Name(s)____ Period ______ Date __________________

Resource - How and Why Does the Public Key Crypto Work?

How does the *Public Key Crypto Widget* actually work?

Reference: Public Key Crypto Widget

0. **Modulo is commutative**
First, a fact about modulo that's important to realize: If you MOD a number once, the result is less than the modulus. Therefore, if you MOD a number *and then* MOD the result of *that*, you get the same number. Any number that is less than the modulus, when MODed, results in itself. For example:

23 MOD 17 = 6 → 6 MOD 17 = 6 → 6 MOD 17 = 6…..and so on.

Also, modulo is commutative, which means that if you multiply some numbers and MOD the result, that has the same effect as MODing all the numbers in the first place, then multiply them, then MODing the result of that.

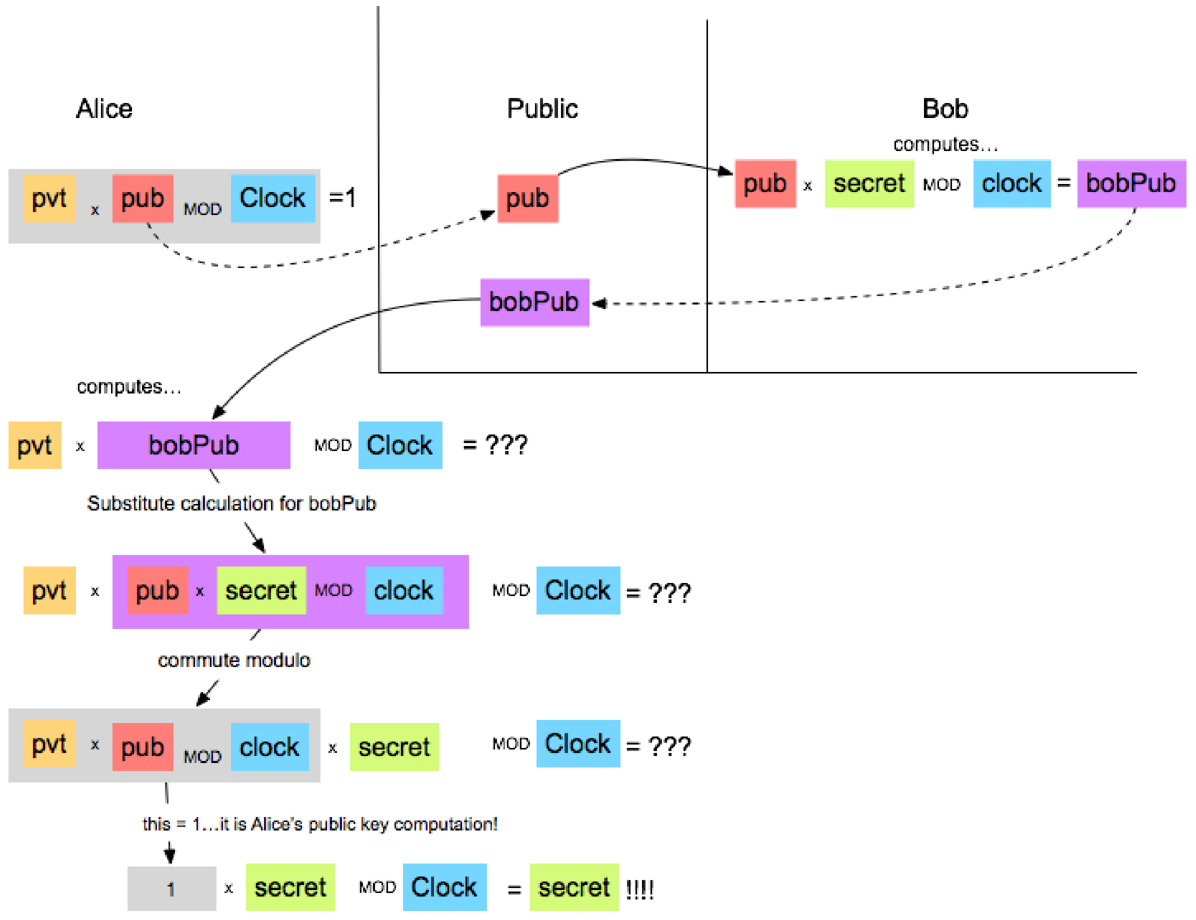
(27 x 49) MOD 17 == 14 ---> [(27 MOD 17) x (49 MOD 17)] MOD 17 == 14

1. **Prime numbers**
The numbers we let you use for modulus (clock size) are all prime numbers. Prime numbers are useful because no numbers divide them, except for 1 and themselves. Modulo is a form of division. So if we only divide values by prime numbers, the results will be unique and distributed evenly over the range of possible values.

2. **Modular multiplicative inverse**
Alice's public key is not random; it is generated based on the modulus (clock size) and her choice of private key. Alice's public key is calculated in a special way. The public key times the private key MOD the clock should = 1.

(pvt * pub) MOD clock = 1.

If you keep reading, you'll see why we want to calculate it this way. But the math-y term for what we're calculating is the "modular multiplicative inverse" of Alice's private key. Let's now look at the rest of the exchange. You can substitute numbers for these values to confirm it checks out.


Alice
Public
Bob
computes…
pvt x pub MOD Clock =1
pub
pub x secret MOD clock = bobPub
bobPub
computes…
pvt x bobPub MOD Clock = ???
Substitute calculation for bobPub
pvt x pub x secret MOD clock MOD Clock = ???
commute modulo
pvt x pub MOD clock x secret MOD Clock = ???
this = 1…it is Alice's public key computation!
1 x secret MOD Clock = secret !!!!

Is this *actually* computationally hard for Eve to crack?

The answer is, “no, not really.” It's hard for a *human* to figure out, because a feature of modular multiplicative inverses is that they are a unique pair for every modulus you could choose AND they are randomly distributed across the number line. So that means there's no heuristic for "getting close"; if you discover a number that gets you one away from your target, you are no closer than if you were 1000 away. So you can't narrow down the field; the only thing you can do is try every possibility. Since we only had 4-digit numbers, you could brute force attack the problem in seconds with a computer program. (In fact, that's what our widget does.) Even for large numbers, it wouldn't be hard to find the multiplicative inverse.

How close is this to the real thing?

0. Our scheme here most closely resembles the RSA public key encryption system. (RSA doesn’t stand for anything related to cryptography; it’s just the initials of the last names of the 3 people who invented it: Rivest, Shamir, and Adleman.)

1. RSA does rely on multiplication and modulo, but instead of just multiplying numbers plainly, the private keys and secret numbers are used as *exponents*, which makes even bigger numbers that are even harder to reverse after modulo.

2. Rather than finding the multiplicative inverse, to crack RSA you need to find the prime factors of a very large number (i.e., given a large number, find two prime numbers that, when multiplied together, yield that number). This is a much harder problem (computationally) than finding the multiplicative inverse (which is what our widget does).

3. The real thing uses VERY large numbers. In this widget, the biggest numbers only had 4 digits (~13-14 bits). The actual numbers used are over 75 digits long (256 bits)! If you think you can fathom how big a number with 77 digits is, you can't. The distance to the edge of our solar system in *inches* is only a 14-digit number.