

Zero-knowledge Proofs

PETS' 26

Florian Tramer

Interactive Proof Systems

Language: A set of strings $L \subseteq \{0,1\}^*$

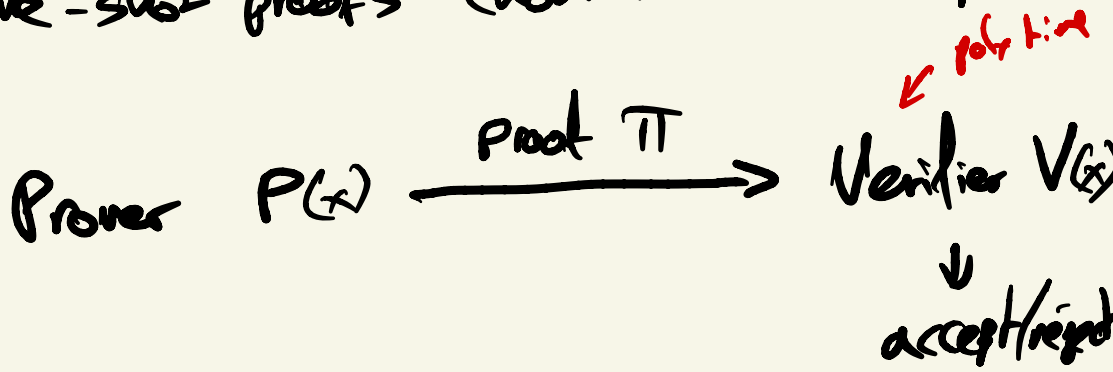
A statement: " $x \in L$ "

Ex:

" N is a product of two 1024 bit primes"

$$L = \{pq \in \{0,1\}^* \mid p, q \text{ are 1024 bit primes}\}$$

One-shot proofs (non-interactive proofs)



Properties

- Completeness: If $x \in L$, then an honest prover P (that follows the protocol) convinces an honest verifier.
- Soundness: If $x \notin L$, then any malicious prover P^* (who can deviate from the protocol) cannot convince the verifier V .

Which languages have a non-interactive proof?

NP

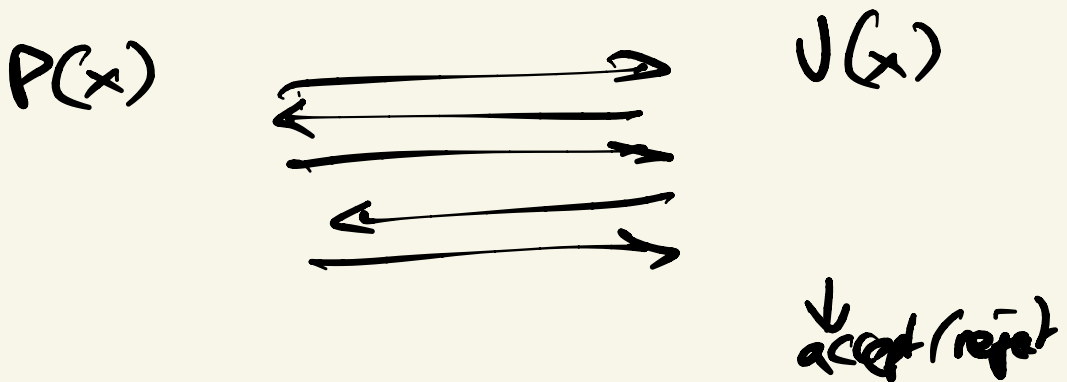
A language is in NP if there exists an efficient, deterministic algorithm M such that:

$$x \in L \Leftrightarrow \exists w \in \{0,1\}^{\text{poly}(|x|)} \text{ s.t. } M(x, w) = 1$$

The interactive proof for NP:



Interactive proofs



Why interaction?

- Prove statements not in NP !

$$IP = PSPACE$$

↑
Lang. that have
an interactive proof

↑
Languages that
can be recognised
w. polynomial
space

- Succinctness : have a proof π of size sublinear in x
- Zero knowledge : prove that x is true "without revealing w "

Definition A multi-round protocol between an unbounded prover and an efficient randomized verifier V .

Given a statement x , denote

$\langle P, V \rangle(x) \in \{0, 1\}$ an execution of the protocol where V accepts/rejects

Properties

• Completeness: $\forall x \in L$,

$$\Pr[\langle P, V \rangle(x) = 1] \geq 2/3$$

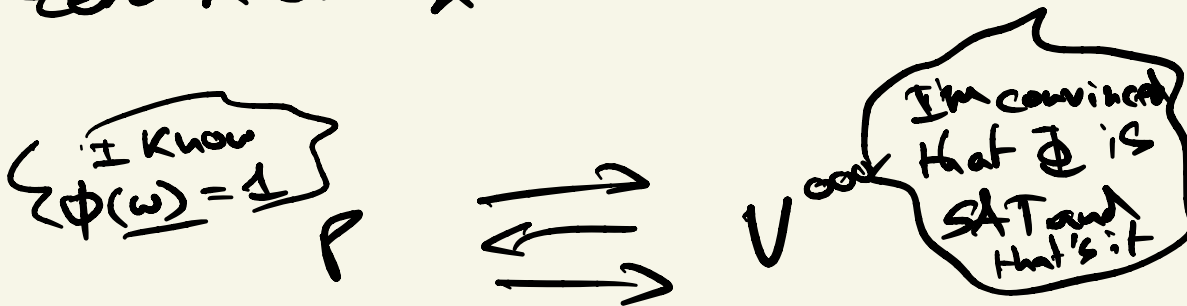
• Soundness: $\forall x \notin L$, for any malicious prover P^* ,

$$\Pr[\langle P^*, V \rangle(x) = 1] \leq 1/3$$

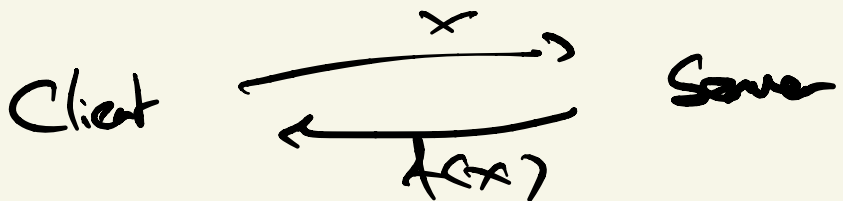
Why $2/3$ and $1/3$? Arbitrary
Amplitude Compl. & Soundness through
repetition

with λ repetitions, we can get
 $1-\lambda$ completeness & λ soundness

Zero knowledge



Verifiable computation



FHE: The server doesn't learn x

IP: The client is convinced that $h(x)$ is correct

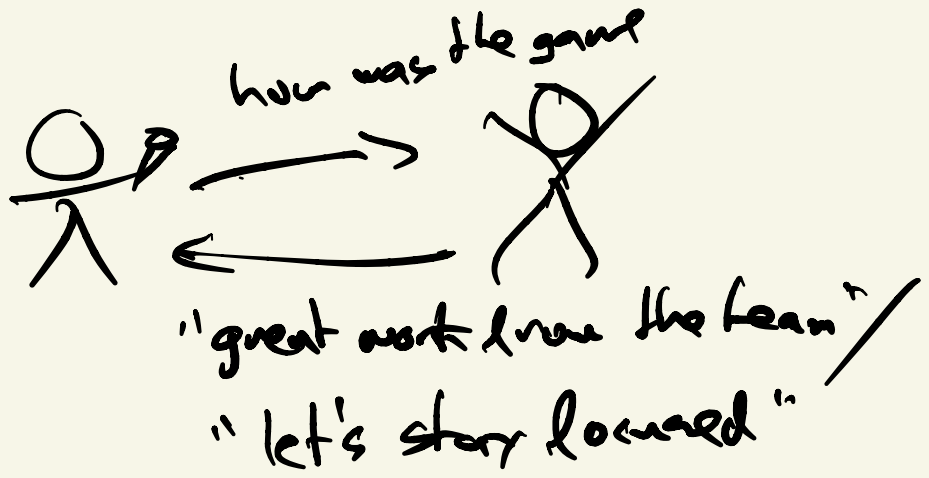
ZKP: $C \xrightarrow{x_s} r = f(x_s, x_s) \xleftarrow{r}$

The client learns nothing
about x_s

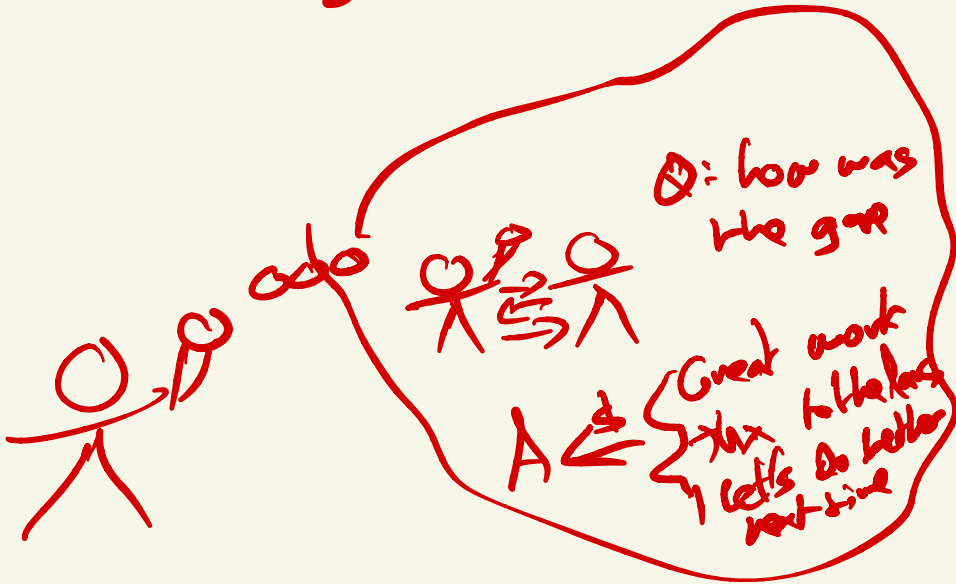
Defining ZK

What does it mean to learn "nothing"
from an interaction?

The footballer interview



Why don't we just simulate this



Defining zk

- We're going to define a Simulator
(an efficient algorithm, just used
for the def. / proof)

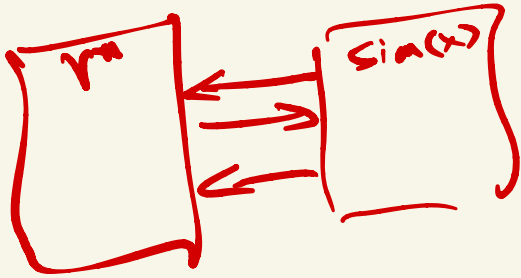
Input: everything the verifier
gets (just x)

Output: a transcript of $\langle P, V \rangle$
That is indist. from a
real one

Let $\text{View}[\langle P, V \rangle(x)]$ is everything
the verifier sees (all inputs, messages
etc)

Def. A proof system for L is zk
iff: $\exists$ a simulator Sim , s.t
 $\forall$ (possibly malicious) verifiers V^*
and $\forall x \in L$:

$$\text{View}[\langle P, V^* \rangle(x)] \approx^c \text{Sim}^{\text{V}^*}(x)$$



Why isn't the simulator just a prover?

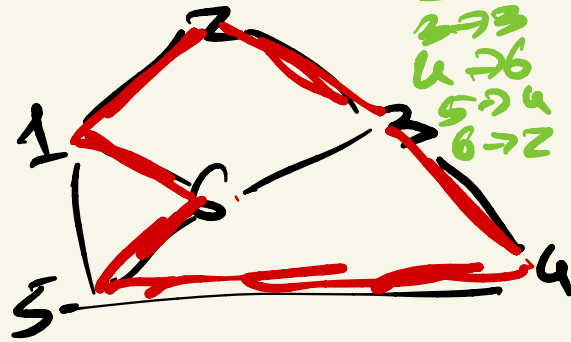
△ The simulator doesn't need to obey message ordering!

ZK proofs for any NP language

How: A proof for a NP-complete problem (Graph Hamiltonicity)

Def: Graph Hamiltonicity

Given a graph $G = (V, E)$



$B_{1,2} = 0$
 $B_{1,5} = 1$
 $B_{1,3} = 1$



The graph has a H. cycle if
it is a cycle that visits each vertex
once

The language: $L = \{ G \mid G \text{ has a H. cycle} \}$

Protocol

$P(G, \text{cycle})$

$V(G)$

pick a random permutation π
of the n vertices

let $B \in \{0,1\}^{n \times n}$ is

the adjacency matrix

of $\pi(G)$:

$B_{ij} = 1$ if $(\pi(i), \pi(j)) \in E$

$\text{Commit}(B_{11}), \dots, \text{Commit}(B_{nn})$
 $\xrightarrow{\text{Commit}(\pi)}$

c

$c \leftarrow \{0,1\}$

if $c=0$:

show the Graph
(open all commitments)

if $c=1$:

show the cycle
(open all commit.
of the edges
on the cycle)

openings

verify commitments
if $c=0$: check
graph isomorphic
to original
if $c=1$: check the
cycle

Then: if the commit. scheme is perfectly binding and comp. hiding then this protocol is a zk proof with

- perfect completeness
- soundness error $1/2$
- computational zk

- By running the protocol λ times we get soundness error $\text{negl}(\lambda)$

Proof:

- Completeness: If the prover knows a cycle, then it always opens correct commit. and V accepts
- Soundness: If the graph does not have a H. cycle the prover has to commit to either:
 - the original graph
 - a graph w. a cycle

they get "caught" w. prob $1/2$!

△ this holds because the commitment scheme is perfectly binding

- Zero knowledge

View(V^*)

- $n^2 + 1$ commitments
- c
- openings of either the graph or the cycle consistent with c

$\text{Sim}^{V^*}(G)$

Guess $\hat{c} \leftarrow \{0, 1\}$

if $\hat{c} = 0$: commit to $\Pi(G)$

if $\hat{c} = 1$: commit to a cycle graph on n edges

send the commit to V^* who returns c

if $c \neq \hat{c}$ abort and restart

open the commitments consistent with c

Output $(G, \text{commit}, c, \text{ops})$

$\text{View}(V) \quad \approx^c \quad \text{Sim}(G)$

"real commitments" $\approx^c$ "simulated commitments"

G by the hiding property
of the comm. scheme

real challenge $c \equiv$ simulated challenge

real openings $\equiv$ simulated openings

Is $\text{Sim}^{VA}(G)$ efficient?

$$\Pr[\hat{c} \neq c] \leq 1/2 + \underbrace{\text{negl}(\lambda)}$$

hiding of the
commitments

A note on amplifying soundness

if we add $2k$ and repeat the protocol
 λ times, we have to be careful
that the simulator is still efficient

Proofs of Knowledge

What does a zk proof ^{← sender} guarantee

if $V(x)$ accepts, then $x \in L$ w.h.p
it doesn't imply that the prover knows
a witness

Ex: identification protocol

Client

Server

key: α I prove I know α $X = g^\alpha$

What's the language?

$$L = \{ X \in G : \exists \alpha \text{ s.t. } X = g^\alpha \}$$

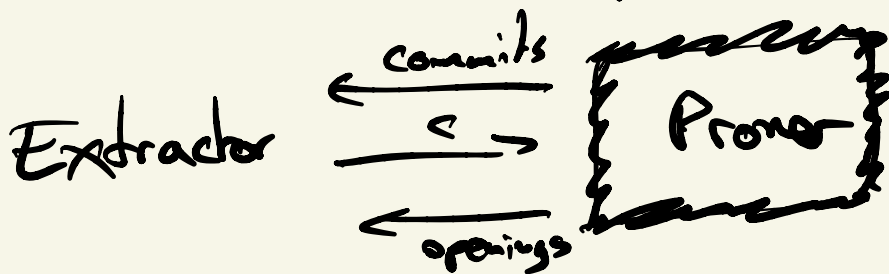
Here's a zk proof for L :

given $x \in G$, output "yes"

How do we prove that you know something? And what does that mean if you is an algorithm?

Knowledge = I can "torture" the value out of you

How do we torture the prover?



At any point the extractor can rewind the prover to an earlier state

Def Proof of Knowledge

A proof system $\langle P, V \rangle$ for a NP language L with knowledge error ϵ is a Pok

if $\exists$ an efficient extractor E s.t for all x and all P^*

$$\Pr[u(x, w) = 1 : w \leftarrow \text{Ext}^{P^*}(x)] \geq \Pr[\langle P, V \rangle(x) = 1] - \epsilon$$

 proof of knowledge subsumes soundness

Our zk proof for Graph H. is a Pk

(Internal) proof:

* Assume P^* convinces V w.p. 100%

Ext: run P^* to get commits

send $c=0$ to P^*

and get Π (w.p. 100%)

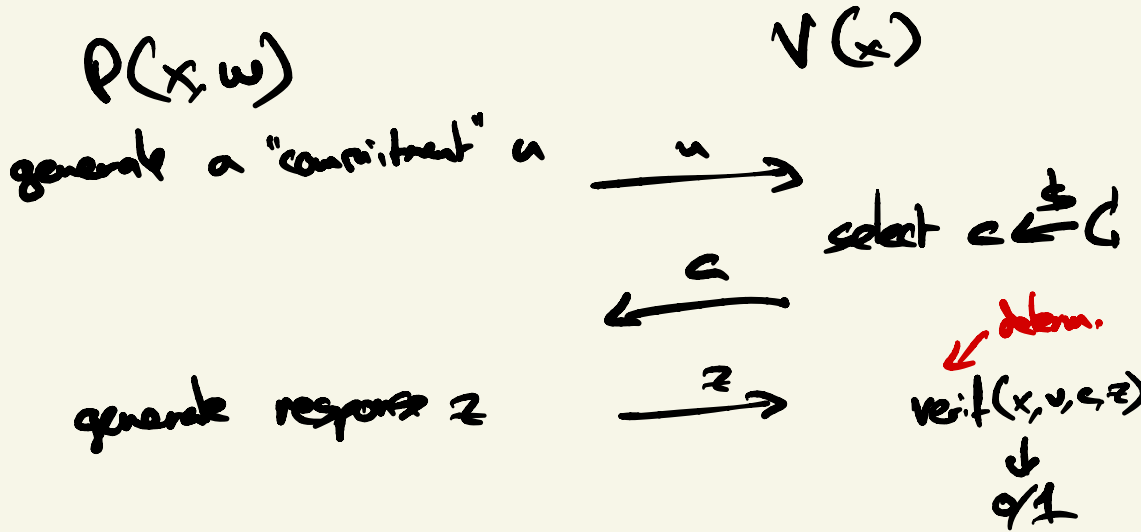
Rewind

send $c=1$ to P^*

and get cycle in $\Pi(G)$
(w.p. 100%)

Given both Π and the
cycle in $\Pi(G)$ you
can recover the cycle
in G

Sigma Protocol



Def (Sigma Protocol) is a 3-round protocol $\langle P, V \rangle$ as above

- Perfect completeness
- Honest verifier zk: $\exists \text{Sim}$
s.t. $\text{Sim}(x, c)$ for $x \in \mathcal{L}$, $c \in \mathcal{C}$
outputs (x, u, c, z) that is comp.
indist. from the verifier's view
in an honest execution

- "Special" soundness: If an efficient algorithm Ext st given two accepting transcripts (u, c, z) and (u, c', z') where $c \neq c'$, it outputs a witness w st $M(x, w) = 1$

Thm: A Sigma protocol has soundness error at most $1/|C|$

Proof: if $x \notin L$, then for any commitment u there exists at most one challenge $c \in C$ that will cause the verifier to accept
 (otherwise, the extractor could recover w but not exist)
 so the probability that the prover convinces V is $\leq \frac{1}{|C|}$

Composition: Because we only need
HV zk composing Σ
protocols preserve all
properties (see H/W)

Non interactive zk

$$P \xrightarrow{\pi} V$$

this seems impossible if we want zk.

zk implies $\exists$ a simulator that can simulate
a proof given just $x \in L$

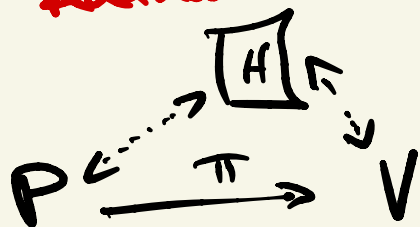
By soundness, $\text{Sim}(x)$ where $x \notin L$ should
produce a proof that the verifier rejects

$\hookrightarrow V(\text{Sim}(x))$ decides the language

$\hookrightarrow L \in \text{BPP} \neq P$

We can get around this impossibility
by changing our model of computation

Random Oracles are back!

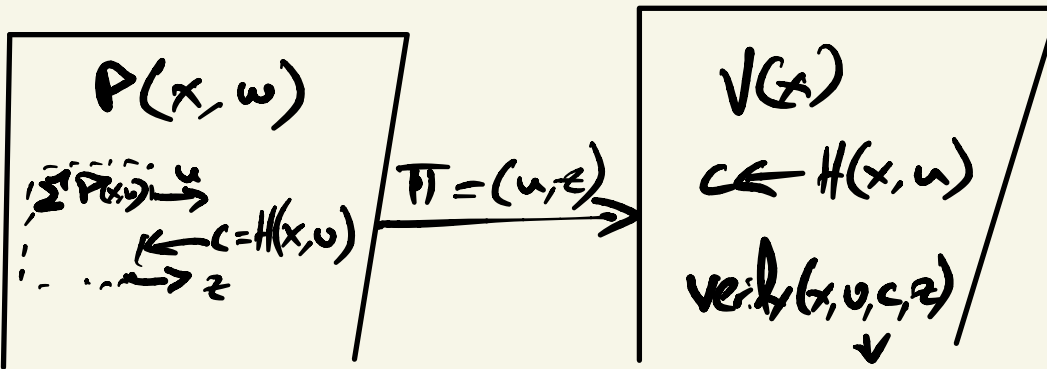


The extra power given to the simulator
is that they can "program" the RO

Result: NIZK for NP!

Fiat-Shamir Transform

Idea: Prove "simulates" a Σ protocol
and uses the RO to produce the decommitment.



Notes: for a NIZK, we can't "repeat" the protocol (there is no protocol) you need Σ to have negl. soundness error

Proof that FS gives a NIZK:

- Completeness: by design

- ZK: The simulator picks a random challenge $c \in \mathcal{C}$ and runs the simulator from the Σ protocol on (x, c) to get (u, c, z) then it programs the RO so

$$H(x, u) = c$$

and outputs $\pi = (u, z)$

(Aside: a hash function is relation intractable if for any ^{guess} relation $R(\text{inp}, \text{out})$ it's hard to find $\text{out} = H(\text{inp})$ s.t. R holds

- Pok: Ext runs the prover until the prover calls the RO on (x, u) and get $c \leftarrow \mathcal{R}$ and outputs (u, c, ϵ) then rewind the prover to its RO call, and return $c' \leftarrow \mathcal{R}$ ($c' \neq c$) and outputs (u, c', ϵ') . Then call the Σ -protocol ext.
-

Fiat-Shamir "heuristic"

Hope: if we replace RO by a hash function, say SHA-3, everything is fine ☺