

# Secure Multi-Party Computation With Malicious Parties

---

Vitaly Shmatikov

# 1-out-of-4 Oblivious Transfer

---

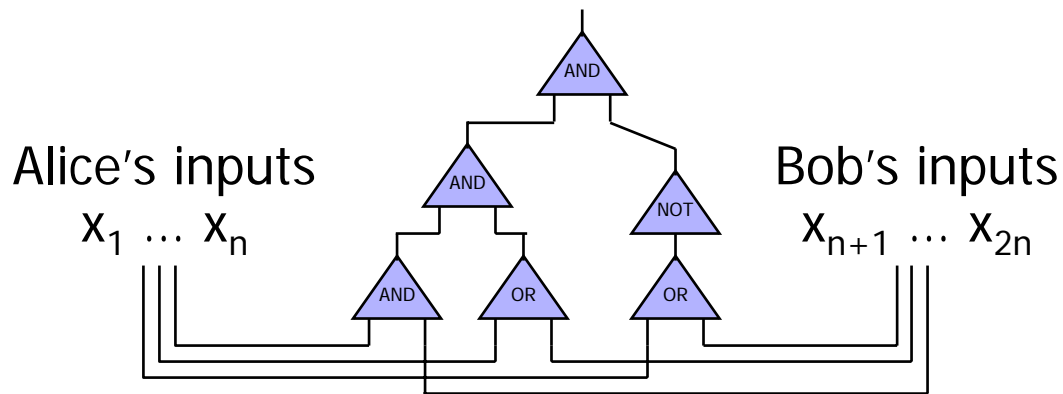


- Very similar to 1-out-of-2 oblivious transfer
- How to construct a 1-out-of-4 OT protocol given an 1-out-of-2 protocol?

# Boolean Circuits

---

- ◆ Alice and Bob want to compute function  $f(a,b)$ 
  - Assume for now Alice and Bob are semi-honest
- ◆ First, convert the function into a **boolean circuit**

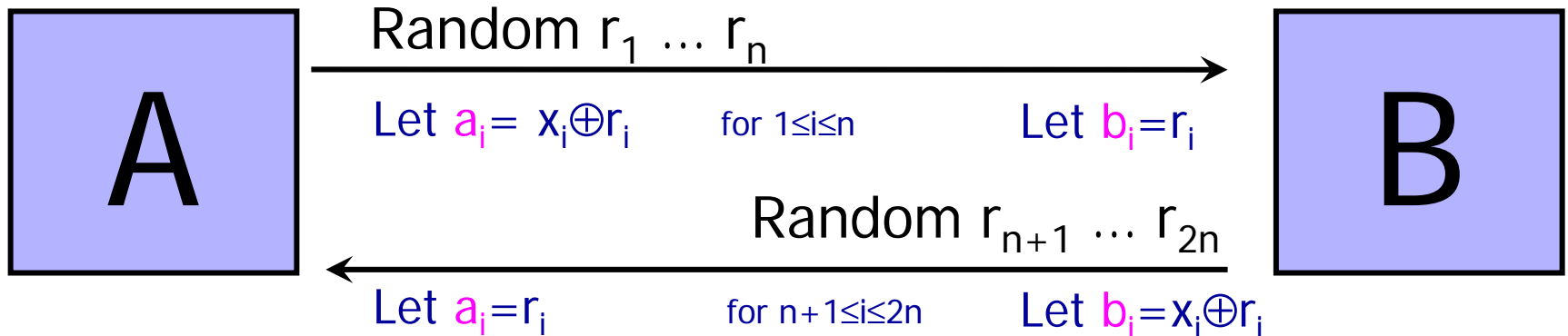


- ◆ Next, parties securely **share** their inputs

# Input Sharing

Knows  $x_1 \dots x_n$

Knows  $x_{n+1} \dots x_{2n}$

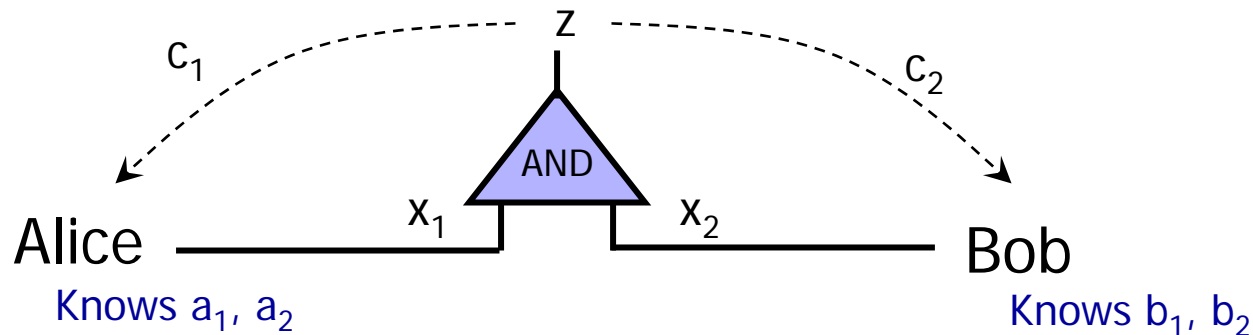


◆ After this exchange, for all inputs  $x_i \dots$

$$x_i = a_i \oplus b_i$$

- Alice still doesn't know Bob's inputs, and vice versa
- This is information-theoretically secure

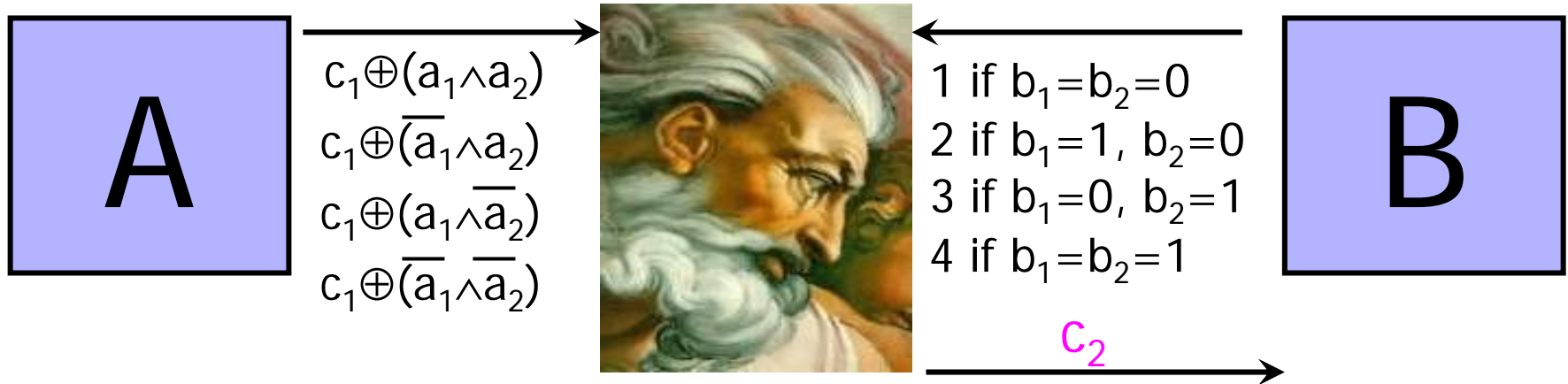
# Evaluating an AND Gate



- Inputs  $x_1$  and  $x_2$  are shared between Alice and Bob
  - $x_1 = a_1 \oplus b_1$ ,  $x_2 = a_2 \oplus b_2$
- At the end of the protocol, Alice learns bit  $c_1$  and Bob learns bit  $c_2$  such that...
  - Bits  $c_{1,2}$  are chosen at random
  - $c_1 \oplus c_2 = x_1 \wedge x_2 = (a_1 \oplus b_1) \wedge (a_2 \oplus b_2)$ 
    - Output of the gate is shared between A and B just like inputs are

# Use Oblivious Transfer

Pick random  $c_1$



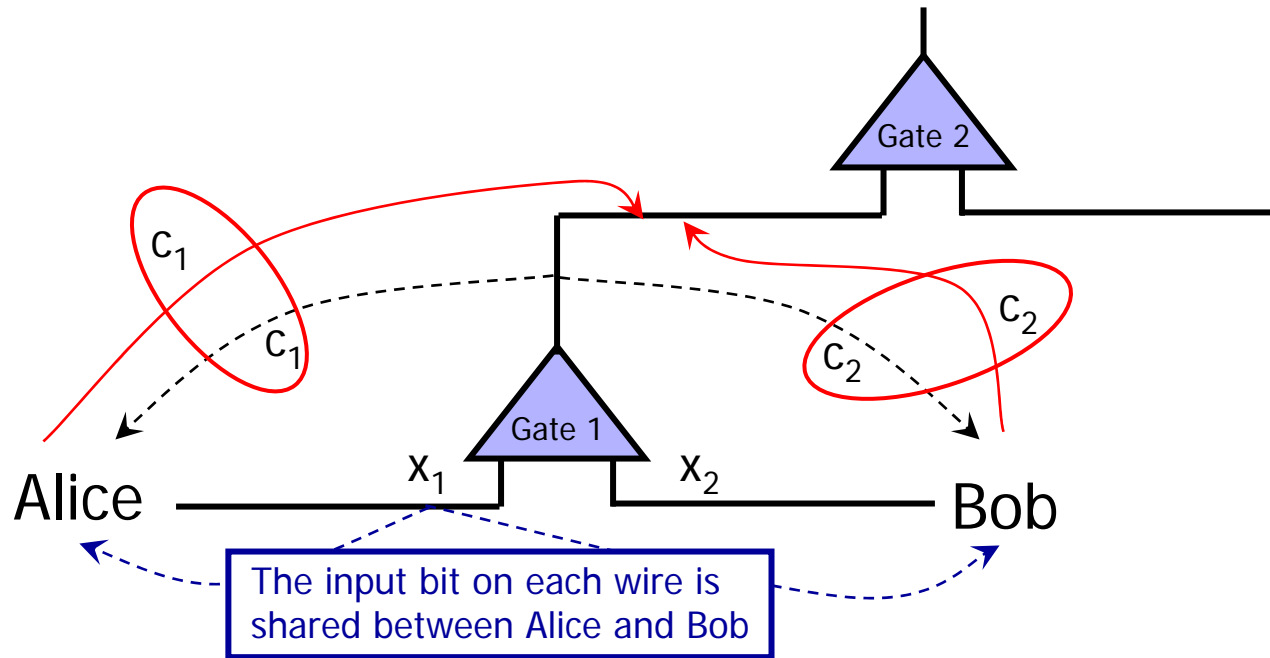
- In every case,  $c_1 \oplus c_2 = (a_1 \oplus b_1) \wedge (a_2 \oplus b_2) = x_1 \wedge x_2$
- Can use similar tricks for other gate types
- Why do I give the ideal functionality only, not the actual protocol?

# Is Secure OT Enough?

---

- ◆ Last time, we saw an oblivious transfer protocol which is secure even if parties are malicious
  - Chooser commits to his input and proves in zero knowledge that he performed his computations correctly
- ◆ Suppose each gate of the circuit is evaluated using an oblivious transfer protocol which is secure with malicious parties...
- ◆ Do we obtain a protocol for securely computing any function with malicious parties?

# Oops!



- ◆ When output of Gate 1 is used as input into Gate 2, both parties must use the same pair of bits that was obtained from evaluating Gate 1 (why?)

# Security with Malicious Parties

---

- ◆ Details omitted (this is less than a sketch!)
- ◆ Intuition: every party commits to its inputs and proves in ZK that it did its computation correctly
  - This includes proving that output of one OT protocol is consistent with input of another OT protocol
- ◆ Main advantage: secure building blocks compose into secure protocols
  - Each building block (commitment, OT, etc.) is indistinguishable from its ideal functionality (IF)
  - IFs can be composed without compromising security
  - Can build a secure protocol from secure primitives

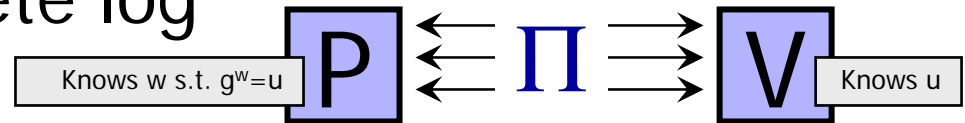
# Issues

---

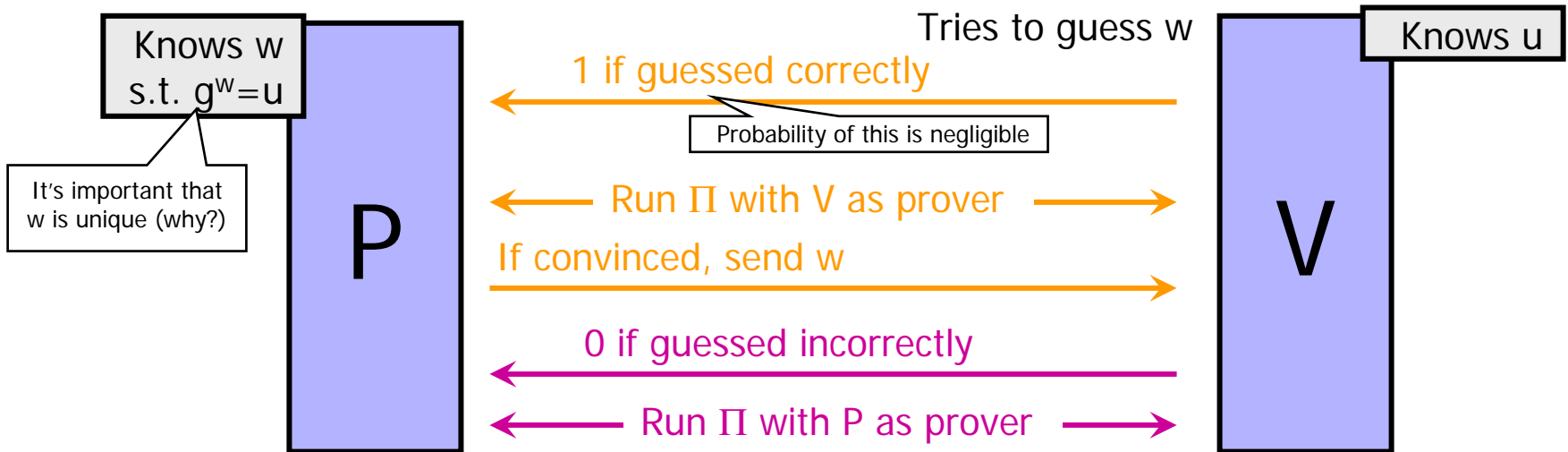
- ◆ **Parallel composition** of zero-knowledge proofs does not work in general
  - One proof is zero-knowledge, but two proofs executed concurrently are no longer zero-knowledge
- ◆ How can **cryptographic primitives** be formulated as secure multi-party problems?
  - Many protocols use encryption, digital signatures, etc.
- ◆ **Adaptive corruptions**: adversary may corrupt an honest party in the middle of protocol execution
  - Proofs with “rewinding” don’t work any more (why?)

# ZKPK of Discrete Log

- ◆ Suppose we have some ZK protocol  $\Pi$  for proving knowledge of discrete log



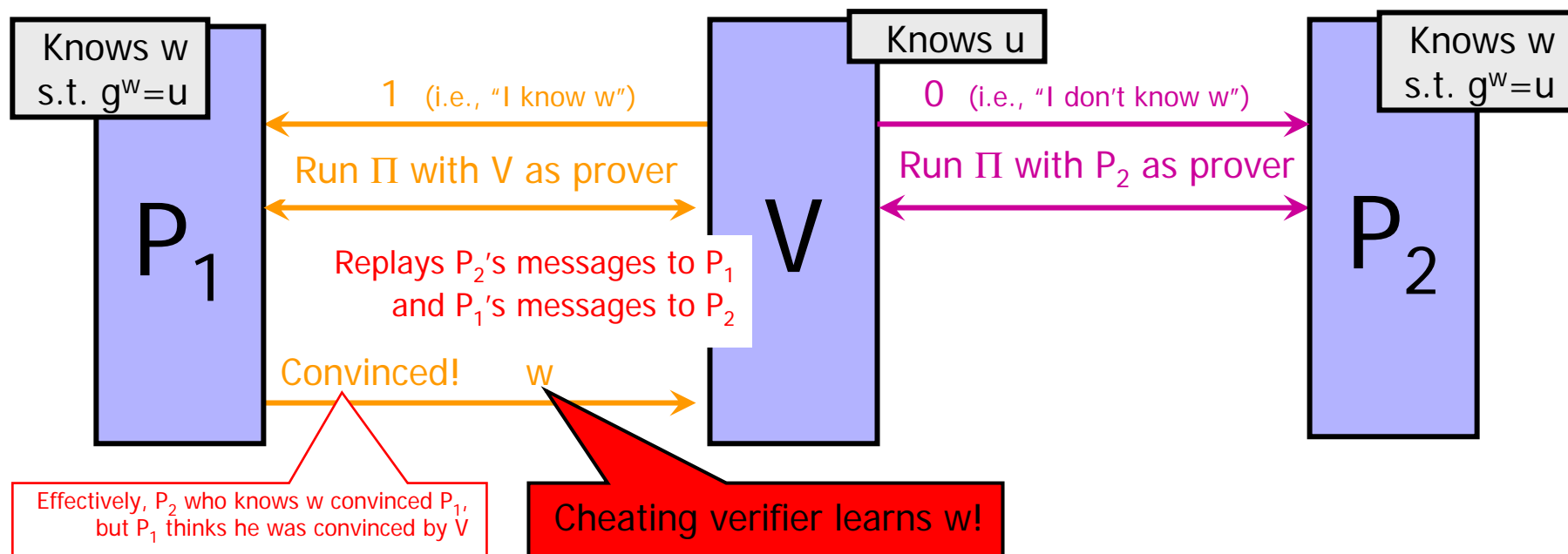
- ◆ Here is another protocol  $\Pi'$



$\Pi'$  is a sound zero-knowledge protocol of discrete-log knowledge (why?)

# Concurrent Composition

- ◆  $V$  runs two instances of protocol  $\Pi'$  in parallel



This protocol is clearly NOT zero-knowledge (why?)