Introduction to Zero-Knowledge

Vitaly Shmatikov

Commitment

- Temporarily hide a value, but ensure that it cannot be changed later
  - Example: sealed bid at an auction
- 1st stage: commit
  - Sender electronically “locks” a message in a box and sends the box to the Receiver
- 2nd stage: reveal
  - Sender proves to the Receiver that a certain message is contained in the box

Properties of Commitment Schemes

- Commitment must be hiding
  - At the end of the 1st stage, no adversarial receiver learns information about the committed value
  - If receiver is probabilistic polynomial-time, then <u>computationally</u> hiding; if receiver has unlimited computational power, then <u>perfectly</u> hiding
- Commitment must be binding
  - At the end of the 2nd stage, there is only one value that an adversarial sender can successfully "reveal"
  - Perfectly binding vs. computationally binding
- Can a scheme be perfectly hiding and binding?

Discrete Logarithm Problem

- Intuitively: given $g^x \bmod p$ where p is a large prime, it is "difficult" to learn x
  - Difficult = there is no known polynomial-time algorithm
- g is a generator of a multiplicative group Z_p^*
  - Fermat's Little Theorem
    - For any integer a and any prime p, $a^{p-1}=1 \bmod p$.
  - $g^0, g^1 \ldots g^{p-2} \bmod p$ is a sequence of distinct numbers, in which every integer between 1 and p-1 occurs once
    - For any number $y \in [1 .. p-1]$, $\exists$ x s.t. $g^x = y \bmod p$
  - If $g^q=1$ for some $q>0$, then g is a generator of Z_q, an order-q subgroup of Z_p^*

Pedersen Commitment Scheme

- **Setup:** receiver chooses...
  - Large primes p and q such that q divides p-1
  - Generator g of the order-q subgroup of Z_p^*
  - Random secret a from Z_q
  - $h=g^a \bmod p$
    - Values p,q,g,h are public, a is secret
- **Commit:** to commit to some $x \in Z_q$, sender chooses random $r \in Z_q$ and sends $c=g^x h^r \bmod p$ to receiver
  - This is simply $g^x(g^a)^r=g^{x+ar} \bmod p$
- **Reveal:** to open the commitment, sender reveals x and r, receiver verifies that $c=g^x h^r \bmod p$

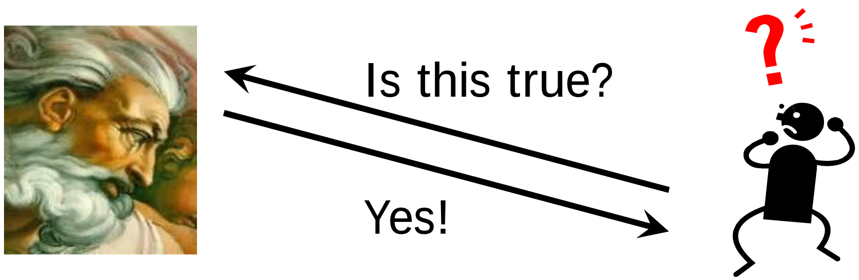
Security of Pedersen Commitments

- **Perfectly hiding**
  - Given commitment c, every value x is equally likely to be the value commited in c
  - Given x, r and any x', exists r' such that $g^x h^r = g^{x'} h^{r'}$

    $r' = (x-x')a^{-1} + r \mod q$ (but must know a to compute r')
- **Computationally binding**
  - If sender can find different x and x' both of which open commitment $c=g^x h^r$, then he can solve discrete log
    - Suppose sender knows x,r,x',r' s.t. $g^x h^r = g^{x'} h^{r'} \mod p$
    - Because $h=g^a \mod p$, this means $x+ar = x'+ar' \mod q$
    - Sender can compute a as $(x'-x)(r-r')^{-1}$
    - But this means sender computed discrete logarithm of h!

Zero-Knowledge Proofs

- An interactive proof system involves a prover and a verifier
- Idea: the prover proves a statement to the verifier without revealing anything except the fact that the statement is true
  - Zero-knowledge proof of knowledge (ZKPK): prover convinces verifier that he knows a secret without revealing the secret
- Ideal functionality ☺



Properties of ZKPK

- Completeness
  - If both prover and verifier are honest, protocol succeeds with overwhelming probability
- Soundness
  - No one who does not know the secret can convince the verifier with nonnegligible probability
    - Intuition: the protocol should not enable prover to prove a false statement
- Zero knowledge
  - The proof does not leak any information

Zero-Knowledge Property

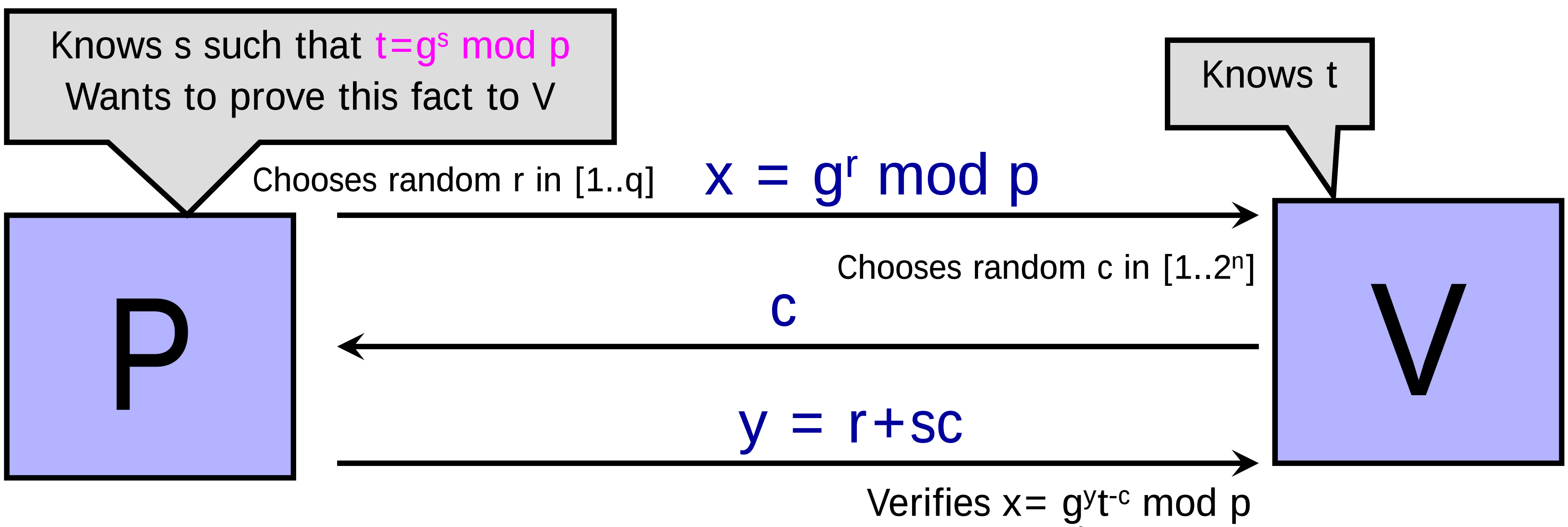
- The proof does not leak any information
- There exists a simulator that, taking what the verifier knows before the protocol starts, produces a fake "transcript" of protocol messages that is indistinguishable from actual protocol messages
  - Because all messages can be simulated from verifier's initial knowledge, verifier does not learn anything that he didn't know before
  - Indistinguishability: perfect, statistical, or computational
- Honest-verifier ZK only considers verifiers that follow the protocol

Soundness Property

- No one who does not know the secret can convince the verifier with nonnegligible probability
- Let A be any prover who convinces the verifier...
- ...there must exist a knowledge extractor algorithm that, given A, extracts the secret from A
  - Intuition: if there existed some prover A who manages to convince the verifier that he knows the secret without actually knowing it, then no algorithm could possibly extract the secret from this A

Schnorr's Id Protocol

- System parameters
  - Prime p and q such that q divides p-1
  - g is a generator of an order-q subgroup of Z_p*

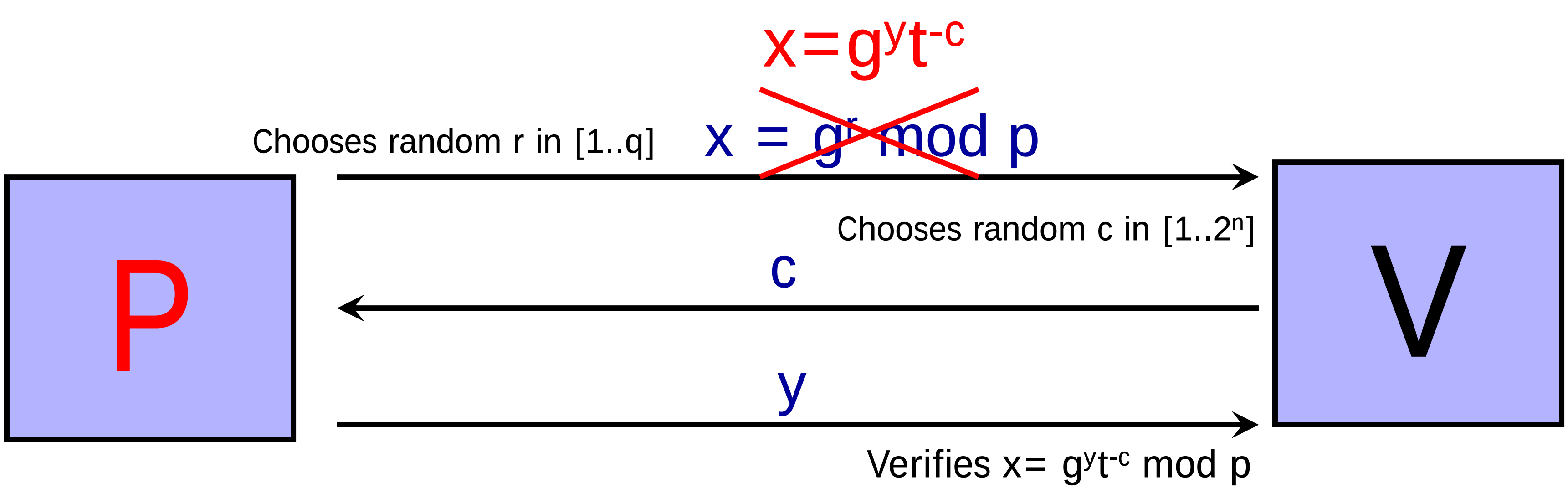


$= g^{r+sc}(g^s)^{-c}$ mod p $= g^r$ mod p

P proves that he knows discrete log of t without revealing its value

Cheating Sender

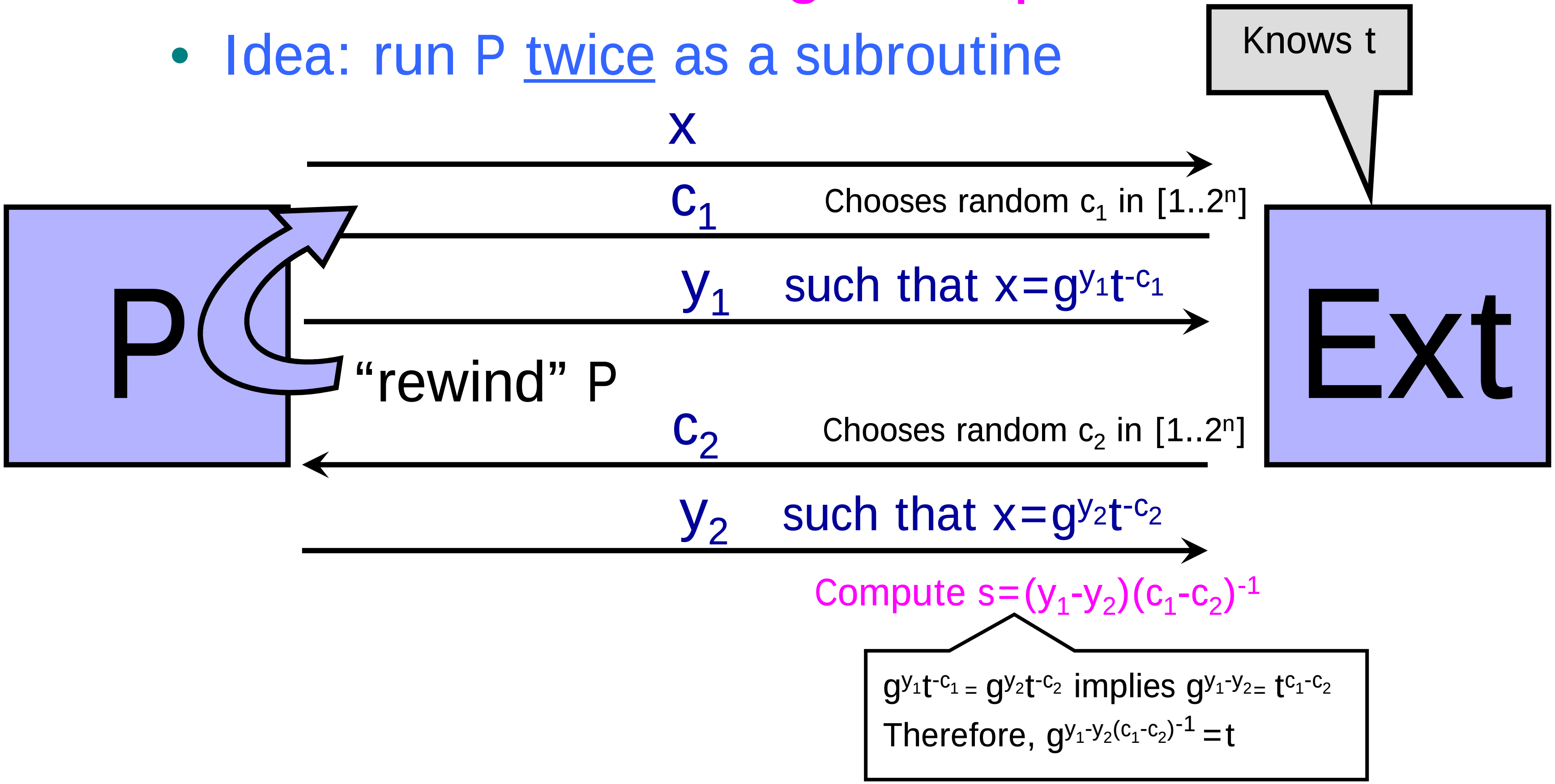
- Prover can cheat if he can guess c in advance
  - Guess c, set $x=g^y t^{-c}$ for random y in 1st message
  - What is the probability of guessing c?



P proves that he “knows” discrete log of t even though he does not know s

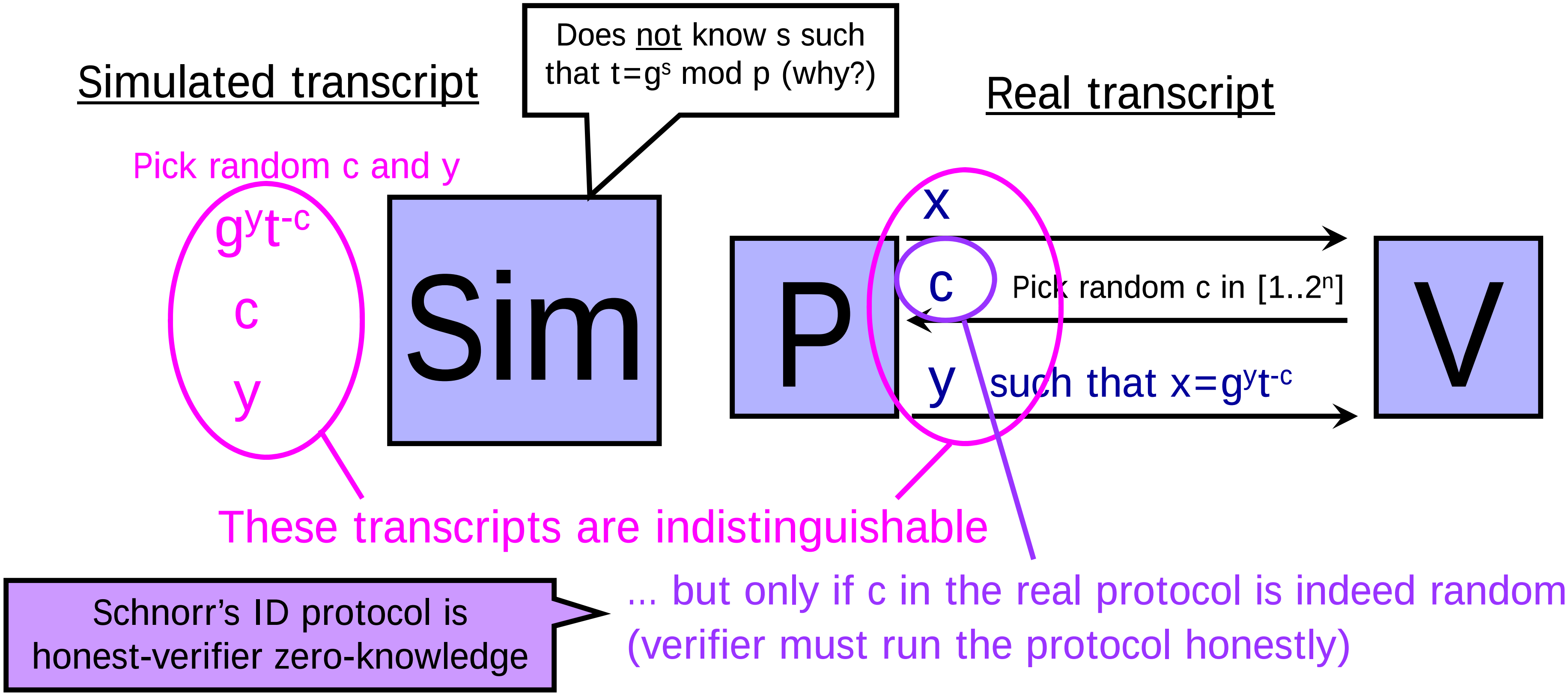
Schnorr's Id Protocol Is Sound

- Given P who successfully passes the protocol, extract s such that $t=g^s \bmod p$
  - Idea: run P <u>twice</u> as a subroutine



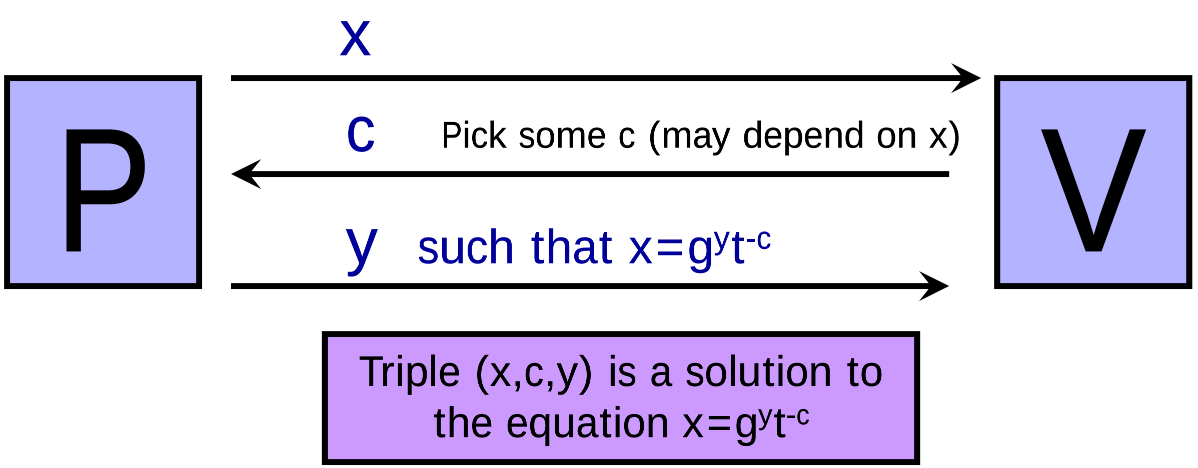
Schnorr's Id Protocol Is HVZK

- Simulator produces a transcript which is indistinguishable from the real transcript



Schnorr’s Id Protocol Is Not ZK

- Schnorr’s ID protocol is <u>not</u> zero-knowledge for malicious verifier if challenge c is large



Verifier may not be able to come up with such a triple on his own.
Therefore, he learned something from the protocol
(protocol is not zero-knowledge!)