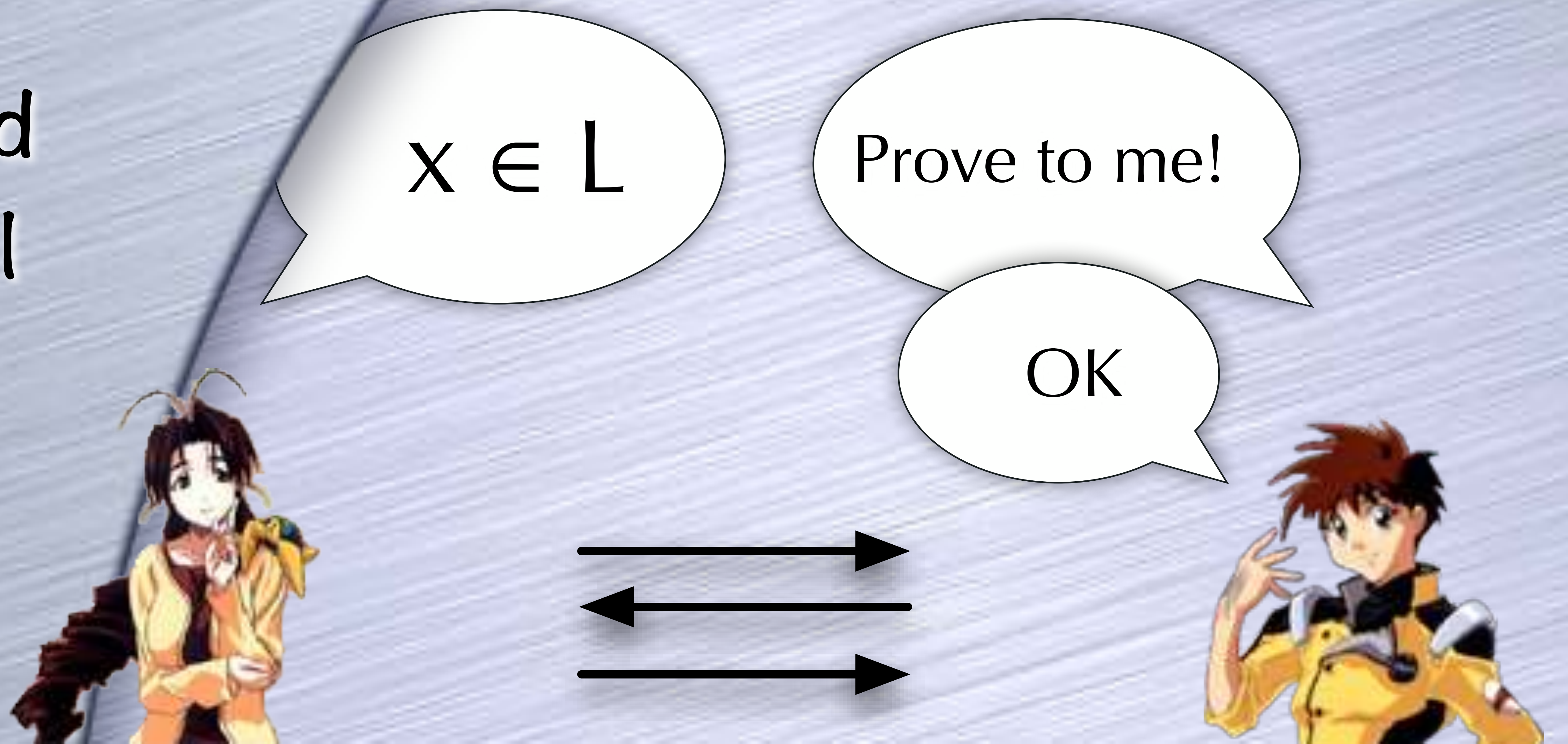
Zero Knowledge Proofs

Lecture 12

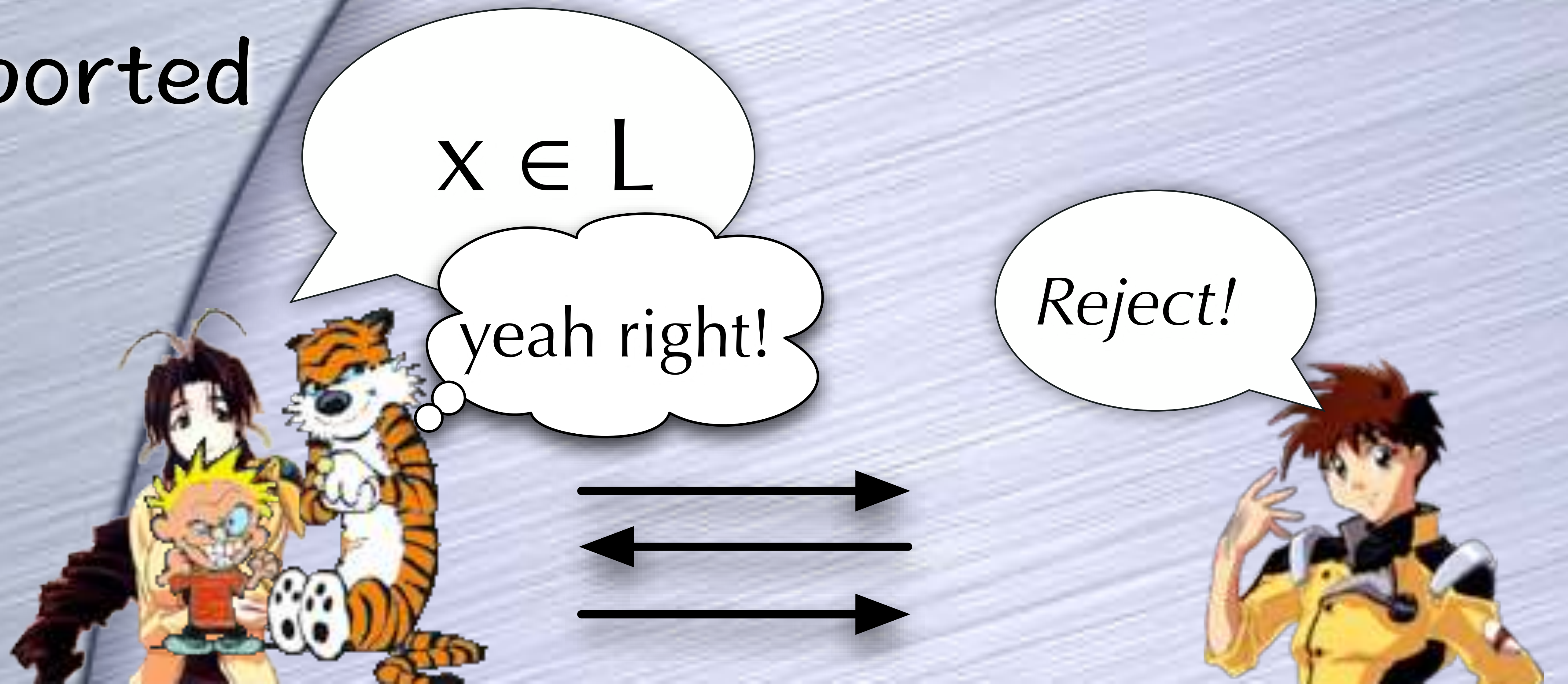
Interactive Proofs

- **Prover** wants to convince **verifier** that x has some property
  - i.e. x belongs to some set L ("language" L)
- Computationally bounded verifier, but all powerful prover (for now)



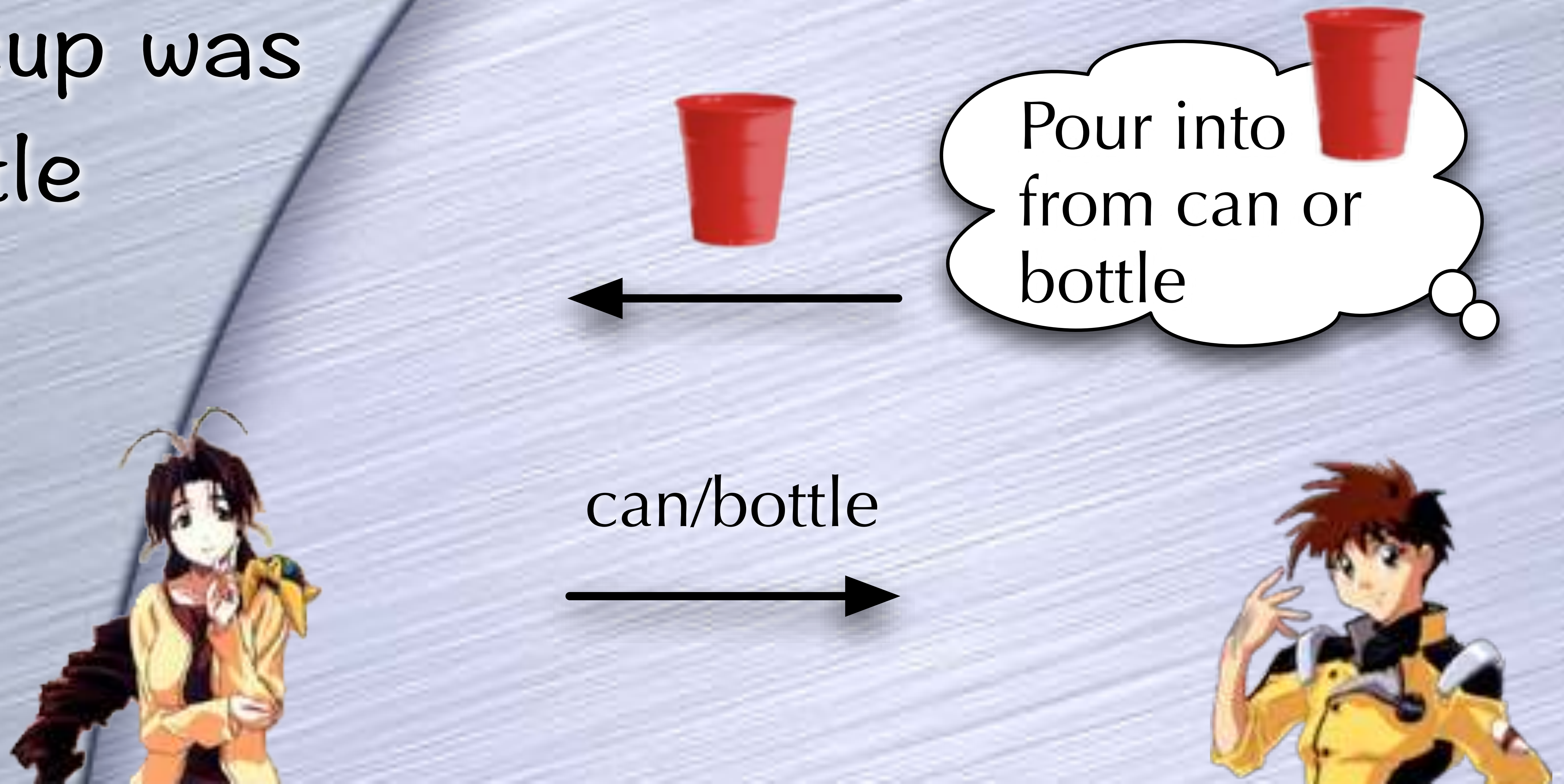
Interactive Proofs

- Completeness
  - If x in L, honest Prover will convince honest Verifier
- Soundness
  - If x not in L, honest Verifier won't accept any purported proof



An Example

- Coke in bottle or can
  - Prover claims: coke in bottle and coke in can are different
- IP protocol:
  - prover tells whether cup was filled from can or bottle
  - repeat till verifier is convinced



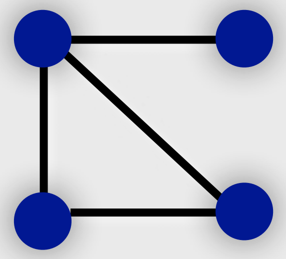
An Example

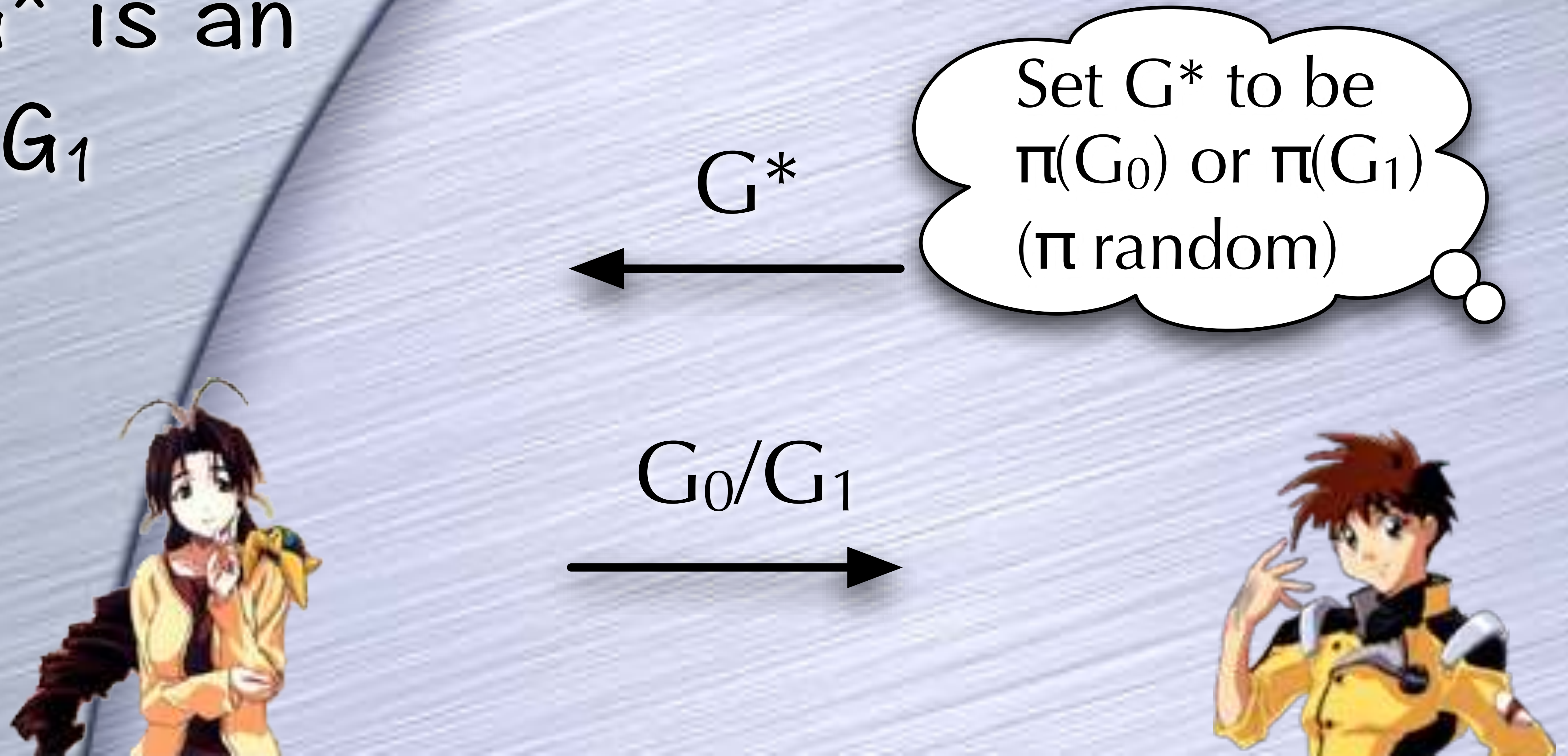
- Graph Non-Isomorphism
  - Prover claims: G_0 not isomorphic to G_1
- IP protocol:
  - prover tells whether G^* is an isomorphism of G_0 or G_1
  - repeat till verifier is convinced

Isomorphism: Same graph can be represented as a matrix in different ways:

e.g. $G_0 = \begin{matrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}$ and $G_1 = \begin{matrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{matrix}$

both are isomorphic to the graph represented by the drawing

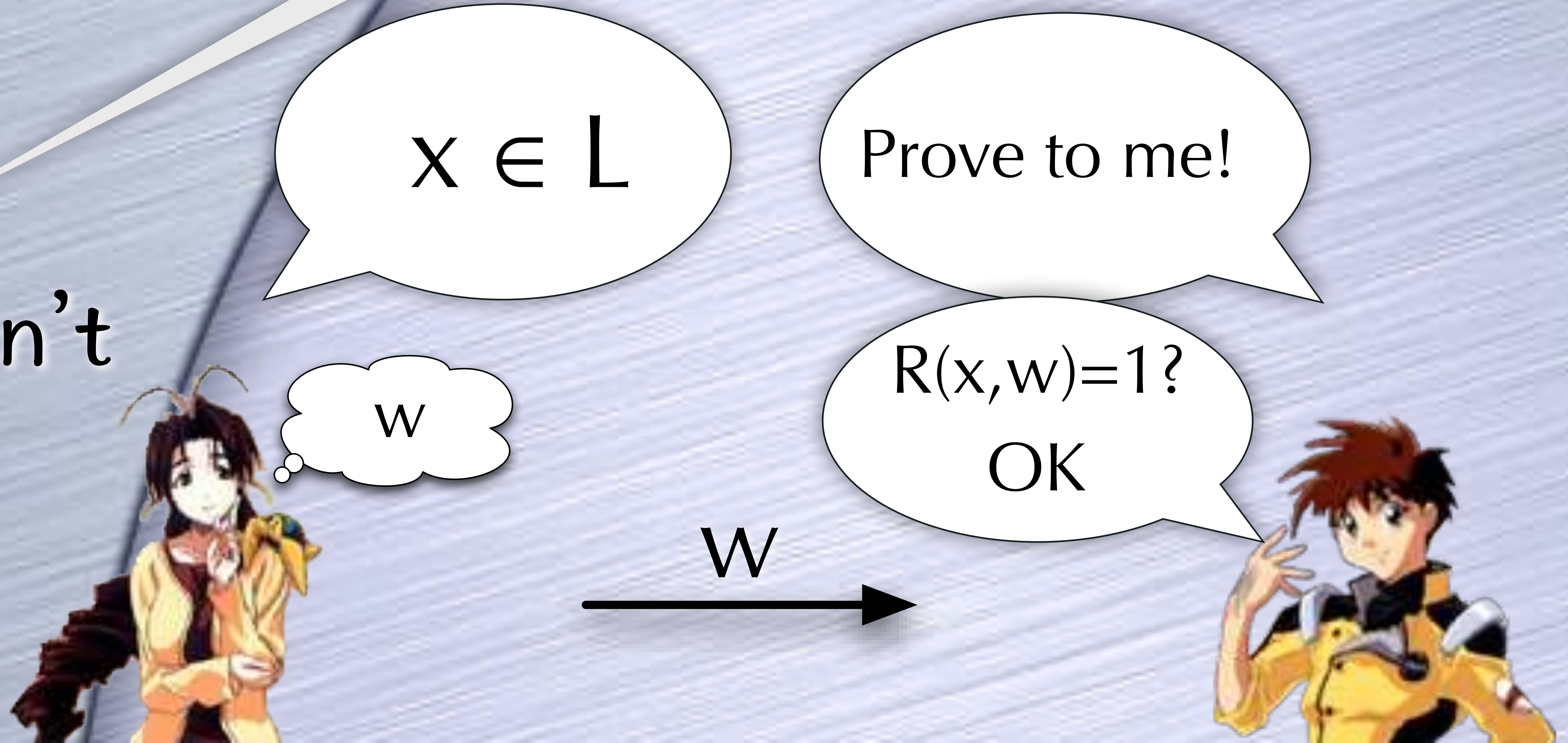




Proofs for NP languages

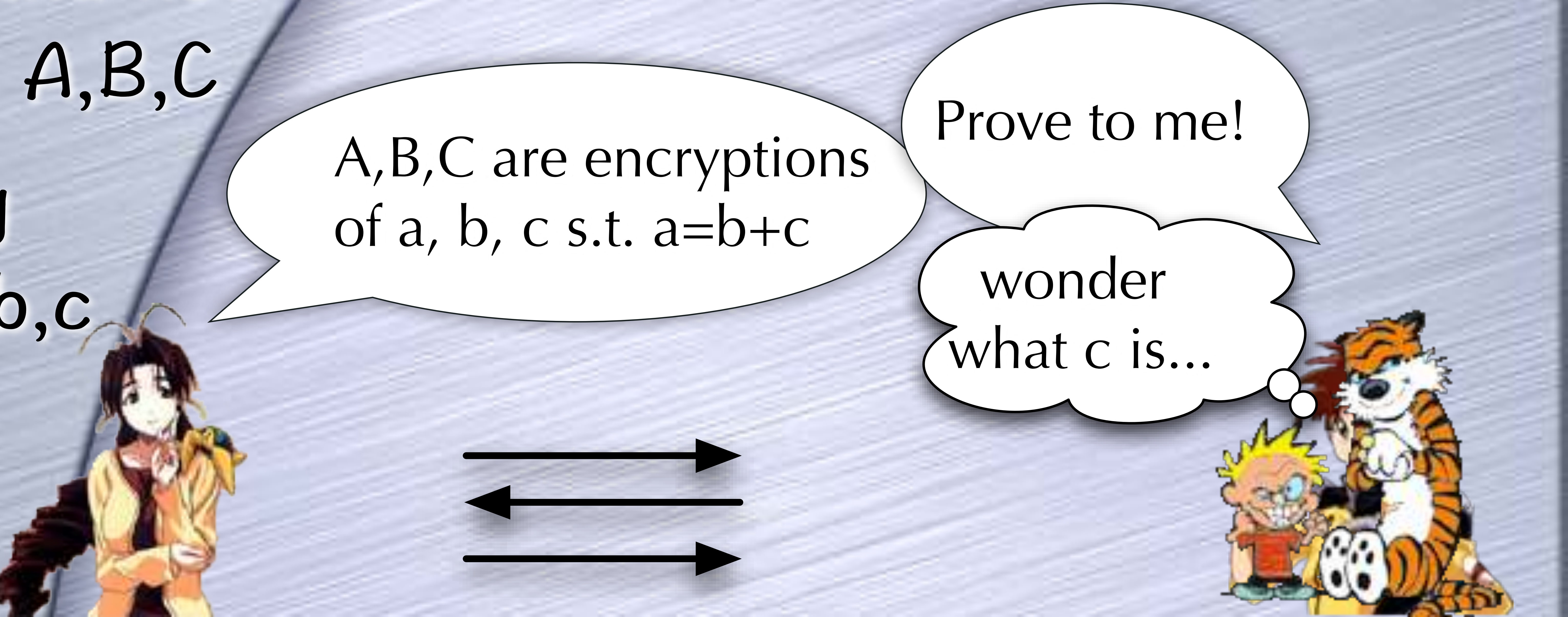
- Proving membership in an NP language L
  - $x \in L$ iff $\exists w$ $R(x,w)=1$ (for R in P)
    - e.g. Graph Isomorphism
- IP protocol:
  - prover just sends w
- But what if prover doesn't want to reveal w?

NP is the class of languages which have <u>non-interactive</u> and <u>deterministic</u> proof-systems



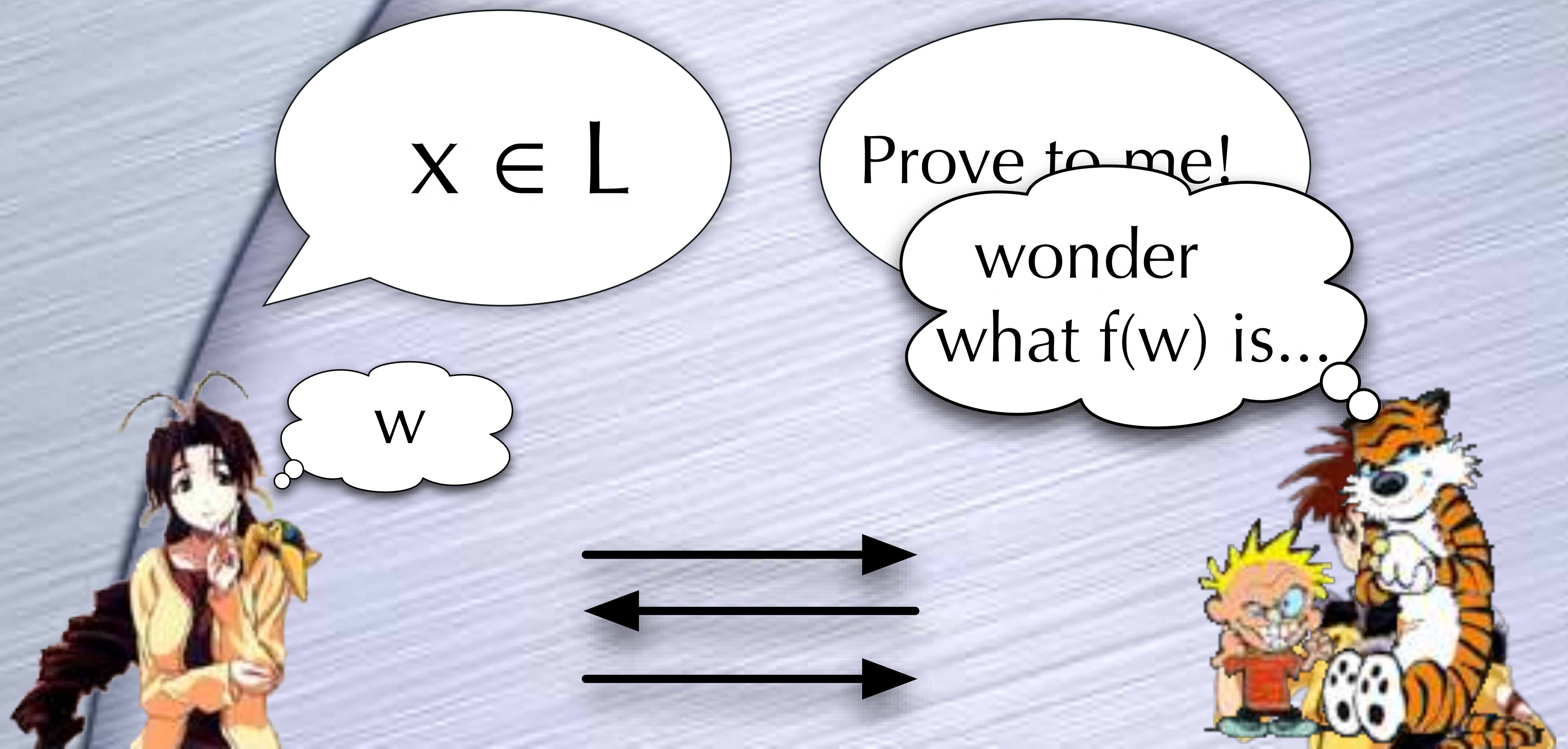
Zero-Knowledge Proofs

- In cryptographic settings, often need to be able to verify various claims
  - e.g., 3 encryptions A,B,C are of values a,b,c s.t. a=b+c
  - Option 1: reveal a,b,c and how they get encrypted into A,B,C
  - Prove without revealing anything at all about a,b,c except that a=b+c ?



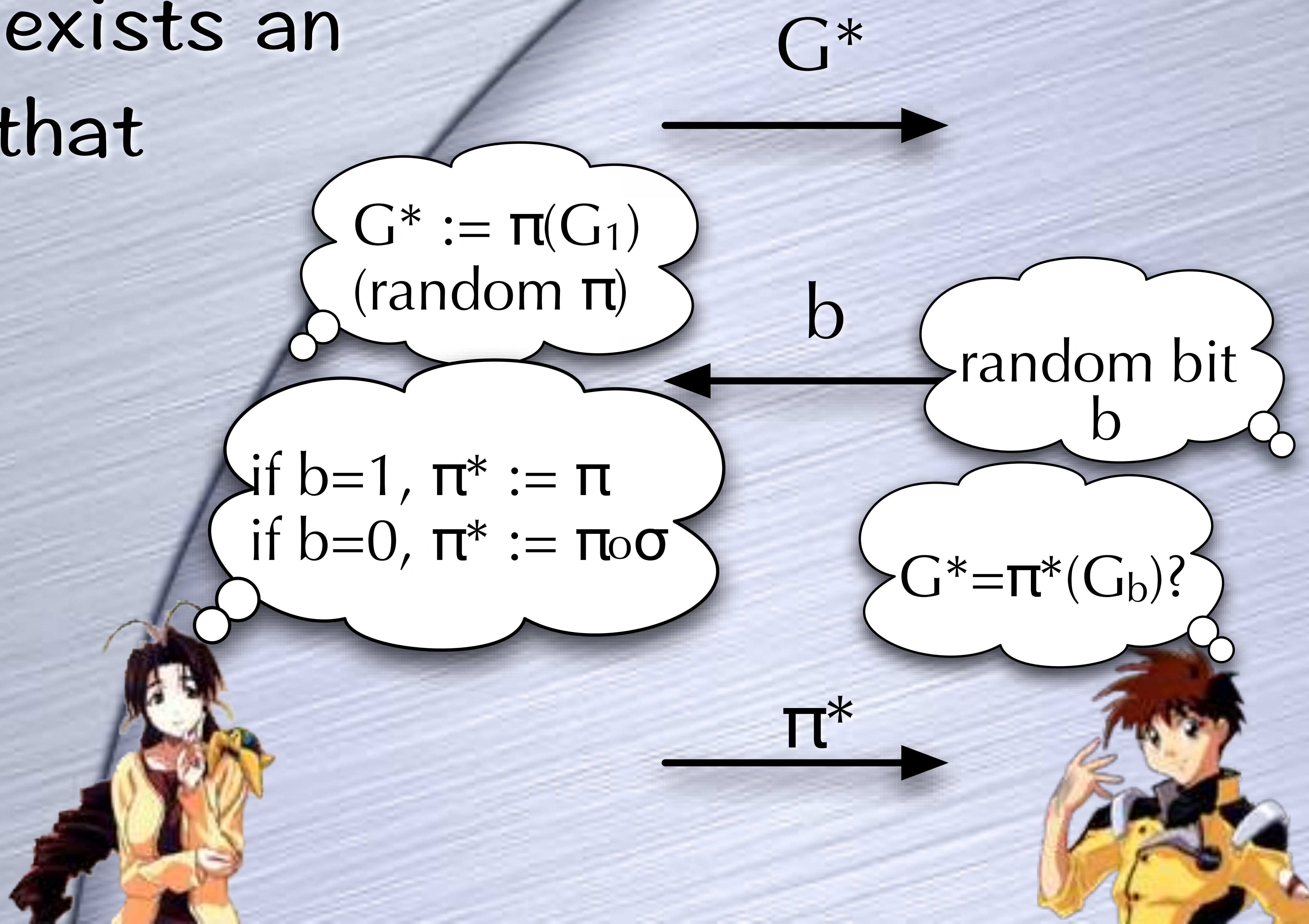
Zero-Knowledge Proofs

- Verifier should not gain any knowledge from the honest prover
  - except whether x is in L
- How to formalize this?
  - Simulation!



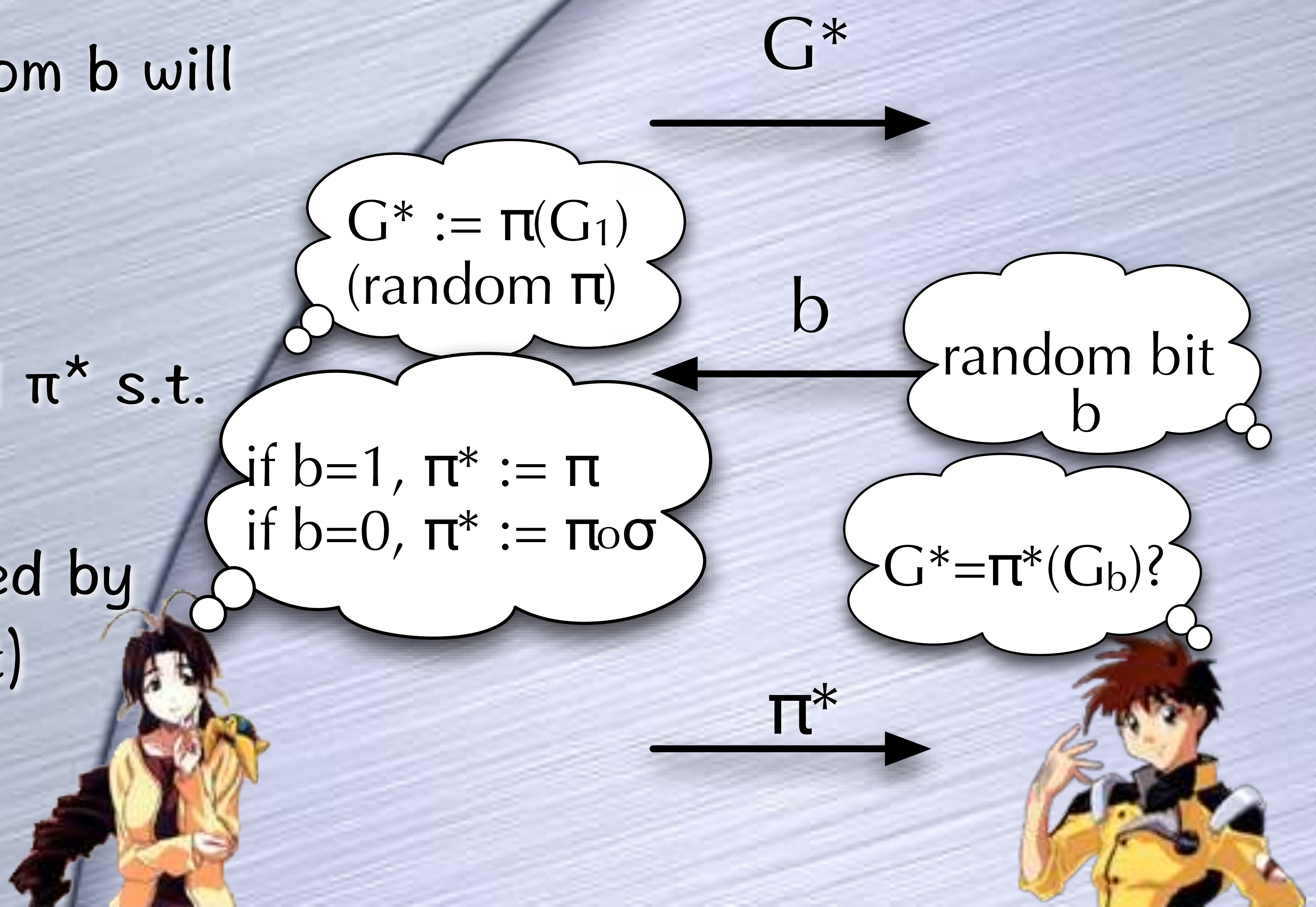
An Example

- Graph Isomorphism
  - (G_0,G_1) in L iff there exists an isomorphism σ such that $\sigma(G_0)=G_1$
- IP protocol: send σ
- ZK protocol?



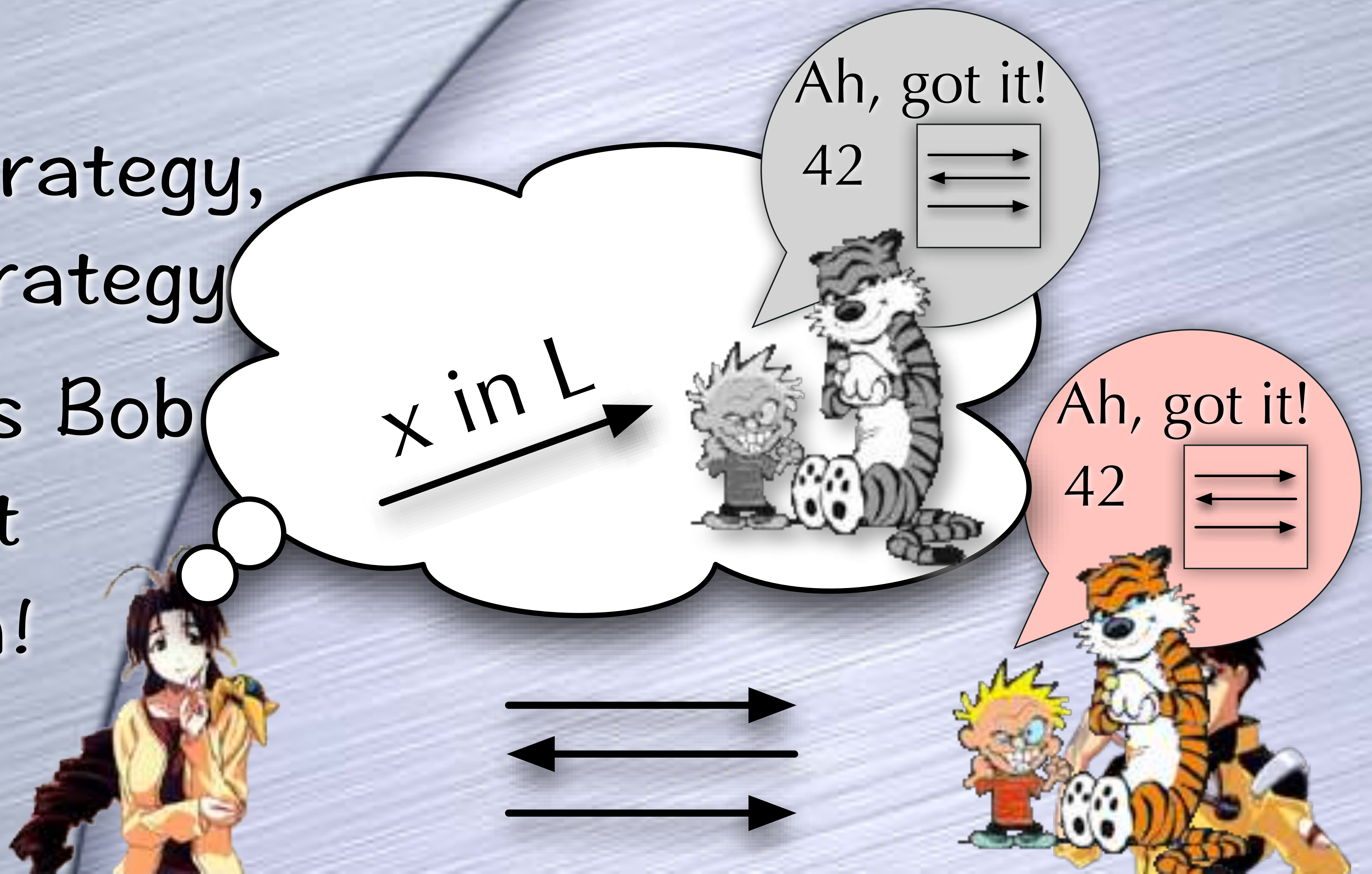
An Example

- Why is this convincing?
  - If prover can answer both b's for the same G* then $G_0 \sim G_1$
  - Otherwise, testing on a random b will leave prover stuck w.p. 1/2
- Why ZK?
  - Verifier's view: random b and π* s.t. $G^* = \pi^*(G_b)$
  - Which he could have generated by himself (whether $G_0 \sim G_1$ or not)



Zero-Knowledge Proofs

- Interactive Proof: Complete and Sound
- And has ZK Property:
  - Verifier's view could have been "simulated"
  - For every adversarial strategy, there is a simulation strategy
- Even though the view gives Bob no additional knowledge, it convinces him of the claim!



The Legend of William Tell

A Side Story

Bob: *William Tell is a great marksman!*

Charlie: *How do you know?*

Bob: *I just saw him shoot an apple placed on his son's head! See this!*

Charlie: *That apple convinced you? Anyone could have made it up!*

Bob: *But I saw him shoot it...*

The Legend of William Tell

A Side Story

***Bob**: William Tell is a great marksman!*

***Charlie**: How do you know?*

***Bob**: I just saw him shoot an apple placed on his son's head! See this!*

***Charlie**: That apple convinced you? Anyone could have made it up!*

***Bob**: But I saw him shoot it...*

Bob: G_0 and G_1 are isomorphic!

Charlie: How do you know?

Bob: Alice just proved it to me! See this:

$$G^*,\ b,\ \pi^* \text{ s.t. } G^* = \pi^*(G_b)$$

Charlie: That convinced you? Anyone could have made it up!

Bob: But I picked b at random and she had no trouble answering me...

Simulation

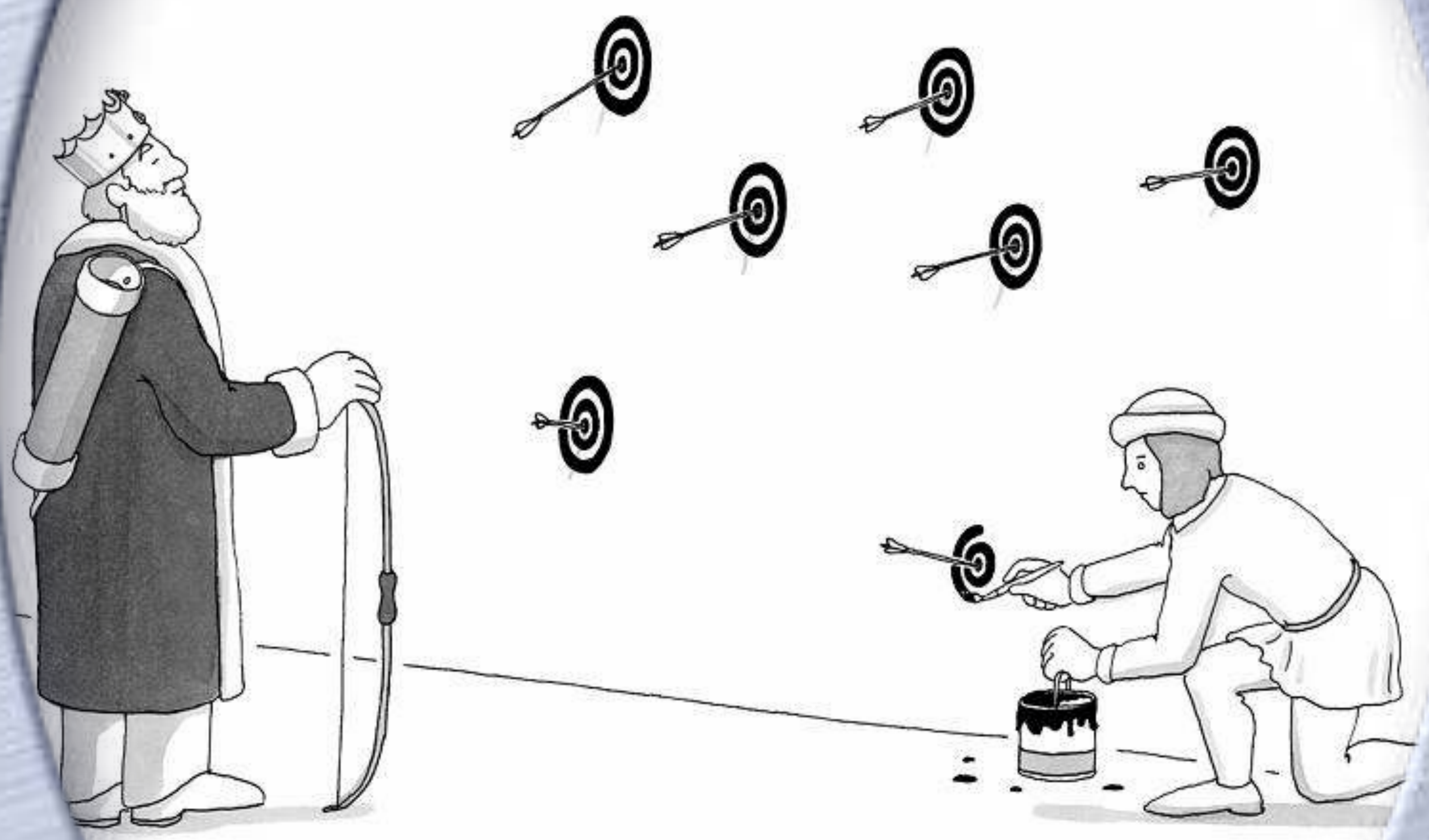
Another Analogy

- Shooting arrows at targets drawn randomly on a wall

  vs.

- Drawing targets around arrows shot randomly on to the wall

- Both produce identical views, but one of them is convincing of marksmanship

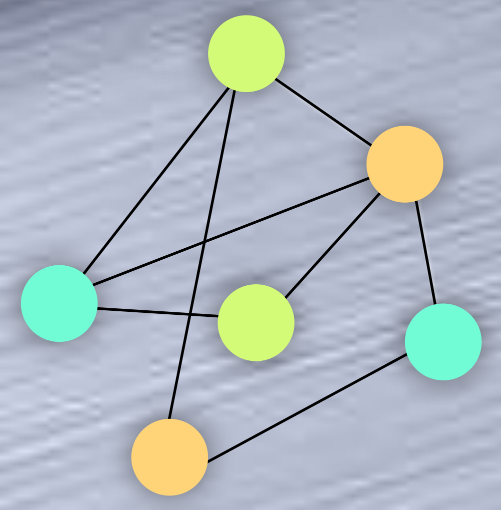


by CHARLIE HANKIN New Yorker Cartoons

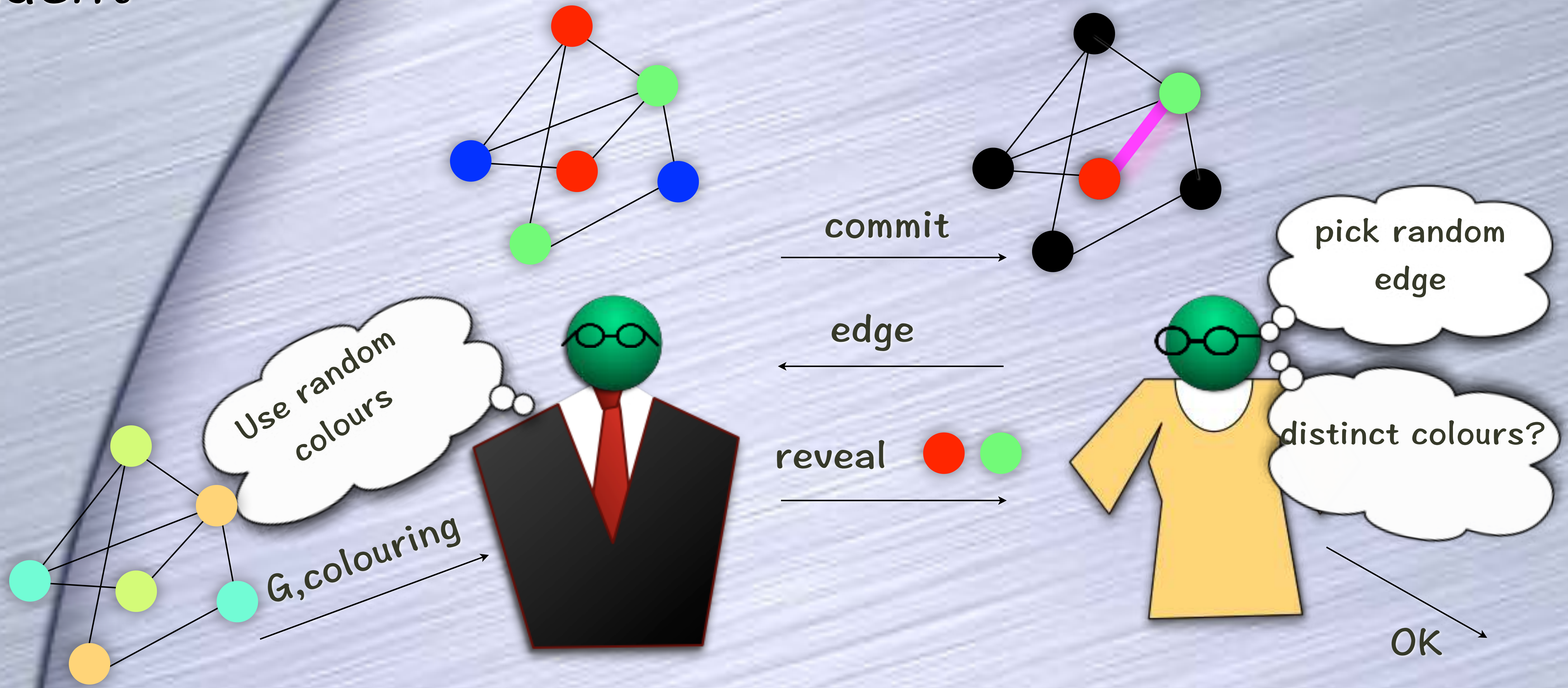
Commitment

- **Commitment** is a useful tool in many ZK proofs
  - Committing to a value: Alice puts the message in a box, locks it, and sends the locked box to Bob, who learns nothing about the message
  - Revealing a value: Alice sends the key to Bob. At this point she can't influence the message that Bob will get on opening the box.
- Implementation in the Random Oracle Model: Commit(x) = H(x,r) where r is a long enough random string, and H is a random hash function (available as an oracle) with a long enough output. To reveal, send (x,r).
  - ⚠️ Recall: ROM is a heuristic model: Can do provably impossible tasks in this model! Commitment protocols exist in the standard model too.

A ZK Proof for Graph Colourability



- To prove that nodes of a graph can be coloured with at most 3 colours, so that adjacent nodes have different colours
- Uses a commitment protocol as a subroutine
- At least 1/#edges probability of catching a wrong proof
- Repeat many times with independent colour permutations
- Graph 3-colourability is an NP-complete problem
  - A ZK proof system for any NP language L:
    $x \in L$ iff $G_x \in 3COL$
    So prove $G_x \in 3COL$ instead



Proof of Knowledge

- Proof of Knowledge: If an adversary can give valid proofs (with significant probability), then there is an efficient way to extract a witness from that adversary
- A ZK Proof of knowledge of discrete log of $Y=g^y$
  - P→V: $R := g^r$
    V→P: x
    P→V: $s := xy + r$ (modulo order of the group)
    V checks: $g^s = Y^x R$
  - Proof of Knowledge:
    - Firstly, $g^s = Y^x R \Rightarrow s = xy+r$, where $R = g^r$
    - If after sending R, P could respond to two different challenges x_1 and x_2 as $s_1 = x_1y + r$ and $s_2 = x_2y + r$, then can solve for y
  - ZK: simulation picks s, x first and sets $R = g^s/Y^x$

Cf. a Variant of Schnorr signature, with (SK,VK)=(y,Y) :
Signature = (R,s) where

Pick $R := g^r$
Let $x = H(m||R)$
Let $s := xy + r$

Verification: $g^s = Y^{H(m||R)} R$