Zero Knowledge Proofs

A zero-knowledge proof involves two parties: a prover and a verifier. The prover wishes to convince the verifier about the truth of some assertion. For example, the prover might wish to convince the verifier that a number N has a prime factor between K_1 and K_2. i.e. there is a prime P: $K_1 \leq P \leq K_2$ & $P|N$. A zero-knowledge proof of this assertion results in the verifier being convinced that the assertion is true, but getting no further information. For example, if the prover were to simply send P to the verifier in the above example, this would be a proof of the assertion, but not a zero-knowledge proof – since the verifier learns the value of P, which he did not know before. One property of zero-knowledge proofs is that the verifier cannot ~~convince~~ turn around and convince any other party about the truth of the assertion, since he learnt no information other than the fact that the assertion is true.

Challenge-response systems: Passwords

Let us examine a password based access control scheme. For example, in the Unix password scheme, the server stores for each user's password, x, the value $y = f(x)$. Here f is a one-way function applied to x. When the user types in x, the server computes $f(x)$ and compares to the stored value. There are a number of problems with this scheme:

1) x is being sent to the server in the clear.
2) If the server is briefly compromised, the user's password is compromised forever.
3) The scheme is vulnerable to a man-in-the-middle attack.

Ideally we would use a zero-knowledge protocol to implement a challenge-response password system. The user plays the role of the prover and the server plays the verifier. The user proves to the server that he knows $x : f(x) = y$. The conversation leaks no ~~information~~ information about x. So if the server is temporarily compromised, no future harm is done. The same holds for a man-in-the-middle attack. The

attacker cannot turn around and convince the server at a future date that he knows x.

Notice that in this example, the prover is proving that he "knows x". This is a zero-knowledge proof of knowledge.

A puzzle:

Zero-knowledge proofs are a very counter-intuitive concept. Let us first illustrate the concept in a simple setting of a kid's puzzle "Where's Waldo?" This is a puzzle book where each page consists of a very detailed picture with many characters. The goal is to find Waldo in this picture. Assume that the prover has solved the puzzle and wishes to prove to the verifier that she has without revealing any information to him about the location of Waldo. Here is how she would proceed:

She takes a piece of cardboard twice as large as the picture and cuts a small flap in it, just big enough to reveal the figure of Waldo. She then positions the picture behind the ~~flap~~ cardboard so that Waldo is under the (invisible) flap. The verifier gets one of two challenges: he can either ask to see Waldo — in response the prover reveals the flap and opens it to show Waldo. Or he asks to see that the cardboard is really hiding the puzzle they are working on. In this case, the prover removes the entire cardboard to reveal the original picture.

Let us analyze this protocol:

1) Why is the verifier convinced that the prover knows where ~~E~~ Waldo is?
Intuitively it is because she was willing to answer either of the two challenges. Answering both challenges — revealing the picture as well as the location of the flap

reveals ~~the~~ Waldo's location. We can formalize this via a knowledge verifier. The knowledge verifier is allowed to backup in the protocol. So after challenging her to open the flap, he says "no, I changed my mind". Can you reveal the picture to me". Since the prover was willing to answer either challenge, the knowledge verifier is asking for information that the prover must have to successfully carry ~~out~~ out the protocol. But this information is sufficient to locate Waldo. So we can conclude that the prover really knew Waldo's location.

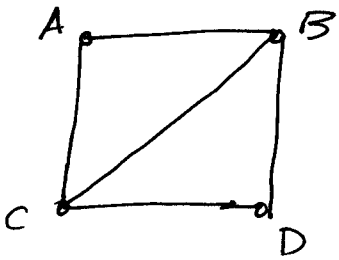
2) Why is the protocol zero-knowledge? For this we must assume that the picture of Waldo is known to the verifier. Only his location is unknown. The verifier's view of the "proof" consists of either just the picture of Waldo behind a flap or the entire puzzle. But he could generate either view by himself without the help of the prover.

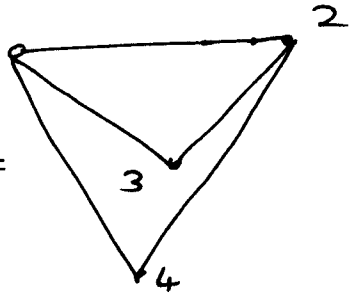
To summarize, the verifier is convinced because the prover was willing to answer either challenge. He gets zero-knowledge because she answers only one of the challenges.

Zero-knowledge Proof of Graph Isomorphism.

Graph Isomorphism: Given two graphs G_1, G_2 on n vertices each, is there a ~~mapping~~ bijection between vertex sets V_1 & V_2, ($\pi : V_1 \rightarrow V_2$), such that $(w_1, w_2) \in E_1$ iff $(\pi(w_1), \pi(w_2)) \in E_2$.

example:

$G_1 =$ 

$G_2 =$ 

G_1 & G_2 are isomorphic. The bijection

$\pi(A) = 3 \quad \pi(B) = 1 \quad \pi(C) = 2 \quad \pi(D) = 4$

is an isomorphism.

Graph isomorphism is a hard problem. We don't know of any ~~subexponential~~ fast algorithm

to solve this problem on worst-case inputs. Suppose the prover knows an isomorphism, π, between G_1 & G_2. One way she could convince the verifier is by sending him the bijection π. But this is obviously not zero-knowledge. Here is a zero-knowledge protocol:

Prover	Verifier
Pick a random permutation σ, and compute $\sigma(G_2) = H$	
send $H = \sigma(G_2)$ →	
	← flip coin to select 1 or 2
if 1: reveal $\sigma\pi$ →	
if 2: reveal σ →	

The prover picks a random permutation of G, call it H. She sends this to the verifier. The verifier challenges her to show either 1) that H is isomorphic to G_1 or 2) that it is isomorphic to G_2. 2) is easy since σ provides the isomorphism. 1) is equally easy since the prover knows that π maps G_1 to G_2 and so $\sigma\cdot\pi$ maps G_1 to H.

Knowledge-verifier: The knowledge verifier asks the prover to answer both challenges. So he gets both σ and $\sigma\pi$. From these it is easy to recover the isomorphism π between G_1 & G_2: since $\pi = \sigma^{-1}\cdot(\sigma\pi)$. This proves that the prover "must know" an isomorphism between G_1 & G_2.

Zero-knowledge: Why is the protocol zero-knowledge? We must show that no matter what the verifier does, we can simulate the verifier's view without interacting with the protocol. A key observation is that if σ is a random permutation. Then so is $\sigma\circ\pi$. So our simulator picks a random permutation σ_2 with probability $\frac{1}{2}$ & computes $H = \sigma_2(G_2)$, and with probability $\frac{1}{2}$ picks a random permutation σ_1^* and computes $H = \sigma_1^*(G_1)$. The simulator sends this to the verifier who selects either 1) or 2). With probability $\frac{1}{2}$ the choice, i, matches that of the simulator, who can return σ_i. With probability $\frac{1}{2}$ we have to rerun the simulation. Thus the simulation takes twice as long, but provides exactly the same view.