What the heck is a zero-knowledge proof of knowledge, anyway, and what does it mean?

D.R. Stinson

May 25, 2016

Suppose that Alice has a private key, a, and she wishes to prove to Bob that she knows the value of a. Corresponding to a there is a public key, v. There is a known mathematical relationship between a and v, but we assume that it is computationally infeasible to compute a given v. On the other hand, given v and a, it is easy to verify that a is the private key corresponding to v.

For example, it might be the case that $v = \alpha^a$ in some abelian group in which the discrete log problem is infeasible. It is easy to verify that $v = \alpha^a$, but if much more difficult to compute a, given v and α.

Alice and Bob will engage in some kind of challenge-and-response protocol. Actually, Alice initially sends a *commitment*, γ, to Bob. Bob sends his *challenge*, r, to Alice, and then Alice generates an appropriate *response*, y. Bob verifies that Alice's response y is valid, and if it is, he accepts. Usually Alice will use her private key to compute her response, y, as a function of γ and r; and Bob will use Alice's public key, v, to verify that the reponse y is valid.

As mentioned earlier, we are making the following intractibility assumption:

(0) It is infeasible to compute Alice's private key, a, given her public key, v.

Some interactive protocols often satisfy the following two properties:

(1) proof of knowledge, and

(2) zero-knowledge.

An interactive protocol is a *proof of knowledge* if it is impossible (except with a very small probability) to impersonate Alice without knowing the value of Alice's key. This means that the only way to "break" the protocol is to actually compute a. This in itself does not mean that the protocol is secure, however. For example, consider the following Silly Protocol 1:

(i) Alice chooses a random γ and sends it to Bob.

(ii) Bob chooses a random $r \neq 0$ and sends it to Alice.

(iii) Alice computes $y = a\,r + v\gamma$ and sends it to Bob.

(iv) Bob computes $a = (y - v\,\gamma)/r$, and checks that a is the private key corresponding to the public key v. If so, he ouptuts "accept".

Silly Protocol 1 is a proof of knowledge. However, it is obvious that it is insecure, because Bob (or anyone else, for that matter) can compute Alice's secret key after Alice identifies herself to Bob. The problem is that the protocol reveals the value of the private key, which is a bad thing! What we want is a protocol that does not reveal the value of the private key, or any information about it.

The example considered above motivates the second concept. A protocol is termed *zero-knowledge* if it reveals no information about Alice's private key. Stated another way, computing Alice's private key is not made easier by taking part in the protocol (in Bob's role as the verifier) in some specified number of sessions. Silly Protocol 1 is clearly not zero-knowledge.

Consider the following Silly Protocol 2:

(i) Alice chooses a random γ and sends it to Bob.

(ii) Bob chooses a random r and sends it to Alice.

(iii) Alice computes $y = v + r + \gamma$ and sends it to Bob.

(iv) Bob checks that $y = v + r + \gamma$. If so, he outputs "accept".

This protocol is clearly zero-knowledge, but it is not a proof of knowledge. Alice's private key is never used in the protocol, and therefore anyone can impersonate Alice.

We claim that a protocol that satisfies **(0)**, **(1)** and **(2)** is provably secure. The attack model is that the adversary, Oscar, is allowed to play the role of Bob in several sessions of the protocol. Then Oscar tries to impersonate Alice, and his goal is to have Bob "accept". The protocol is provably secure, provided that the computational assumption **(0)** is valid.

Let us justify the claim that a protocol that satisifes the three conditions listed above is secure. Suppose that Oscar can successfully impersonate Alice in the given attack model. Then we can make the following inferences:

- Because the protocol is a proof of knowledge, it must be the case that Oscar has computed Alice's private key.
- Because the protocol is zero-knowledge, Oscar's computation of Alice's private key is not made easier by his having taken part in earlier sessions of the protocol.
- Therefore Oscar must have been able to compute Alice's private key, knowing only the value of her public key.
- This contradicts the intractibility assumption.
- We conclude that it is impossible for the adversary to impersonate Alice, and therefore the protocol is secure.