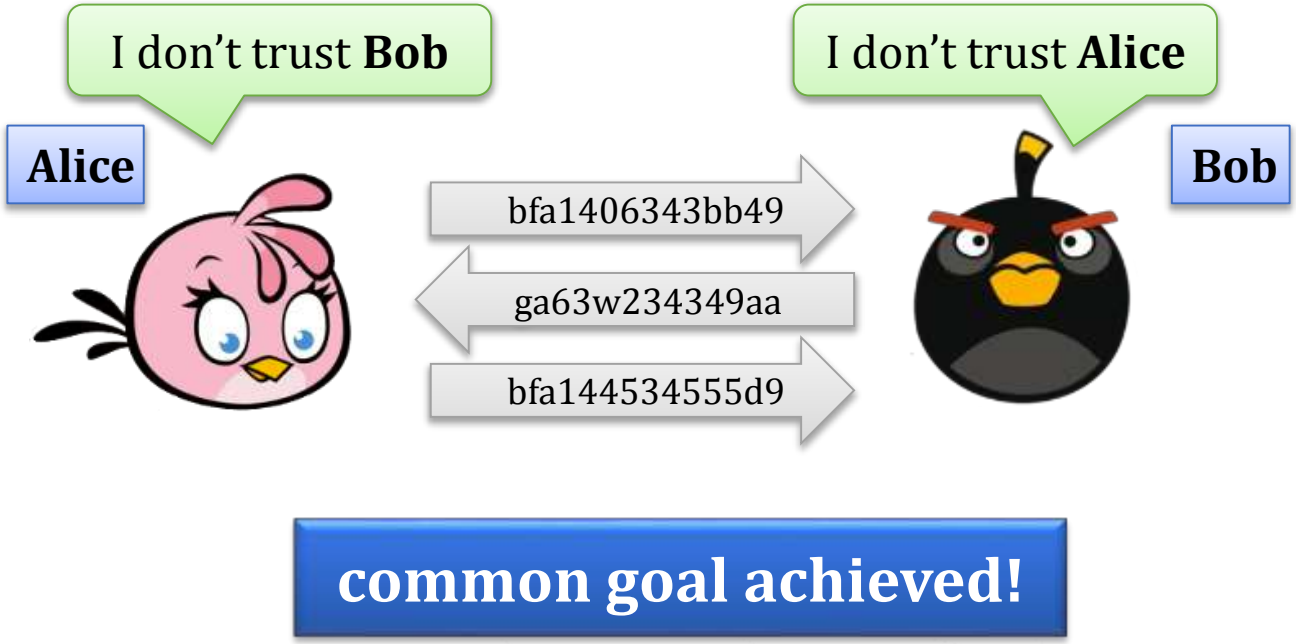
Multiparty Computation (MPC) protocols

Protocols where the **users of the protocol** don't trust each other, but nevertheless

they want to achieve a common goal



With a “trusted third party” – it’s easy

ideal world:

But can we do it **without** a trusted third party?

real world:

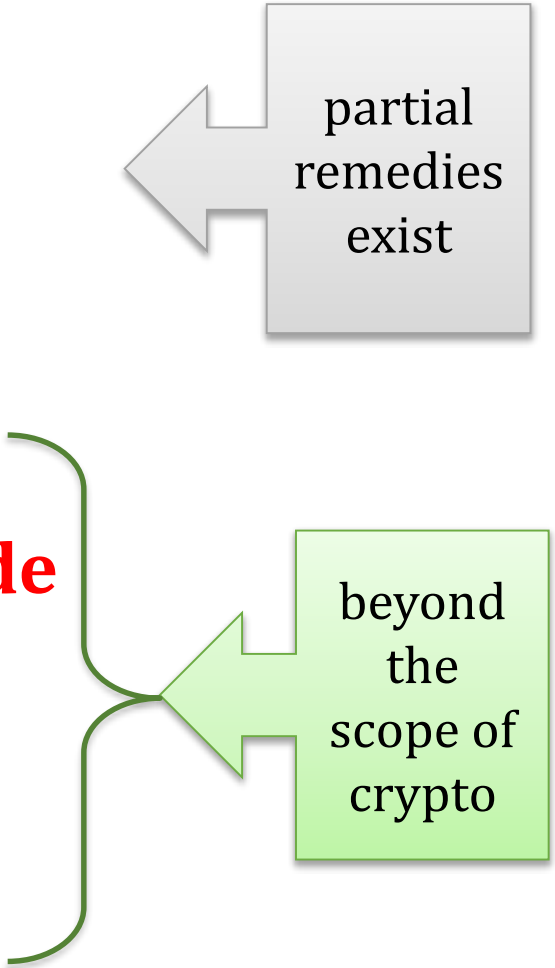
In other words: can we “simulate” the **ideal world** in the **real world**?

The limitations

- **lack of fairness** when there is no honest majority
  (we will explain it in a moment),

- no way to force the parties to **provide true input**,

- and to **respect the outcome**.

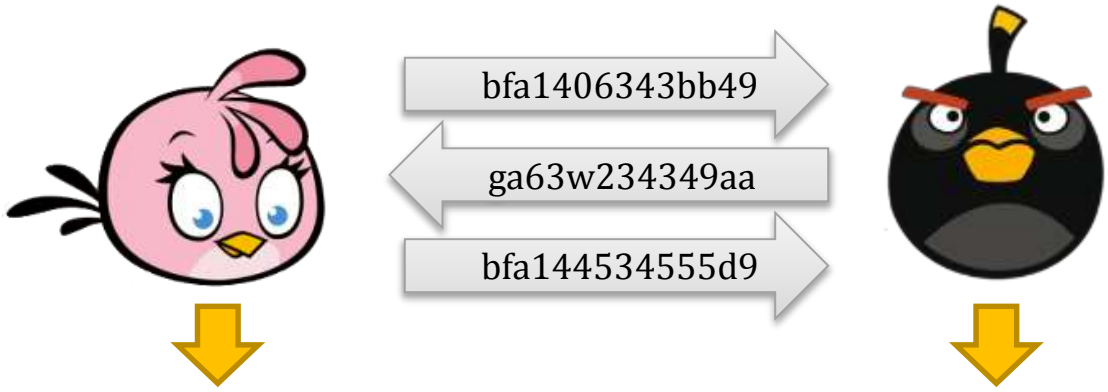


Our idea

Example: **Two party lotteries**

- **a random party earns 1 BTC**
- **the other one looses 1 BTC**

Looks similar to the “coin-tossing problem”.



output: **Y** **Y**

where Y =

 with probability **1/2**

 with probability **1/2**

How to solve the coin-tossing problem?

Idea

Remember the old game:

rock-paper-scissors?

Alice

Bob

			
	draw	Alice wins	Bob wins
	Bob wins	draw	Alice wins
	Alice wins	Bob wins	draw

Let's simplify this game

		Alice	
		A=0	**A=1**
Bob	**B=0**	**Alice wins**	**Bob wins**
	B=1	**Bob wins**	**Alice wins**

In other words: **Alice** wins iff **A xor B = 0**.

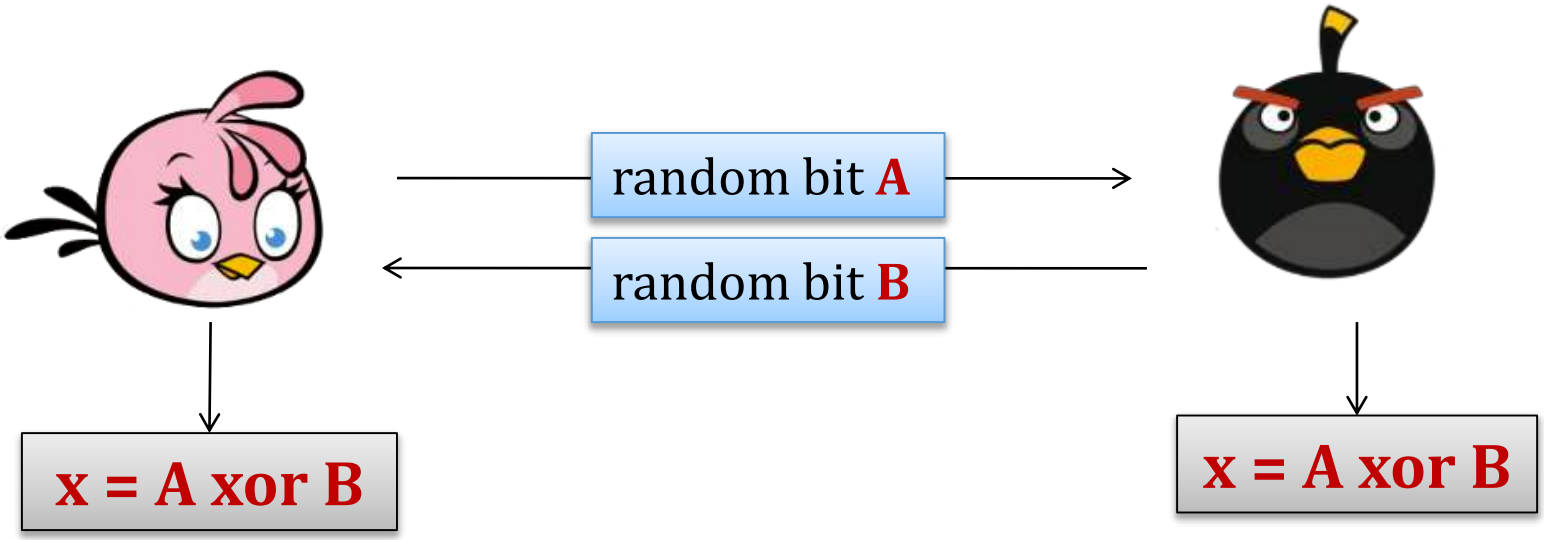
Another way to look at it

Bob
has an input A

Alice
has an input B

they should jointly compute
x = A xor B
(in a secure way)

What to do?



Problem:

A and **B** should be sent at the same time

(e.g. if **A** is sent before **B** then a malicious **Bob** can set **B := x xor A**, where **x** is chosen by him).

How to guarantee this?

Seems hard:

the internet is not synchronous...

A solution:

bit commitments

Commitment schemes – an intuition

a bit **A**

Alice "commits herself to **A**"

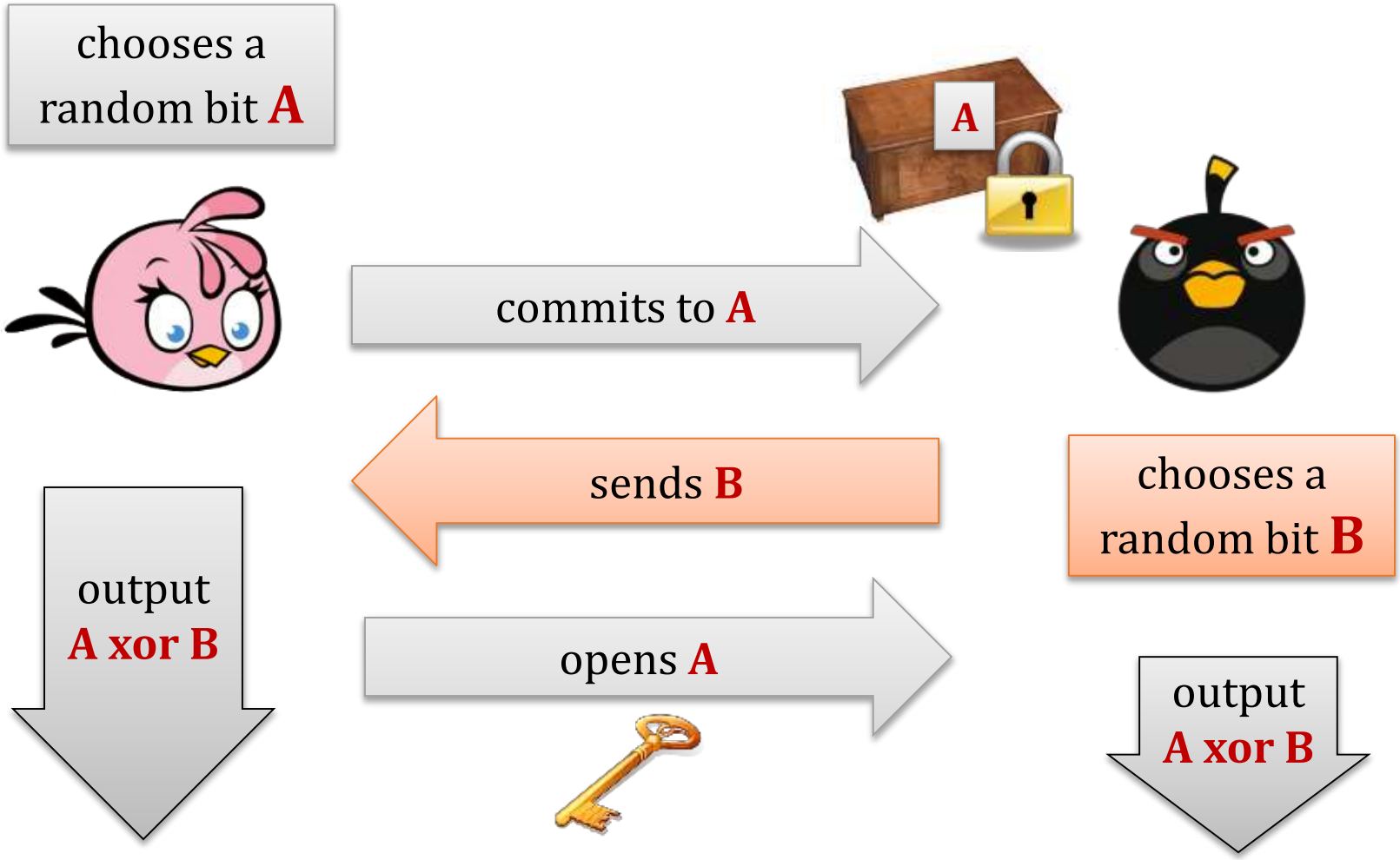
Alice sends a **locked** box to **Bob**

[binding] from now **Alice** cannot change **A**,
[hiding] but **Bob** doesn't know **A**

Alice can later send the key to **Bob**

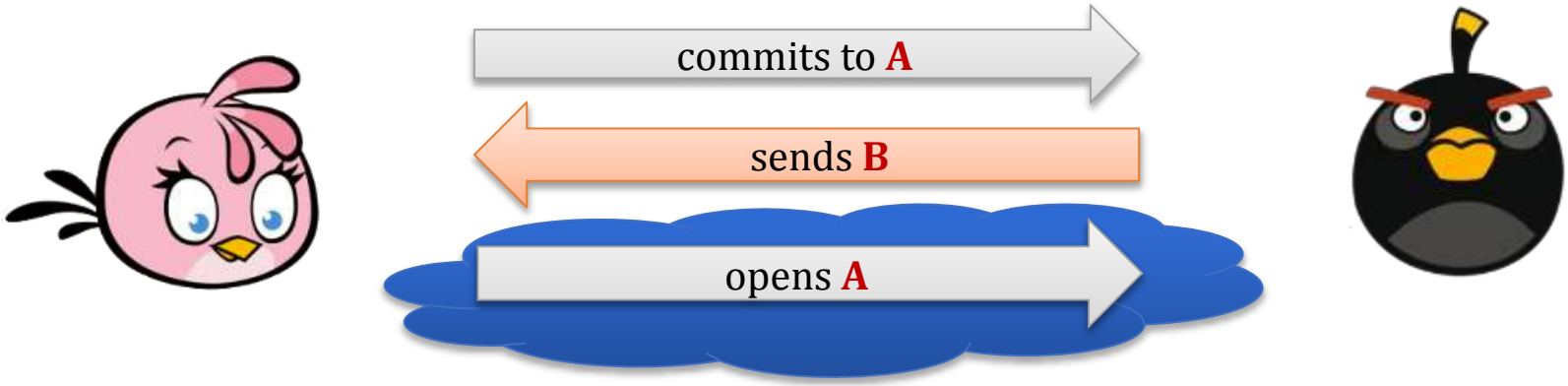
Alice "opens the commitment"

How does it solve the coin-flipping problem?



Problem 1

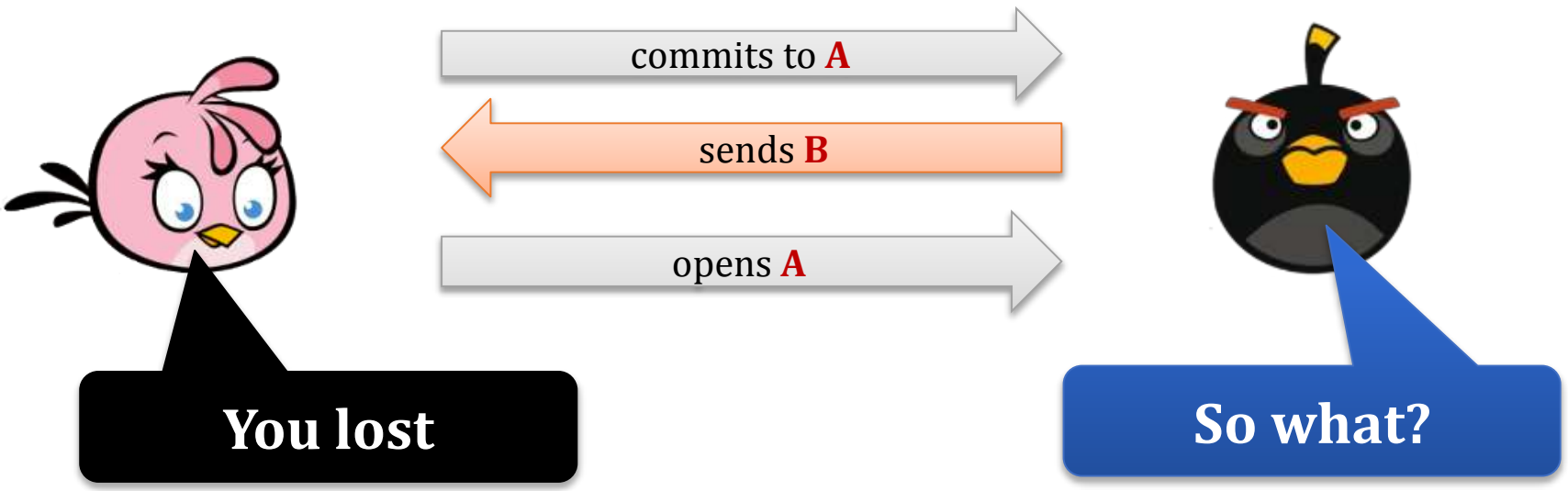
How to force Alice to open the commitment?



This is precisely the **lack of fairness** problem.

It's **inherent** to most of the interesting MPC protocols...

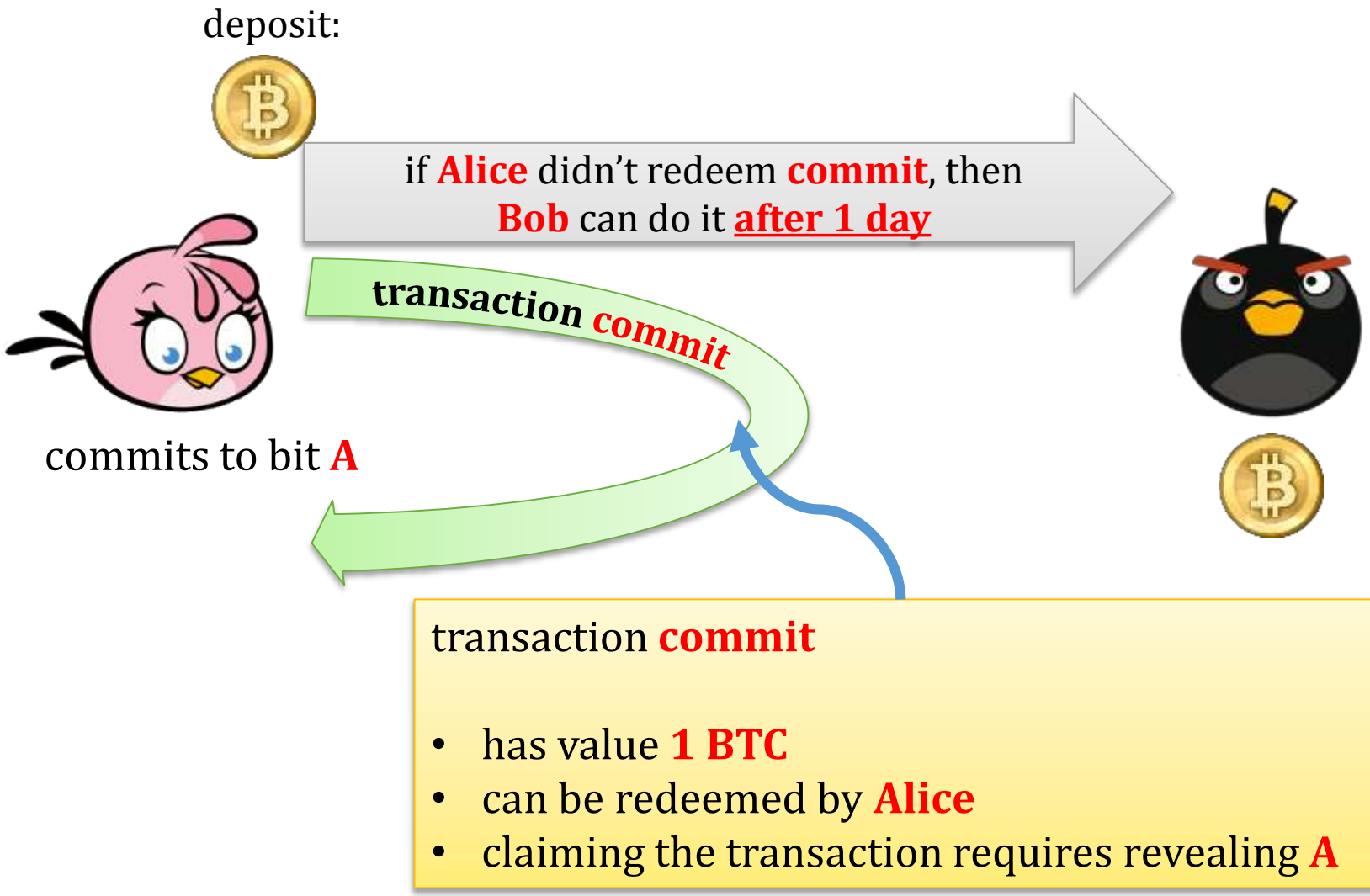
Problem 2



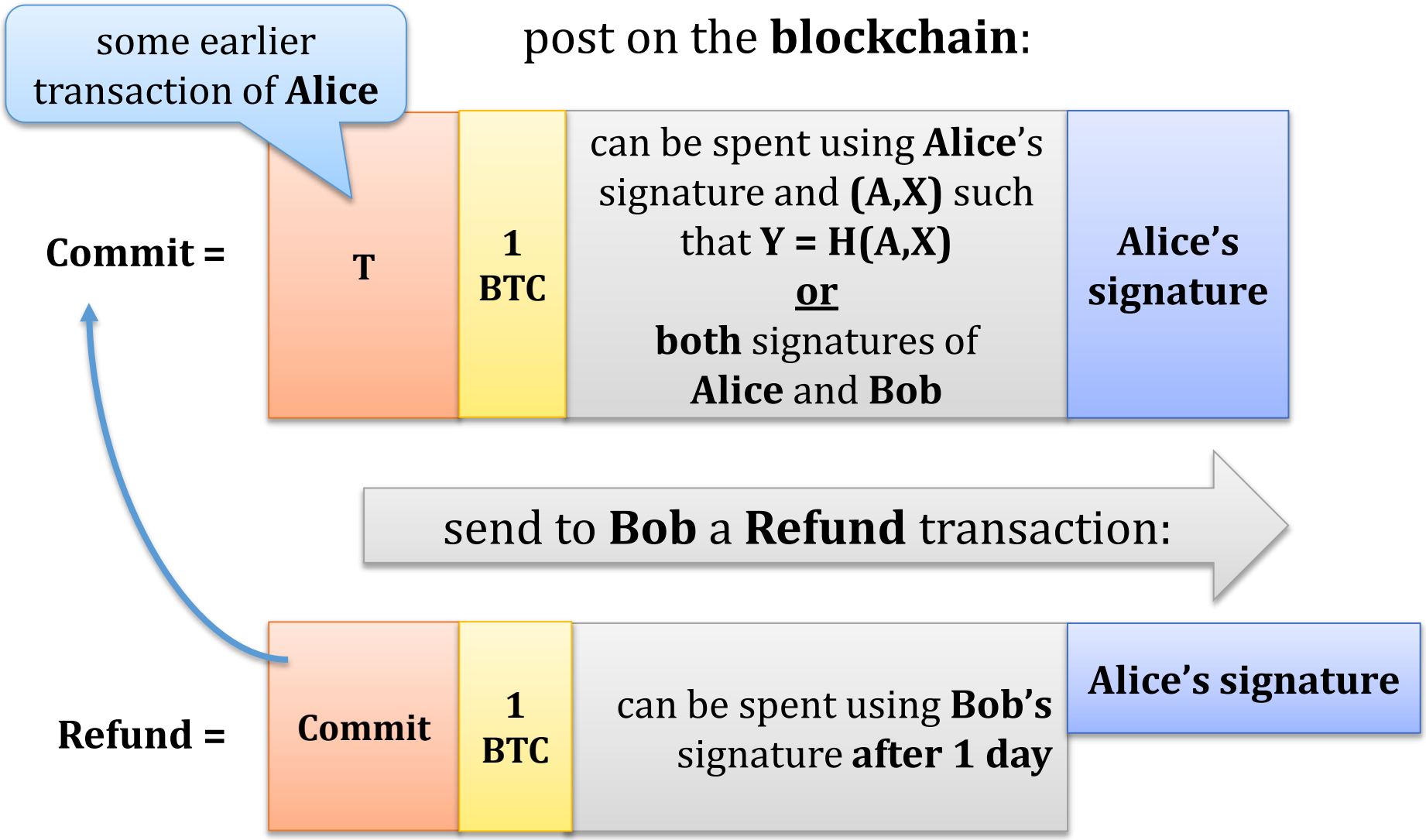
This is the problem of **forcing the parties to respect the output**.

Even more inherent (it is present also in the "**ideal world**" solution)

Idea: force the parties to open their commitments using the “deposits”



How can Alice commit to **A**?



This solves the problem of the lack of fairness!

commits with a **Bitcoin-based** commitment to **A**

sends **B**

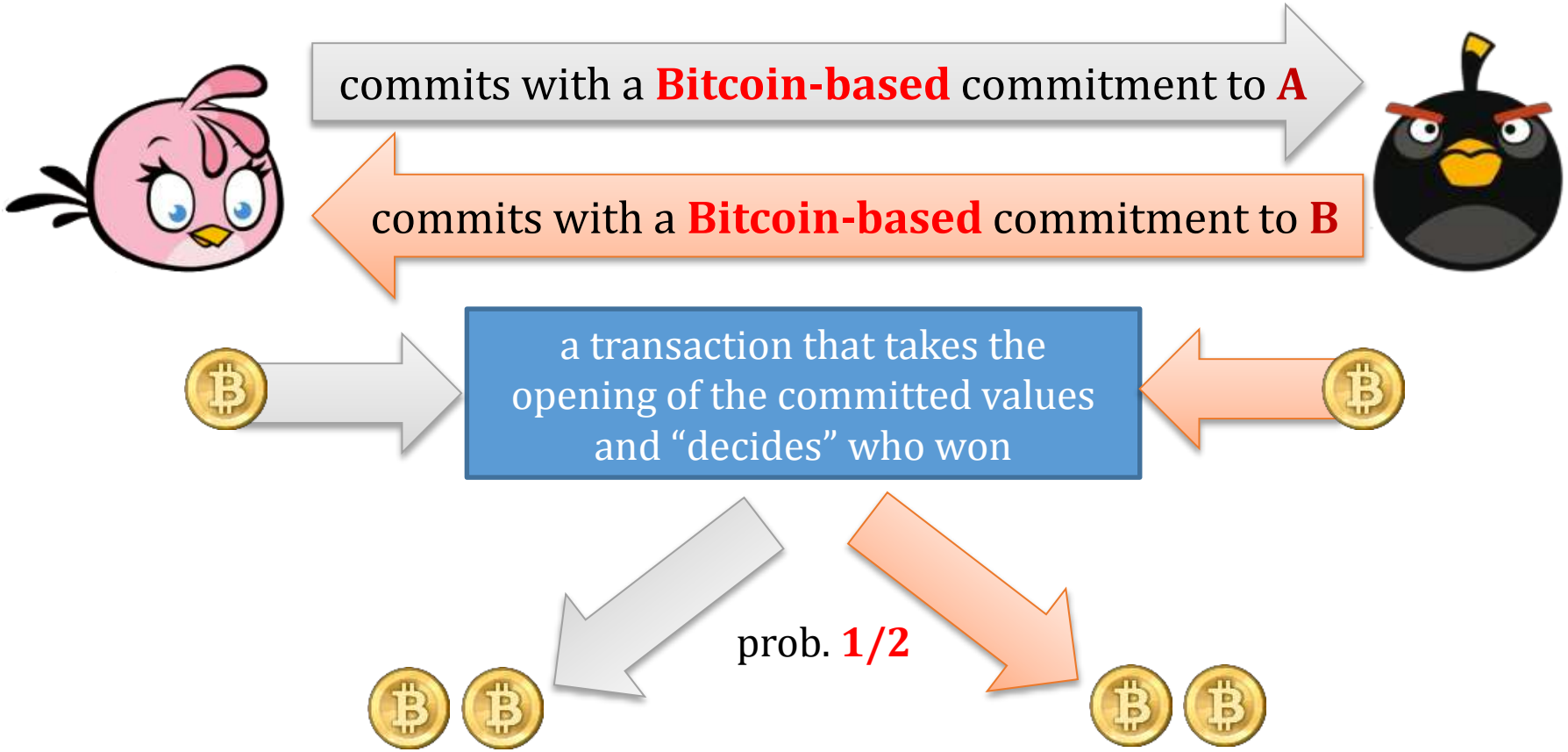
opens **A**

If **Alice** does not open her commitment within **1 day** then **Bob** can get her **1 BTC** by posting the **Refund** transaction with his signature

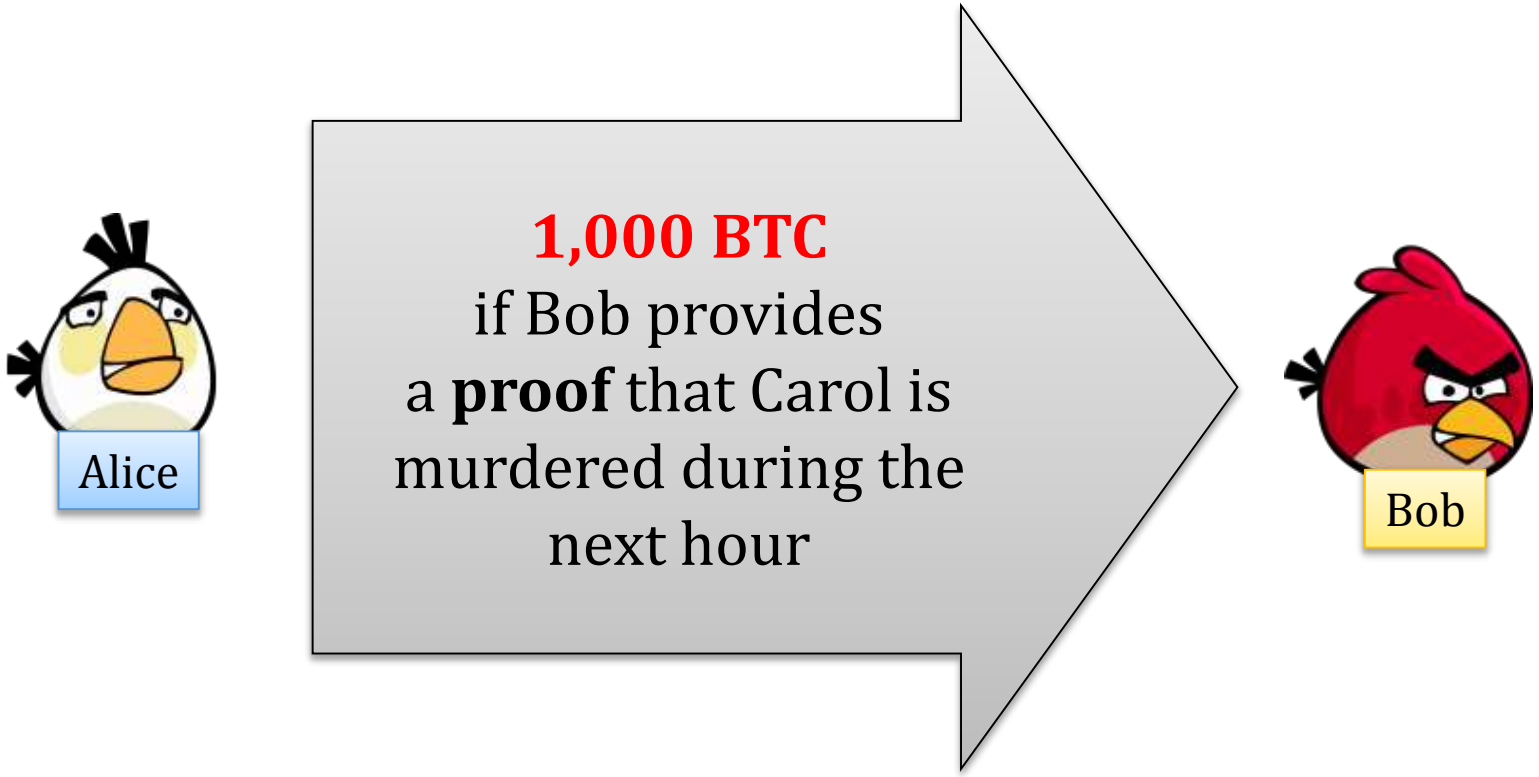
Otherwise she gets her **1 BTC** back.

What about the problem of respecting the outcome?

This can also be solved. Main idea:

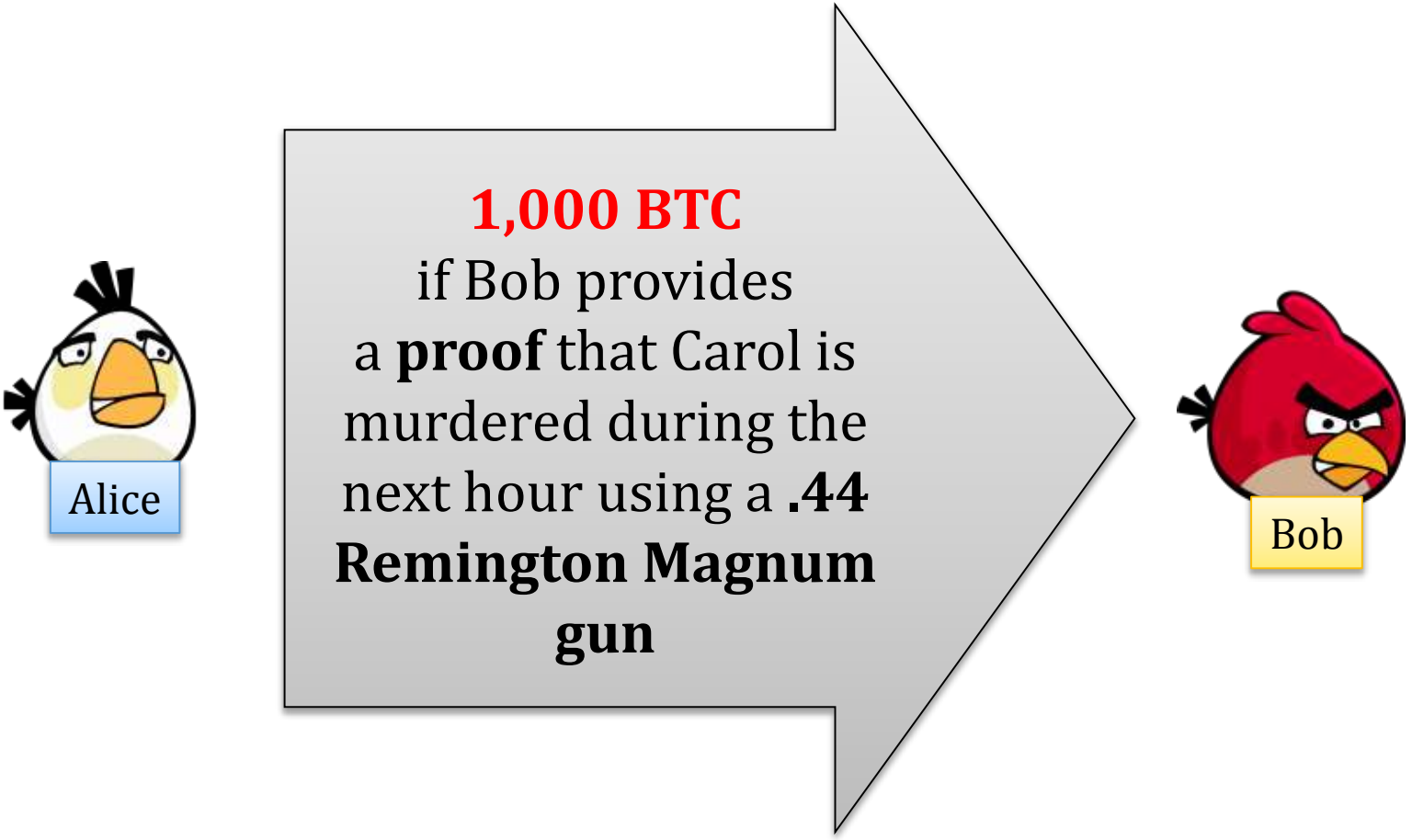


“Murder contract”



Question: what if Bob is just lucky and Carol was murdered by someone else?

Solution: add some details



How a such a “proof” can look like?

Examples:

- **signed article** from some press agency,
- “**authenticated data feed**”,
- several sources combined

Example

Alice

1,000 BTC

if Bob provides
an article containing texts:

- "**Carol was murdered**"
- "**.44 Remington Magnum gun**"

signed by Associated Press

Bob

Two technical problems

1. such conditions are **impossible to express using Bitcoin syntax**
2. a **separate "contract"** is needed for every potential hitman

Solution:

a currency designed for doing contracts.

Features

- has a concept of a "**contract**" that can be posted on the public register, and give money to **anyone** who provides some "solution"

- allows to create **arbitrarily complicated contracts**.