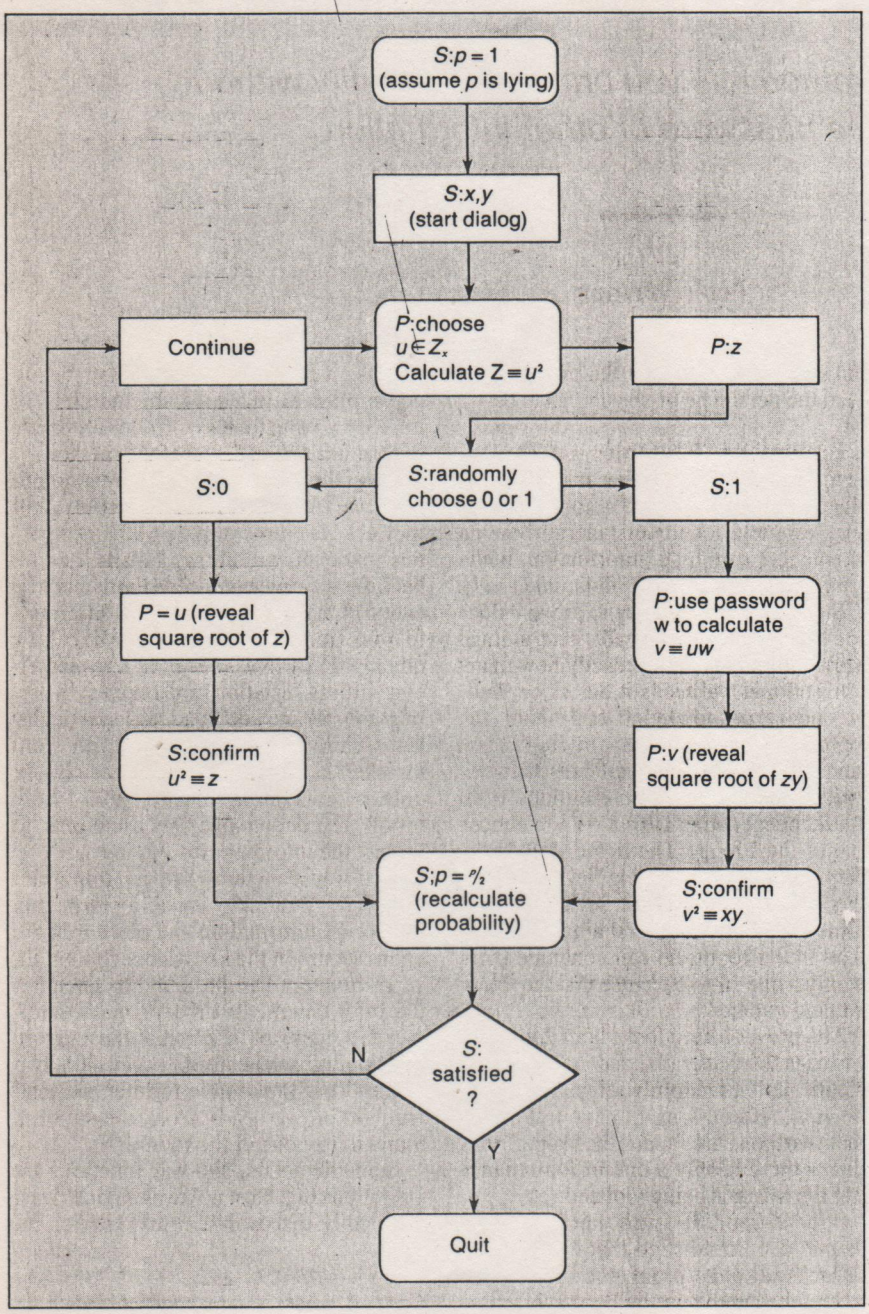

Figure 1: *Flow diagram of a zero-knowledge dialog between skeptic S and prover P. Squares indicate data revealed over the communications link; circles represent internal processing.*

First, we need some theoretical background.

Quadratic Residues

Given y relatively prime to x (i.e., x and y have no common factors except 1), y is said to be a quadratic residue of x if there exists a w such that $w^2 \bmod x = y$. For example, 9 and 10 are relatively prime, and $7^2 = 49 \bmod 10 = 9$, so 9 is a quadratic residue of 10.

For shorthand, let Z_x symbolize the set of integers relatively prime to x and QR_x symbolize the set of all elements in Z_x that are quadratic residues of x. For instance, $Z_{10} = \{1, 3, 7, 9\}$ and $QR_{10} = \{1,9\}$ since only those two numbers have square roots in Z_{10}: $1 = 1^2 \bmod 10 = 9^2 \bmod 10$, and $9 = 3^2 \bmod 10 = 7^2 \bmod 10$.

Quadratic residuosity makes a good one-way function because it is easy to square a number modulo x but difficult to find the square root of a number modulo x when the number is relatively prime to x and the factors of x are unknown.

Three other properties of quadratic residues are important here. First, the fastest way known to compute whether y is a quadratic residue of x is to start by factoring x into primes. Since this is hard when x is the product of large prime numbers (in excess of 100 digits each), a strong system must start with a very large x. The zero-knowledge interaction will also reveal nothing about the factors of x. Second, every $y \in QR_x$ has an equal number of square roots w such that $w^2 \bmod x = y$. The third property concerns products of two integers. If $y,z \in QR_x$ then $yz \in QR_x$; if $y \in QR_x$ but $z \in Z_x - QR_x$ then $yz \in Z_x - QR_x$. These facts can be proved using group theory or at least verified by working through a few sample cases.

A Working Example

In the working example, the prover is given x and y and asked to prove that y is a quadratic residue without revealing its square root. The square root is, in effect, the password, and the zero-knowledge techniques let the prover keep the password secret from the skeptic while still showing that he knows it. This prevents the password from being stolen by an eavesdropper.

Here is a more detailed view of the protocol. Keep in mind that all computations are done modulo x even when not explicitly so stated.

The skeptic S starts by giving the prover P the number pair (x,y). P will prove that he knows a square root w (i.e., $w^2 \bmod x = y$) without revealing what it is. P randomly selects u, a member of Z_x, squares it, and sends the result $z = u^2 \bmod x$ to S.

S then sends P a random bit, 0 or 1. If the bit is 0, P must reply with u, and S confirms that u is indeed a square root of the first number z. If the bit is 1, P uses the secret value w to form the product $v = uw \bmod x$, which he sends to S. S checks that this value is indeed a square root of the product zy. (Remember, $z = u^2$ and $y = w^2$ so $zy = u^2w^2 = vy$.)

In either case, P's correct response convinces S with a probability of ½ that P does know a square root of y. The process is repeated until the probability of cheating grows small enough to satisfy S. Figure 1 diagrams the process.

The technique works because it is not possible for both u^2 and u^2y to have square roots unless y is a member of QR_x. Since P doesn't know which he will be asked to provide ($\sqrt{u^2}$ or $\sqrt{u^2y}$), he cannot try to fake the choice of u. However, since P reveals only u or uw, it is impossible for S (or an eavesdropper) to derive a square root w of y from the information provided. Without a w, it is impossible to calculate a square root of u^2y, as required to satisfy S.

The BASIC program in listing 1 implements this system for the sake of demonstration. Both parties to the dialog are handled in separate routines of the program. The main program sets x and y, and then calls the two routines in turn using global variables to pass values between the two. A short routine, based on Euclid's algorithm, is used by both subroutines to test relative primality.

Several considerations are important to build a strong system. The first is making sure that x is large enough and has at most two factors (other than 1) of equal length. If x is prime, then the size of QR_x is $(x-1)/2$. If x is the product of primes p_1 and p_2, then QR_x has $((p_1-1)/2)((p_2-1)/2)$. (This can be proven with group theory.) Since every $y \in QR_x$ has the same number of square roots, it follows that each has only two or four square roots. This reduces the possibility of finding a square root simply by guessing.

The security of this system can be further strengthened by requiring that each prover be able to handle any of, say, 1000 different x,y pairs. The skeptic computer chooses a pair at random, and the prover must prove (in the zero-knowledge sense) that y is a member of QR_x. Having 1000 possible sets of quadratic residues adds deterrence by increasing the computational burden on any would-be intruder.

Of course, there are caveats to the particular zero-knowledge method outlined in this article, using quadratic residues. Its security relies heavily on the assumption that factoring numbers is too difficult to be done in a reasonable amount of time. If the numbers are chosen incorrectly, the system is not strong enough. Alternatively, if computer technology or mathematical theory advances sufficiently to make factoring a fast process, the quadratic residues method (and many more of today's encryption systems) will be vulnerable.

Practical Uses

Zero-knowledge proofs require a great deal of computation and thus are probably not adaptable to situations that rely

continued

A short routine, based on Euclid's algorithm, is used by both subroutines to test relative primality.

Listing 1: *A BASIC program, written in QuickBASIC, demonstrating the zero-knowledge-proof method using quadratic residues. A sample run is given at the end of the listing.*

```
RANDOMIZE
x = 100
DIM qr(100)
'qr(i)=0 if i is relatively composite to x
'     =1 if i is relatively prime to x
'     =2 if i is relatively prime and is a quadratic residue
FOR i = 1 TO x
  qr(i)=1
NEXT i
loop1:
  'Mark the primes and composites
  FOR i=2 TO x
    j=i
    k=x
    again:
      IF j MOD k = 0 THEN
        qr(i) = 0
      ELSEIF j MOD k = 1 THEN
        qr(i) = 1
      ELSE
        jj=j
        j=k MOD j
        k=jj
        GOTO again
      END IF
  NEXT i
loop2:
  'Mark the quadratic residues
  FOR i=1 TO x
    IF qr(i)>0 AND qr((i*i) MOD x)>0 THEN qr((i*i) MOD x) = 2
  NEXT i
start:
  'Select a y at random
  w = INT(x*RND)
  'Make sure it is a quadratic residue
  IF qr(w)=0 THEN GOTO start
  y = (w*w) MOD x
  PRINT USING "Prover: (Secret password w = ###)";w
  PRINT
  PRINT USING "Skeptic: (x,y) = (### , ###)"; x,y
  'x and y are global variables
  'w is known only to the prover
  'z,b,u,v are the four numbers exchanged
  'between the prover and the skeptic
  prob = 1 'Initial probability that prover is lying
  FOR try=1 TO 10
    PRINT
    PRINT "Round: ";try
    prover1:
      'Set n1=w^2 MOD x and n2 = y*w^2 MOD x
      'Randomly select a u in Z(x)
```

continued

```
      u = INT(x*RND)
      IF qr(u)=0 THEN GOTO prover1
      z=(u*u) MOD x
      PRINT USING "Prover: z = ##";z
    skeptic1:
      'Sees z and asks for square root of z
      'or square root of zy
      b=int(2*RND) 'b = 0 or 1
      Print using "Skeptic: b = #";b
    prover2:
      'Returns the correct square root
      IF b=0 THEN
        PRINT USING "Prover: u = ##";u
      ELSE
        v = (u*w) MOD x
        PRINT USING "Prover: v = ##";v
        END IF
    skeptic2:
      'Checks the prover's response
      IF b=0 AND (u*u) MOD x = z THEN
        PRINT "Skeptic: (u*u) MOD x = z: Ok."
      ELSEIF b=1 AND (v*v) MOD x=(z*y) MOD x THEN
        PRINT "Skeptic: (v*v) MOD x = (z*y) MOD x: Ok."
      ELSE
        IF b=0 THEN
          PRINT "(u*u) MOD x <> z"
          STOP
        ELSE
          PRINT "(v*v) MOD x <> (z*y) MOD x="
          STOP
        END IF
      END IF
    'Compute probability of lying
    prob=prob*.5
    PRINT "Skeptic: Probability of lying = ";prob
  NEXT try

run
Random Number Seed (-32768 to 32767)? 55
Prover: (Secret password w = 77)

Skeptic: (x,y) = (100 , 29)

Round: 1
Prover: z = 9
Skeptic: b = 1
Prover: v = 69
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = .5

Round: 2
Prover: z = 69
Skeptic: b = 1
Prover: v = 51
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = .25

Round: 3
Prover: z = 89
Skeptic: b = 1
Prover: v = 41
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = .125

Round: 4
Prover: z = 61
Skeptic: b = 0
Prover: u = 69
Skeptic: (u*u) MOD x = z: Ok.
Skeptic: Probability of lying = .0625

Round: 5
Prover: z = 89
Skeptic: b = 1
Prover: v = 59
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = .03125

Round: 6
Prover: z = 49
Skeptic: b = 1
Prover: v = 61
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = .015625

Round: 7
Prover: z = 69
Skeptic: b = 1
Prover: v = 49
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = .0078125

Round: 8
Prover: z = 89
Skeptic: b = 1
Prover: v = 9
Skeptic: (v*v) MOD x = (z*y) MOD x: Ok.
Skeptic: Probability of lying = 3.90625E-03

Round: 9
Prover: z = 21
Skeptic: b = 0
Prover: u = 11
Skeptic: (u*u) MOD x = z: Ok.
Skeptic: Probability of lying = 1.953125E-03

Round: 10
Prover: z = 49
Skeptic: b = 0
Prover: u = 93
Skeptic: (u*u) MOD x = z: Ok.
Skeptic: Probability of lying = 9.765625E-04
```

on human participation (such as the typing in of a password, or the response to a series of questions). However, in a world that is rapidly replacing paper with electronics, the zero-knowledge-proof method promises to be quite useful.

Banks, for instance, are heavily computerized businesses; increasingly, they rely on "smart cards" as a means of verifying customer identity. Under these circumstances, electronic eavesdropping can be as devastating to security and privacy as simply overhearing or glimpsing a password. The problem extends to communications between computers over the electronic networks that dominate the money markets; it is quite feasible for one computer to mimic another simply by "playing" the correct data stream. Zero-knowledge proofs may be able to help in these situations.

Zero-knowledge proofs will likely be a major factor in computer security systems of the future. ■

REFERENCES

1. Goldreich, Oded, S. Micali, and A. Wigderson. "Proofs that Yield Nothing but Their Validity." In *Proc. of the 27th Annual Symposium on the Foundations of Computer Science*. IEEE Publication, 1986.

2. Smith, John. "Public Key Cryptography." BYTE, January 1983, page 198.

3. Goldwasser, Shafi, and Silvio Micali. "Probabilistic Encryption." *Journal of Computer and System Sciences*, vol. 28, no. 2, April 1984.