

Solving $x^2 \equiv a \pmod{n}$

We have said nothing (so far) about whether one can solve $x^2 \equiv a \pmod{n}$ when n is a composite number.

We have also said nothing about *how* to solve it if it has a solution.

There are probabilistic polynomial time algorithms (Tonelli and Cipolla) to compute square roots of QR's mod p , where p is prime. They work well for numbers of hundreds of digits, but are too complicated to present here.

Here is a simple algorithm that finds square roots of QR's modulo any prime $p \equiv 3 \pmod{4}$, that is, it works for half of the primes.

If $p \equiv 3 \pmod{4}$, then the solutions to $x^2 \equiv a \pmod{p}$ are $x_1 \equiv a^{(p+1)/4} \pmod{p}$ and $x_2 = p - x_1$.

To see that this works, note that

$$x_1^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \equiv a \pmod{p}$$

since $a^{(p-1)/2} \equiv +1 \pmod{p}$ by Euler's Criterion and the fact that a is a QR mod p .

Now I will tell you how to solve $x^2 \equiv a \pmod{n}$ when $n = pq$ is the product of two primes $p \equiv q \equiv 3 \pmod{4}$, an important special case.

Separately solve $y^2 \equiv a \pmod{p}$, with solutions y_1 and y_2 , and $z^2 \equiv a \pmod{q}$, with solutions z_1 and z_2 . Then use the CRT four times to solve the four systems

$$x \equiv y_i \pmod{p}; x \equiv z_j \pmod{q}$$

for $i = 1, 2; j = 1, 2$. This will produce *four* different roots to $x^2 \equiv a \pmod{n}$.

An application of finding square roots modulo n is the Rabin-Blum Oblivious Transfer or Coin Flipping Protocol. In it, Alice reveals a secret to Bob with probability 0.5.

In the Oblivious Transfer version, Alice doesn't know whether Bob got the secret or not (and this outcome must be acceptable to both participants).

In the Coin Tossing version, Bob tells Alice whether he got the secret. He wins the coin toss if he did get it; loses otherwise.

Alice's secret is the factorization of a number $n = pq$ which is the product of two large primes $p \equiv q \equiv 3 \pmod{4}$.

1. Alice sends n to Bob.

2. Bob picks a random x in $\sqrt{n} < x < n$ with $\gcd(x, n) = 1$. Bob computes $a = x^2 \pmod{n}$ and sends a to Alice.

3. Knowing p and q , Alice computes the four solutions to $x^2 \equiv a \pmod{n}$. They are $x, n-x, y$ and $n-y$, for some y . These are just four numbers to Alice. She doesn't know which ones are x and $n-x$. She chooses one of the four numbers at random and sends it to Bob.

4. If Bob receives x or $n-x$, he learns nothing. But, if Bob receives y or $n-y$, he can factor n by computing $\gcd(x+y, n) = p$ or q .

Why can Bob factor n if he gets y or $n-y$?

Theorem. If $n = pq$ is the product of two distinct primes, and if $x^2 \equiv y^2 \pmod{n}$, but $x \not\equiv \pm y \pmod{n}$, then $\gcd(x+y, n) = p$ or q .

Proof: We are given that n divides $(x+y)(x-y)$ but not $(x+y)$ or $(x-y)$. Hence, one of p, q must divide $(x+y)$ and the other must divide $(x-y)$.

Zero-Knowledge Proofs

This protocol is closely related to the oblivious transfer protocol.

The difference is that Alice wants to convince Bob that she knows the factors of $n = pq$, but does not want to reveal the factors to Bob.

Alice (the prover) convinces Bob (the verifier) that she knows the prime factorization of a large composite number n , but does not give Bob any hint which would help him find the factors of n . That is, if M is a message which tells the factors of n , and S is the set of messages exchanged by Alice and Bob during the zero-knowledge proof, then $H_S(M) = H(M)$. This means that Bob learns nothing about the factorization of n during the protocol that he could not have deduced on his own without Alice's help.

Roughly speaking, Bob gives Alice some quadratic residues modulo n and Alice replies with their square roots. The difficulty with this simple approach is that when Alice replies to Bob with a square root, there is a 50% chance that she will reveal the factorization of n to Bob, as in the oblivious transfer protocol. It is known that computing square roots mod n is polynomial-time equivalent to factoring n .

Here is a good way to do the zero-knowledge proof protocol:

Alice knows n , p and q . Bob knows n but not p or q .

1. Alice chooses a in $\sqrt{n} < a < n$ and computes $b = a^2 \bmod n$.

2. At the same time, Bob chooses c in $\sqrt{n} < c < n$ and computes $d = c^2 \bmod n$.

3. Alice sends b to Bob and Bob sends d to Alice.

4. Alice receives d and solves $x^2 \equiv bd \pmod{n}$.

(Note that this is possible because bd is a QR and she can compute its square root because she knows the factors of n .) Let x_1 be one solution of this congruence.

5. At the same time, Bob tosses a fair coin and gets Heads or Tails each with probability 0.5. Bob sends H or T to Alice.

6. If Alice receives H, she sends a to Bob. If Alice receives T, she sends x_1 to Bob.

7. If Bob sent H to Alice, then he receives a from Alice and checks that $a^2 \equiv b \pmod{n}$. If Bob sent T to Alice, then he receives x_1 from Alice and checks that $x_1^2 \equiv bd \pmod{n}$.

Alice and Bob repeat steps 1 through 7 many (20 or 30) times.

If the check in step 7 is always okay, then Bob accepts that Alice knows the factorization of n .

But if Alice ever fails even one test, then Bob concludes that Alice is lying.

Why does this protocol work?

Why does Bob not learn the factors of n ?