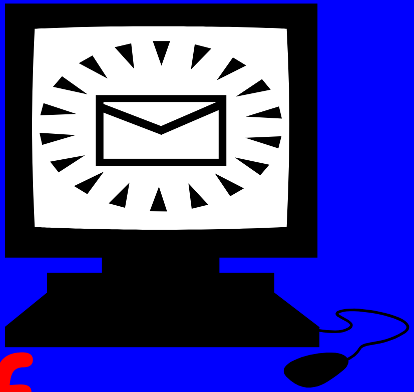
Digital Envelopes, Zero Knowledge, and other wonders of modern cryptography



(How computational complexity enables digital security & privacy)

Attribution

- These slides were prepared for the New Jersey Governor's School course "The Math Behind the Machine" taught in the summer of 2011 by Grant Schoenebeck
- Large parts of these slides were copied or modified from a presentation by Sanjeen Arora who adapted them from a presentation by the original author Avi Wigderson.

Cryptography: 1. secret writing

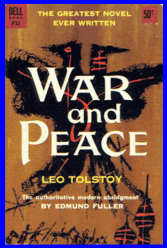
2 : the enciphering and deciphering of messages in secret code or cipher

- Ancient ideas: (pre-1976)
- Complexity-based cryptography (post-1976)

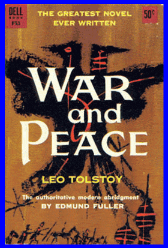
Modern crypto is about much more than just encryption or secret writing.

Cryptography pre-1976 (before computational complexity)

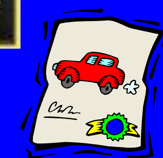
Secret communication

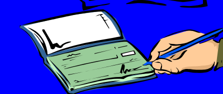


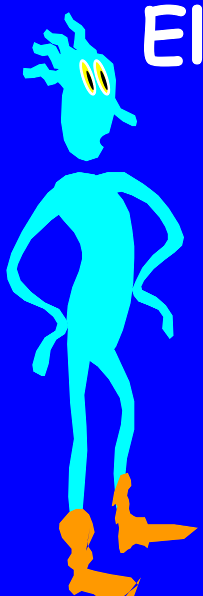
Assuming shared information which no one else has



Tasks	~~Traditional method~~
Encryption	Code books
Identification	Driver License
Money transfer	Notes, checks
Public bids	Sealed envelopes
Elections	Secret ballots





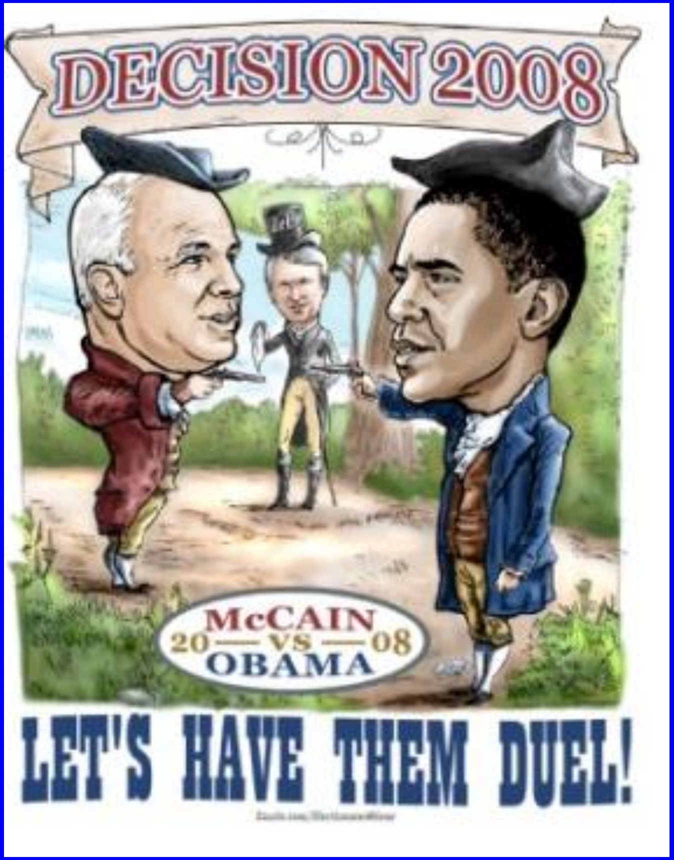


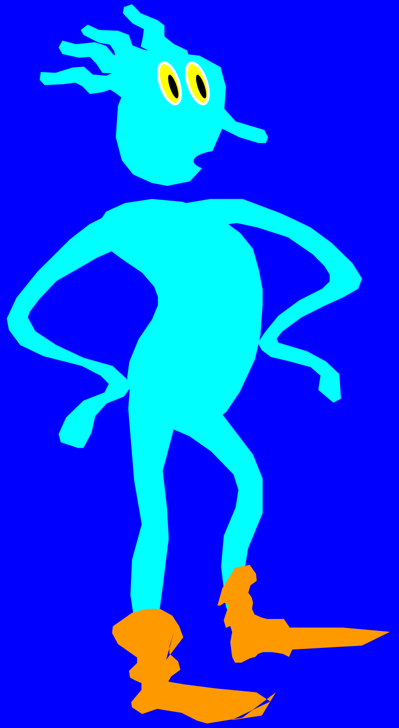
Need to be done online!

Qs. Why do you think this a problem???

Example: Public closed-ballot elections

- Hold an election in this room
  - Everyone can speak publicly (i.e. no computers, email, etc.)
  - At the end everyone must agree on who won and by what margin
  - No one should know which way anyone else voted
- Is this possible?
  - Yes! (A. Yao, Princeton)





What are we assuming here??

Axiom 1: Agents are computationally limited.

Consequence 1: Only tasks having efficient algorithms can be performed

Recall: Creating Problems can be easier than solving them

Multiplication

mult(23,67) = 1541

grade school algorithm:
n^2 steps on n digit inputs

EASY

Can be performed quickly for huge integers

Factoring

factor(1541) = (23,67)

best known algorithm:
$\exp(\sqrt{n})$ steps on n digits

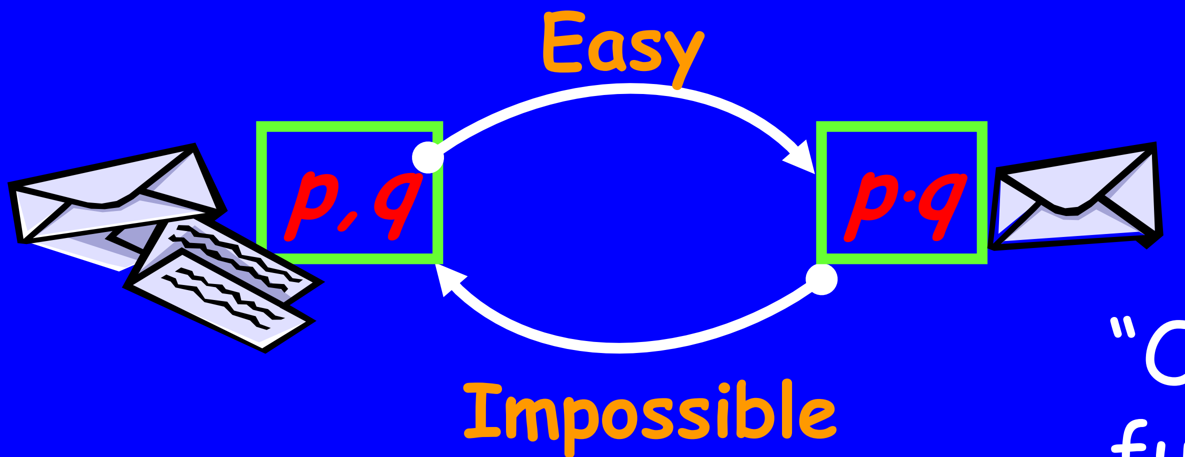
HARD?

We don't know!
We'll assume it.

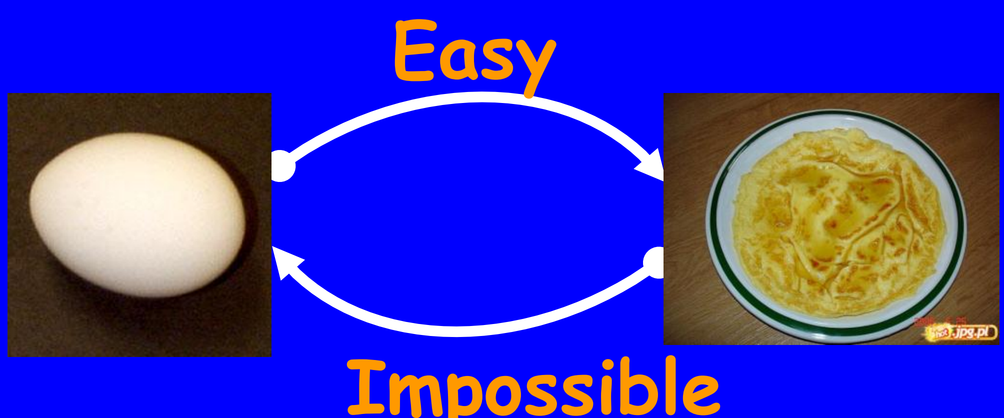
Axiom 2: Factoring is hard!

Axiom 1: Agents are computationally limited

Axiom 2: Factoring is hard

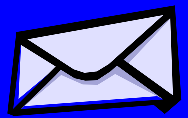


"One way function"

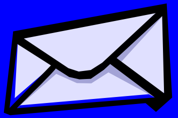


Fact: Axioms --> "Digital Envelope"

x *E(x)*

OPEN   CLOSED

- Easy to insert *x* (any value, even 1 bit)
- Hard to compute content (even partial info)
- Impossible to change content (*E(x)* defines *x*)
- Easy to verify that *x* is the content

Theorem:  $\Rightarrow$ Cryptography

El Gamal

- $P = 2Q+1$
  - E.g. $P = 227$, $Q = 113$
- Assumption: Given x computing $\log_4 x \bmod Q$ is hard.
  - Given 4^y computing y is hard.
- Alice
  - Pick a, compute 4^a, send 4^a to Bob
- Bob
  - Pick b, compute 4^b, sent 4^b to Alice
  - Compute 4^{ab}
- Alice
  - Compute 4^{ab}
- Why is this secure?

El Gamal

- Assumption: Given x computing $\log_4 x \bmod Q$ is hard.
  - Given 4^y computing y is hard.
- Alice
  - Has message x
  - Send $y = 4^x$ to Bob
- Alice
  - Send x
- Bob
  - Check that $4^x = y$
- Why is this secure?

The power of the digital envelope

Examples of increasing difficulty

Mind games of the 1980's – before Internet & E-commerce were imagined

Example: Public bid (players in one room)

Phase 1: **Commit**

$E(130)$ $E(120)$ $E(150)$

Phase 2: **Expose**

130 120 150

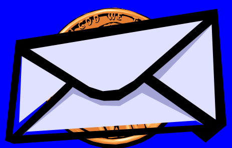
Theorem: $\Rightarrow$ Simultaneity

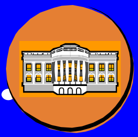
Blum
1981

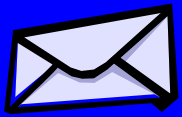
Public Lottery (on the phone)

Alice

Bob

Alice: if  I get the car (else you do)

Bob: flipping...  What did you pick?

Theorem:  $\Rightarrow$ Symmetry breaking

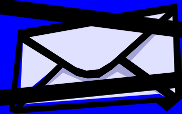
Identification / Passwords

Public password file

Name	E (pswd)
...	...
alice	$P_{alice} = E(...)$
...	...
grant	$P_{grant} = E(\text{haha})$
...	...
bob	$P_{bob} = E(...)$
...	...

Computer: 1 checks if $E(\text{pswd}) = P_{grant}$

2 erases password from screen

~~**Theorem:**  ⇒ Identification~~

Problem: Eavesdropping & repeated use!

Wishful thinking:
Computer should check if I know x such that $E(x)=P_{grant}$ **without** actually getting x

Zero-Knowledge Proof:

- Convincing
- Reveals no information

Copyrights

Dr. Alice: I can prove Riemann's Hypothesis

Prof. Bob: Impossible! What is the proof?

Dr. Alice: Lemma…Proof…Lemma…Proof...

Prof. Bob: ~~Amazing!! I'll recommend tenure~~

Amazing!! I'll publish first

Goldwasser-Micali
-Rackoff 1984

Zero-Knowledge Proof

"Claim"

Bob

Alice ("proof")

Accept/Reject

"Claim" true → Bob accepts
Bob learns nothing

"Claim" false → Bob rejects with high probability

The universality of Zero-Knowledge

Goldreich-Micali-Wigderson 1986

Theorem: Everything you can prove at all,
you can prove in Zero-Knowledge

ZK-proofs of Map Coloring

<u>Input</u>: planar map M

4-COL: is M 4-colorable?

YES!

3-COL: is M 3-colorable?

HARD!

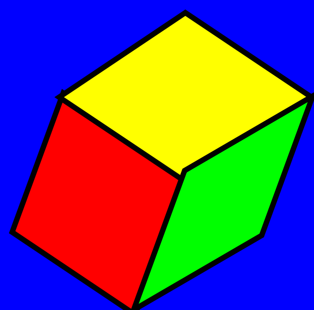
Consider “claim”: map M is 3-colorable

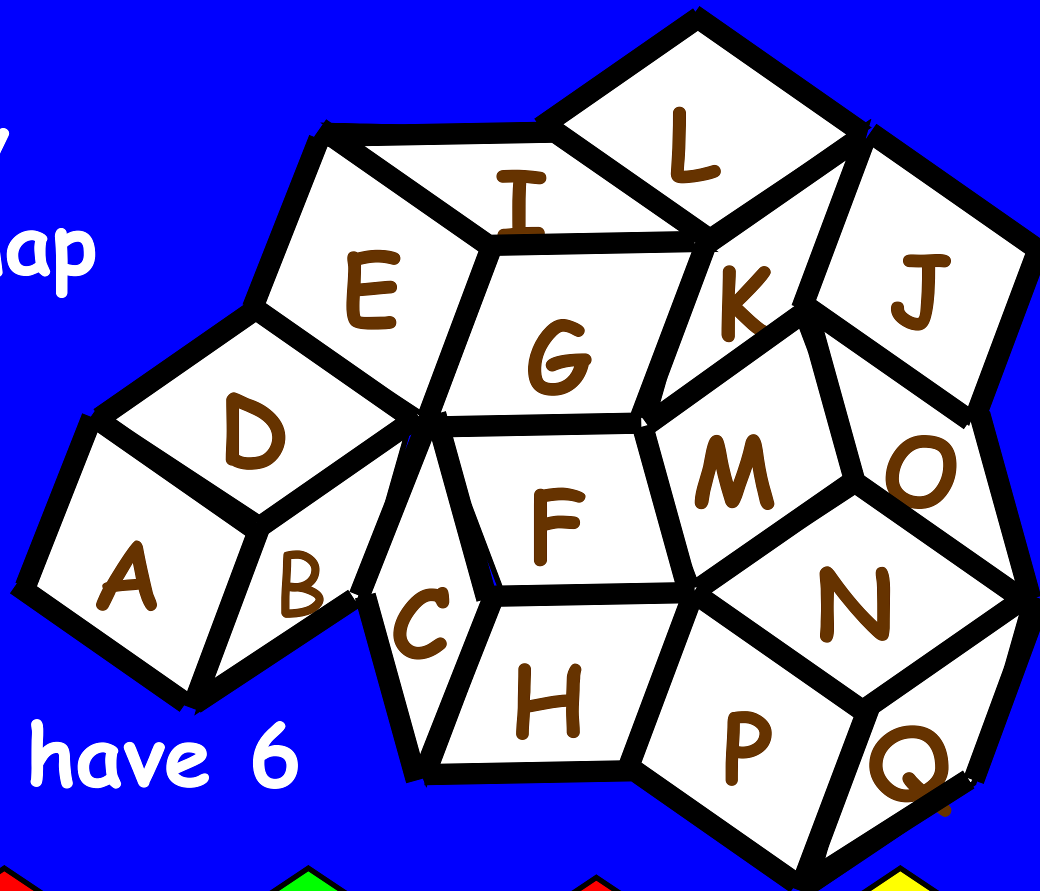
Theorem [GMW]: Such claims have ZK-proofs

I'll prove this claim in zero-knowledge

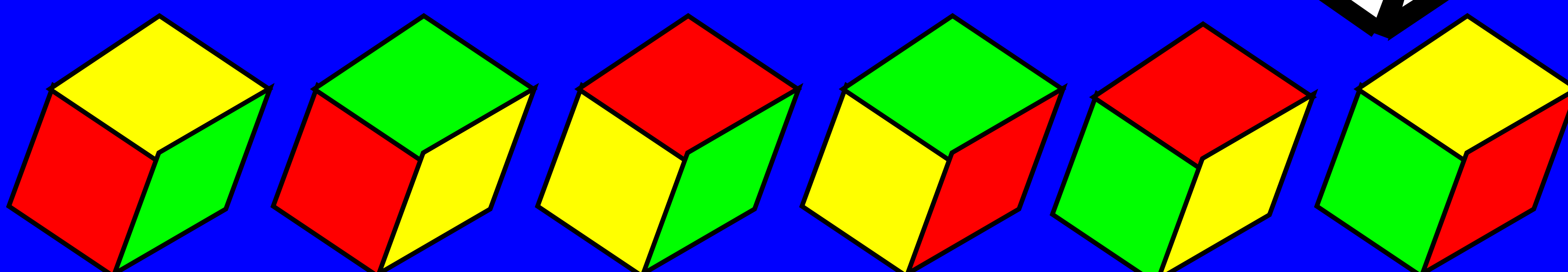
Claim: This map is 3-colorable (with R Y G)

Note: if I have any 3-coloring of any map





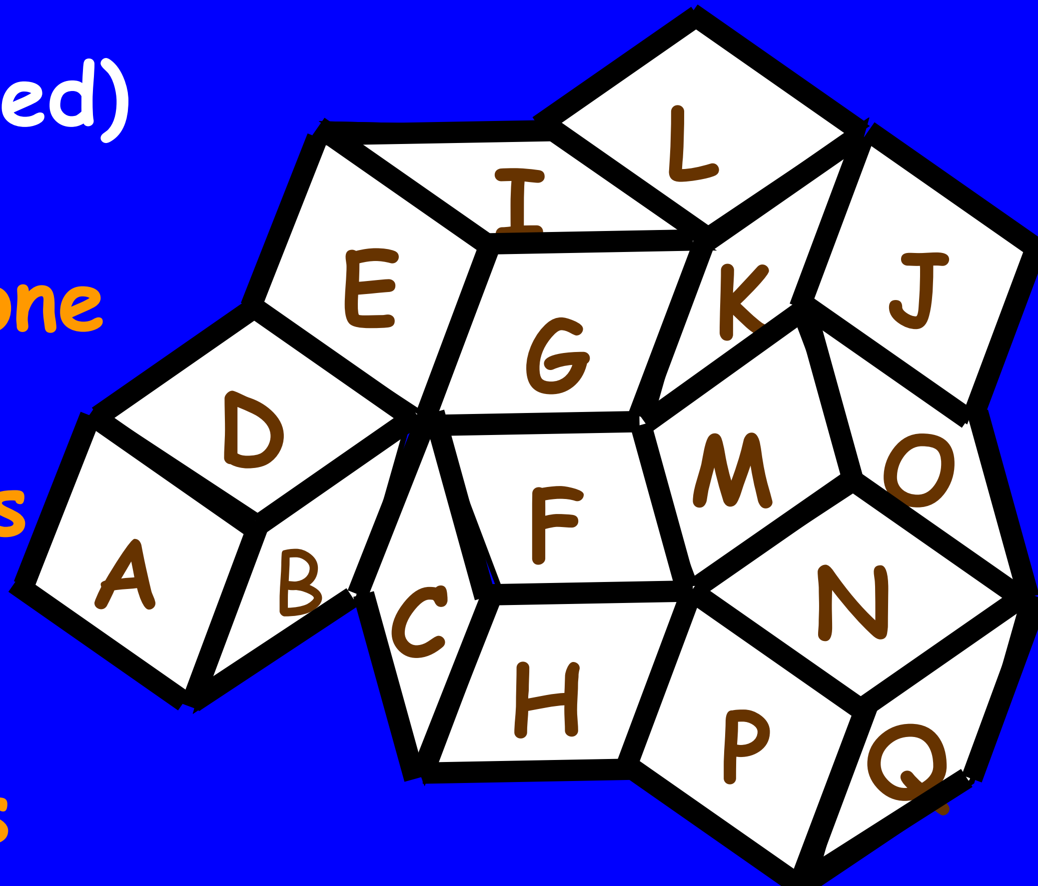
Then I immediately have 6



Structure of proof:

Repeat (until satisfied)

- I hide a random one of my 6 colorings in digital envelopes
- You pick a pair of adjacent countries
- I open this pair of envelopes



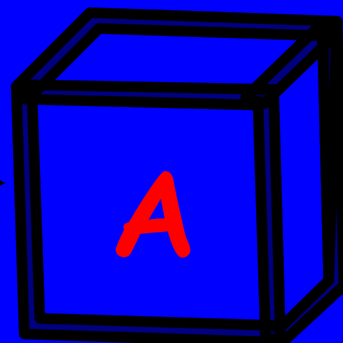
Reject if RR, YY, GG or illegal color

Why is it a Zero-Knowledge Proof?

- Exposed information is useless (Bob learns nothing)
- M 3-colorable → Probability [Accept] =1 (Alice always convinces Bob)
- M not 3-colorable→ Prob [Accept] < .99 → Prob [Accept in 300 trials] < 1/billion (Alice rarely convince Bob)

What does it have to do with Riemann's Hypothesis?

Theorem: There is an efficient algorithm *A*:

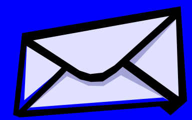
"Claim" +
"Proof length" → 

→ Map M

"Claim" true ⟷ M 3-colorable

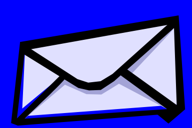
"Proof" ⟶ 3-coloring of M

"Translator" *A* comes from the proof that 3-coloring is NP-complete [Cook71, Levin73]

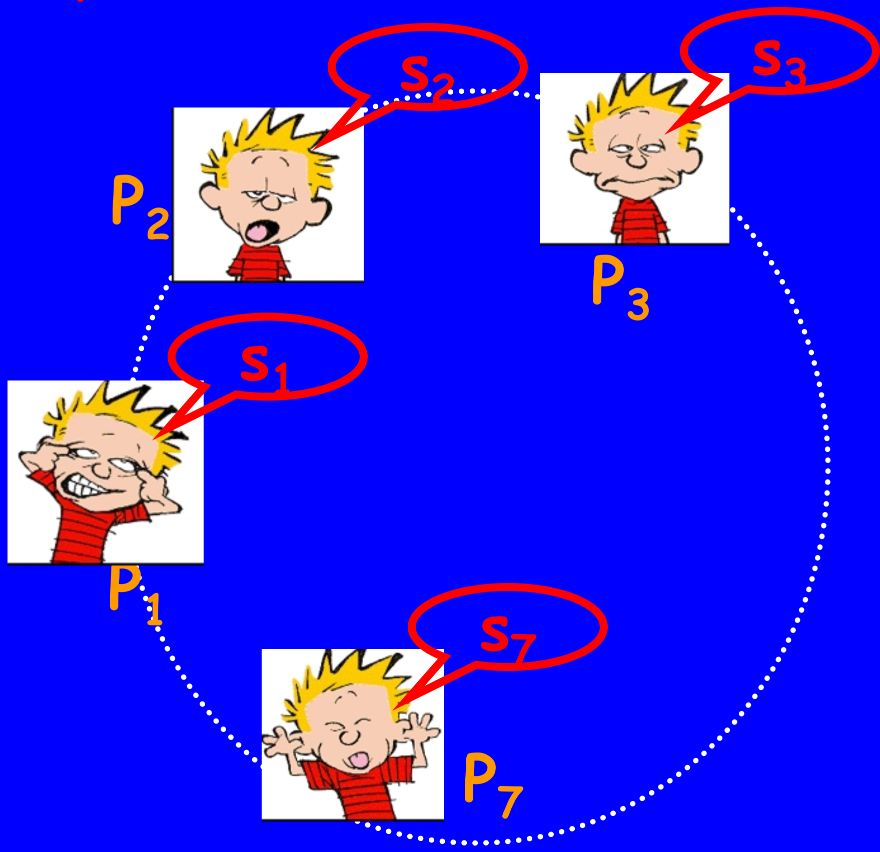
Theorem [GMW]:  + short proof

$\Rightarrow$ efficient ZK proof

$\Downarrow$

Theorem [GMW]:  $\Rightarrow$ fault-tolerant protocols

s_i secret



Making any protocol fault-tolerant

1. P_2 send $m_1(s_2)$
2. P_7 send $m_2(s_7, m_1)$
3. P_1 send $m_3(s_1, m_1, m_2)$

$\vdots \qquad \vdots$

Suppose that in step 1 P_2 sends X

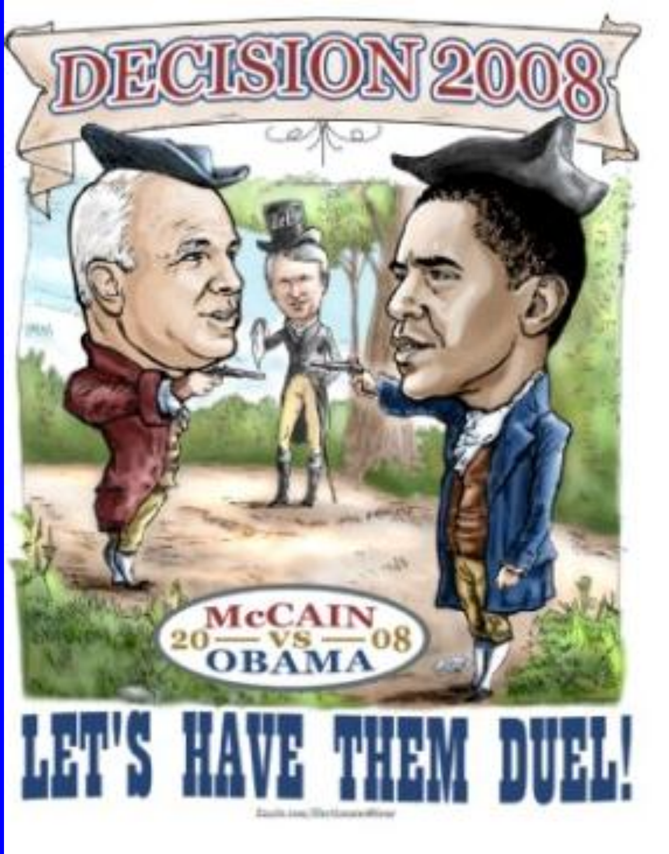
How do we know that $X = m_1(s_2)$?

s_2 is a short proof of correctness!

P_2 proves correctness in zero-knowledge!!

Recall: Public closed-ballot elections

- Hold an election in this room
  - Everyone can speak publicly (i.e. no computers, email, etc.)
  - At the end everyone must agree on who won and by what margin
  - No one should know which way anyone else voted
- Is this possible?
  - Yes! (A. Yao, Princeton)



Requires more ideas than just digital envelope

Some things we didn't have time for today

- RSA public-key cryptosystem and digital signature method
- Yao's computation-scrambling idea
- Various subtle security attacks (chosen ciphertext, chosen plaintext, etc. etc.) and how to guard against them
- Easier and speedier implementations of the zero knowledge idea using modular arithmetic....

Example: Private communication

Alice and Bob want to have a completely private conversation.

They share no private information

Solved using RSA cryptosystem (in conjunction with signature authorities like Verisign)

Summary

Practically every cryptographic task can be performed securely & privately

Assuming that players are computationally bounded and Factoring is hard.

- Computational complexity is essential!
- Hard problems can be useful!
- The theory predated (& enabled) the Internet

- What if factoring is easy (note: believed not to be NP-complete)?
- We have (very) few alternatives.

Major open question: Can cryptography be based on NP-complete problems ?