The Hardness of Factoring

The security of RSA rests on the assumption that the problem of factoring a large number assumption? At first glance it doesn't seem like a particularly challenging problem; we've a math classes before. However, it turns out that as the number grows larger, it becomes ex factors. In 2009, researchers took two years to factor a 232-digit number, and this was us the most efficient algorithms we know! It would seem, then, that factorization is a *comput* and that building a cryptosystem on the foundation of factorization would be a safe bet. Si that RSA cryptography is computationally secure, this is a good sign.

[cartoon-hardness-1] [cartoon-hardness-2]

But how can we show mathematically that factoring is a hard problem? Sure, we could ma Shamir, and Adleman do in the RSA paper) that hundreds of great mathematical minds hav factorization, dating back to Fermat in the 1600s, and no one has come up with an efficier we'd really like to have some mathematical guarantees here.

The field of computational complexity theory is devoted to determining how difficult mathe solve. In complexity theory, we try to assign a given problem to a class, or a set of problen We show that two problems are comparably hard by performing a *reduction*, that is, showi turned into an instance of problem B, solving problem B, and converting the result into a so reduction of A to B. When we reduce A to B, we are essentially saying "Instances of proble instances of type B".

[cartoon-hardness-3]

But how do we *know* whether a problem is intrinsically hard, or whether it only seems hard algorithm or function that hasn't been discovered yet? After all, we used to think that testi or not (primality testing) was "hard" — but then in 2005 the first provable polynomial-tim primality (the AKS test) was put forth by three Indian mathematicians. Is RSA just one alg from being insecure?

To talk about the relative hardness of different problems, we use *complexity classes*. The t are **P** and **NP**. P contains all of the problems that can be deterministically solved in a polyn is, as the size of the problem increases, the time it takes to solve that problem increases p you can express the time it will take to solve a problem of size n in an equation that is pol

NP stands for non-deterministic polynomial time. All of its members have answers that ca time, but unlike P, it is not guaranteed that solving a NP problem will take polynomial time the computer science and math community over whether P=NP, that is, whether every pro polynomial time can also be solved in polynomial time. The problem is of such great impor selected as one of the Millennium Prize Problems, for which the first correct solution will ea and fortune await!

So where does our factorization problem rank in the world of complexity classes? Well, we verify a solution once we have one — we simply multiply the two numbers of the solution, number we were trying to factor. So factorization is in NP. However, we don't have a polyn it (yet) so for now we believe factorization is not in P. This is good, because we want facto polynomial time. But if anyone proves that P=NP, then factorization would be come an eas become insecure.

How would being able to do efficient factorization break RSA's security? Well, recall that th combination of n, the product of two secret primes p, q, and an exponent e. If an adversary could then compute $\phi(n) = (p-1)(q-1)$. Knowing $\phi(n)$, the adversary can then compute t know from the RSA algorithm that $(d * e) \mod \phi(n) = 1$, so we just need to compute the m $\mod \phi(n)$ in order to compute the private key. Once the adversary has Alice's private key, messages sent to Alice.