

Interactive Proofs and A Methodology for Defining Security

Serge Fehr

CWI

Content

Part I: Interactive proofs

- Interactive proofs
- (Interactive) proofs of knowledge
- Zero-knowledge (interactive) proofs
- Applications

Part II: A methodology for defining security

- The ideal/real-world paradigm
- An example

What is a Proof?

Interactive Proofs. Why???

Some properties:

Allows to prove:

- Convinces of the truth of a statement
 - Might be hard to find
 - "Efficiently" verifiable
 - Non-interactive
1. more!
 2. without giving away the proof!

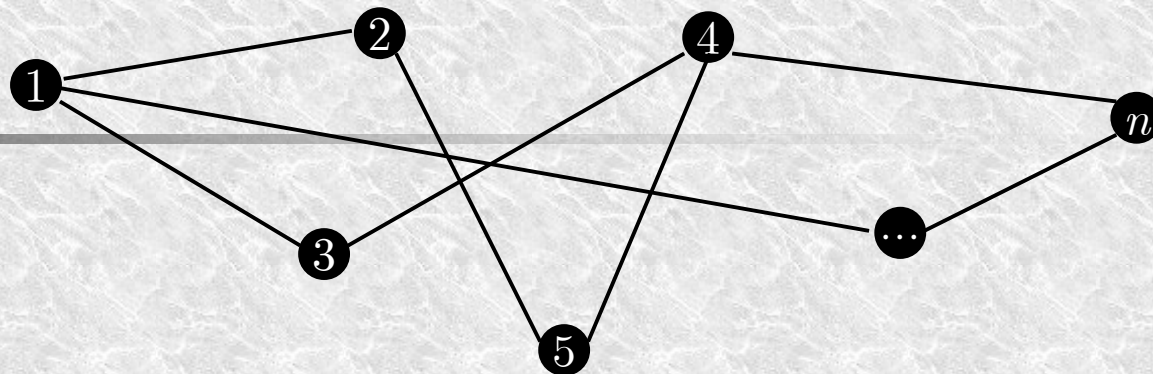
Complexity-theory point of view:

Given some language $L \subseteq \{0,1\}^*$.

Proof for statement $x \in L$: NP-witness w .

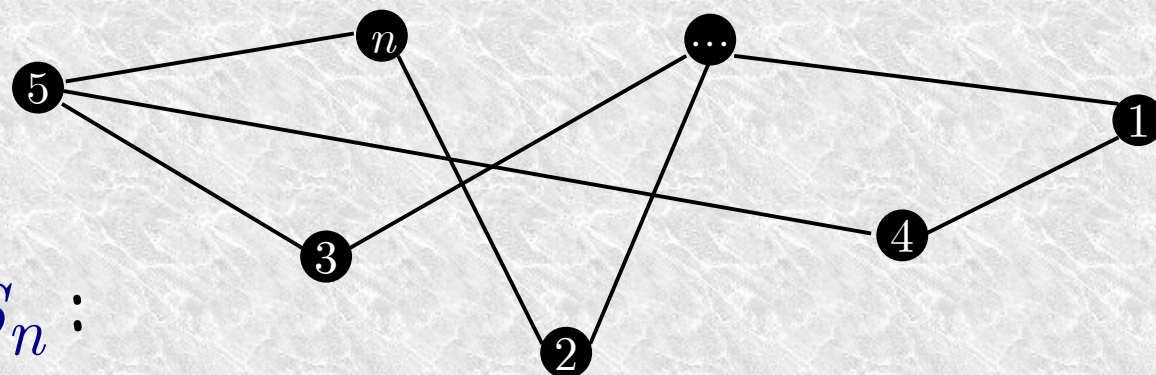
Provable languages = NP-languages.

Graphs



Graph:

$G = (V, E)$ with $V = \{1, \dots, n\}$ and $E \subseteq \{\{i, j\} \mid i, j \in V\}$



For permutation $\pi \in S_n$:

$\pi(G) := (V, \pi(E))$ where $\pi(E) \subseteq \{\{\pi(i), \pi(j)\} \mid \{i, j\} \in E\}$.

G and G' are **isomorphic** ($G \simeq G'$) $\Leftrightarrow \exists \pi \in S_n: \pi(G) = G'$.

It is (believed to be) **computationally hard** to decide if two graphs G and G' are isomorphic.

An Interactive Proof for GI

$$\pi: \pi(G_0) = G_1$$

G_0 and G_1 are isomorphic!

Hmmm! I could simply [GMW91] announce π , but I don't want \mathcal{V} to learn π ...



Prover \mathcal{P}



Verifier \mathcal{V}

$$\sigma \leftarrow S_n$$

$$H := \sigma(G_0)$$

$$\tau := \sigma \circ \pi^{-c}$$

H

c

τ

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

$$H \stackrel{?}{=} \tau(G_c)$$

Analysis

Consider any \mathcal{P}^* :

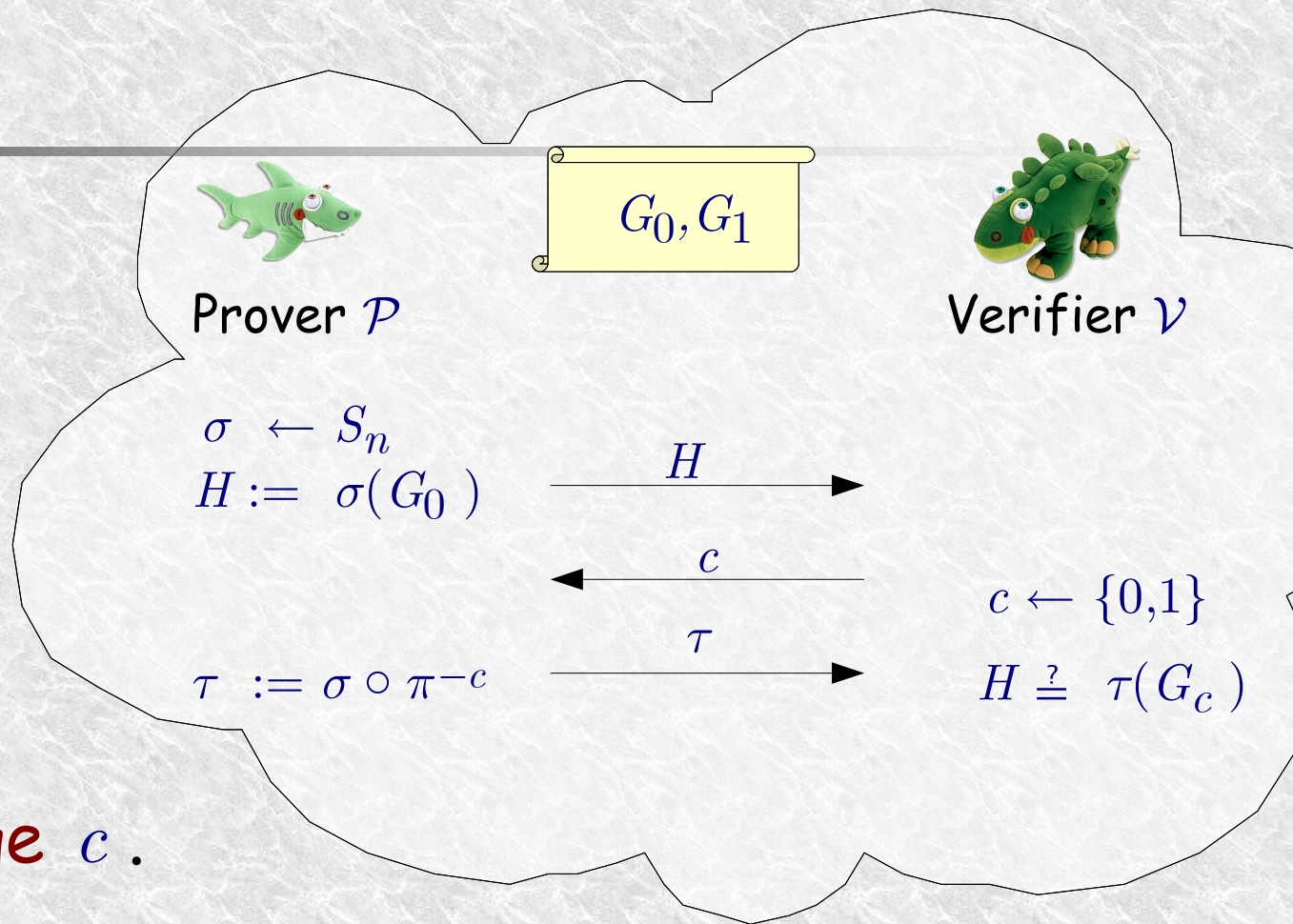
If $G_0 \neq G_1$ then
 $H \neq G_0$ or $H \neq G_1$.

$\Rightarrow \mathcal{P}^*$ can answer
at most 1 challenge c .

$\Rightarrow \mathcal{V}$ detects & rejects with probability at least $\frac{1}{2}$.

Probability can be boosted by repetition.

"Privacy": To be discussed later...



Definition: Interactive Proof

Let L be a language.

[Bab85,GMR85]

A protocol is an **(interactive) proof** for L with soundness error ϵ if it is efficient for \mathcal{V} and

Correctness:

If $x \in L$ then \mathcal{V} accepts with probability (close to) 1.

Soundness:

If $x \notin L$ then for any \mathcal{P}^* :

\mathcal{V} rejects except with probability ϵ .

An Interactive Proof for GNI

[GMW91]

G_0 and G_1 are **not** isomorphic!

Prove it!

G_0, G_1



Prover \mathcal{P}

I don't care about privacy,
but how can I prove at all
that $G_0 \neq G_1$?

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

$\sigma \leftarrow S_n$

$H := \sigma(G_c)$

compute $b : H \simeq G_b$

\xrightarrow{b}

$b \stackrel{?}{=} c$

Back to the GI Proof



Prover \mathcal{P}

G_0, G_1



Verifier \mathcal{V}

To successfully execute the proof it seems that \mathcal{P} needs to know π .

$$\begin{aligned} \sigma &\leftarrow S_n \\ H &:= \sigma(G_0) \end{aligned}$$

H

c

τ

$$\tau := \sigma \circ \pi^{-c}$$

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

$$H \stackrel{?}{=} \tau(G_c)$$

Thus, informally:

The proof not only convinces of the fact that $G_0 \simeq G_1$, but also that \mathcal{P} knows π .

How to formalize this?

Definition: Proof-of-Knowledge (PoK)

[GMR85,BG92]

Let L be an NP-language:

$x \in L \Leftrightarrow \exists$ efficiently verifiable NP-witness $w \in W_L(x)$.

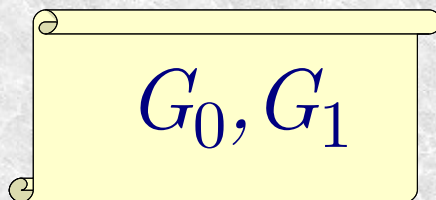
An interactive proof for L is a **proof-of-knowledge (PoK)** with soundness error ε if there exists an efficient extractor \mathcal{E} with black-box access to \mathcal{P}^* , such that for any $x \in L$ and any \mathcal{P}^*

$$\text{Prob}[\mathcal{E}(x) \in W_L(x)] \geq \text{Prob}[\mathcal{V}[\mathcal{P}^*](x) = \text{accept}] - \varepsilon .$$

Knowledge Extractor for the GI Proof



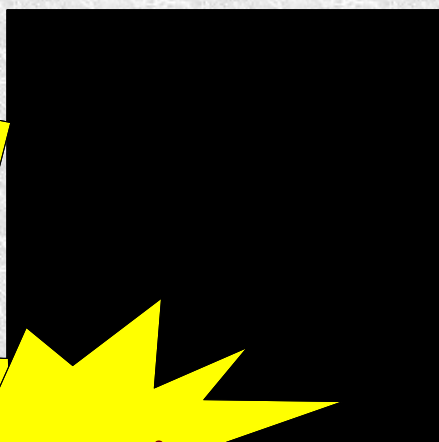
Prover \mathcal{P}^*



G_0, G_1



Extractor \mathcal{E}



Rewind

H

$c = \emptyset$

τ'

$H \stackrel{?}{=} \tau(G_0)$

$H \stackrel{?}{=} \tau'(G_1)$

If $H = \tau(G_0)$ and $H = \tau'(G_1)$ then
output $\pi := \tau'^{-1} \circ \tau$, else fail.

The Privacy Issue of the GI Proof



Prover \mathcal{P}

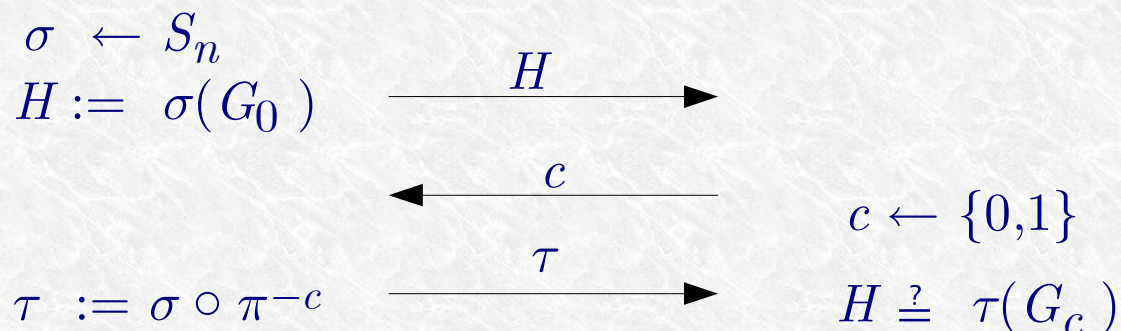
G_0, G_1



Verifier \mathcal{V}

What about "privacy"?

\mathcal{V} does not obviously learn π .



More:

\mathcal{V} does not learn anything:

all info \mathcal{V} obtains, H, c, τ , he can produce from scratch:

sample $c \leftarrow \{0,1\}$ and $\tau \leftarrow S_n$, and compute $H := \tau(G_c)$.

Will see: even holds for dishonest \mathcal{V}^*

Definition: Zero-Knowledge

[GMR85]

An interactive proof for L is **zero-knowledge (ZK)** if

$\forall \mathcal{V}^* \exists$ **simulator** \mathcal{S}

- with **similar running time** than \mathcal{V}^* 's :
- which **does not interact** with \mathcal{P}

such that

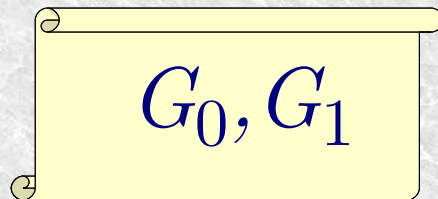
$$\mathcal{S}(x) \approx \mathcal{V}^*[\mathcal{P}](x) \quad \forall x \in L ,$$

i.e., the outputs of \mathcal{V}^* and \mathcal{S} are **equally distributed**.

ZK-Simulator for the GI Proof



Simulator \mathcal{S}



Verifier \mathcal{V}^*

$c' \leftarrow \{0,1\}, \tau \leftarrow \mathcal{S}_n$

$H := \tau(G_{c'})$

H

c

τ

while $c \neq c'$:

rewind & repeat

output whatever \mathcal{V}^* outputs

out

Simple analysis: $\mathcal{S}(G_0, G_1) \approx \mathcal{V}^*[\mathcal{P}](G_0, G_1)$

Composability of ZK-Proofs

ZK is preserved under **sequential composition**:

S can simulate one execution after the other \Rightarrow **running time scales linearly**

Is **not** preserved under **parallel composition**:

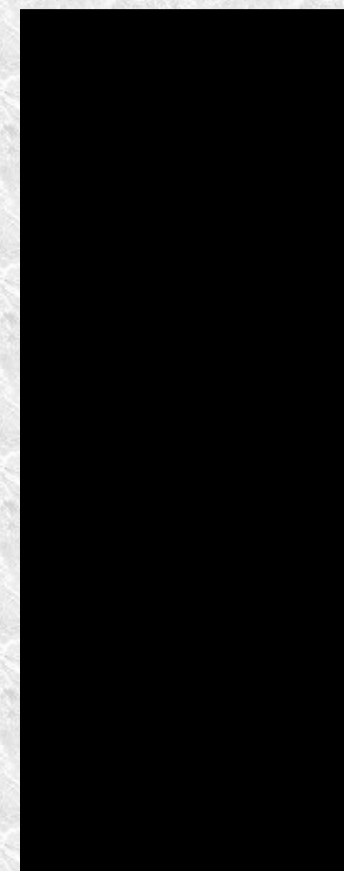
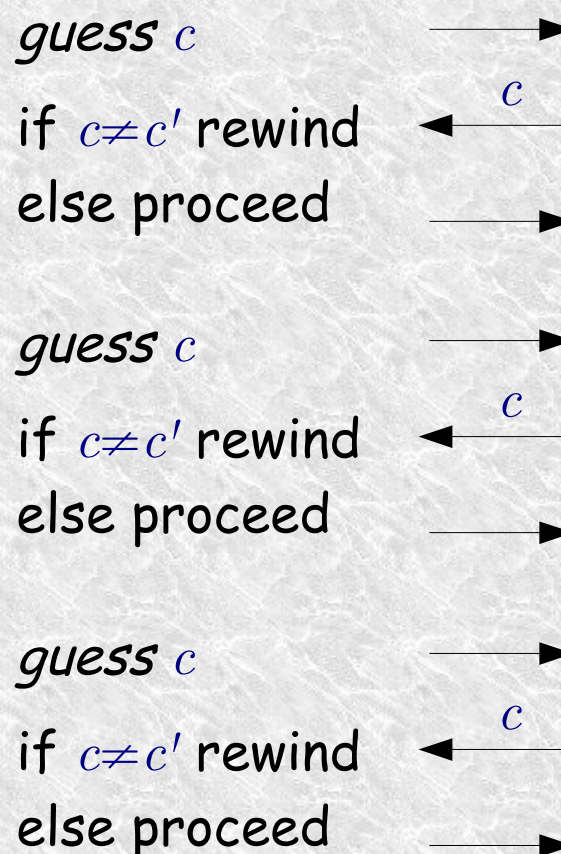
S needs to guess all c 's at once \Rightarrow **exponential running time**



Simulator S



Verifier V^*



out



Another Example: Schnorr's DL-Proof

Setting: G = group of large prime order q (e.g. $G \subset \mathbb{Z}_p^*$)

g = generator of G

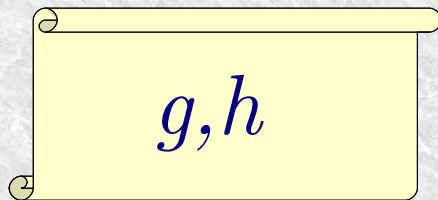
$h = g^x$ for a random $x \in \mathbb{Z}_q$

I know x !



Prover \mathcal{P}

g, h



Prove it!!!



Verifier \mathcal{V}

$$r \leftarrow \mathbb{Z}_q, a := g^r$$



$$z := r + cx$$



$$c \leftarrow C \subseteq \mathbb{Z}_q$$

$$g^z \stackrel{?}{=} ah^c$$

Application I: Secure Identification

PKI set-up: every user U_i is associated with $pk_i = g^{x_i}$,
corresponding $sk_i = x_i$ is (only) known to U_i .

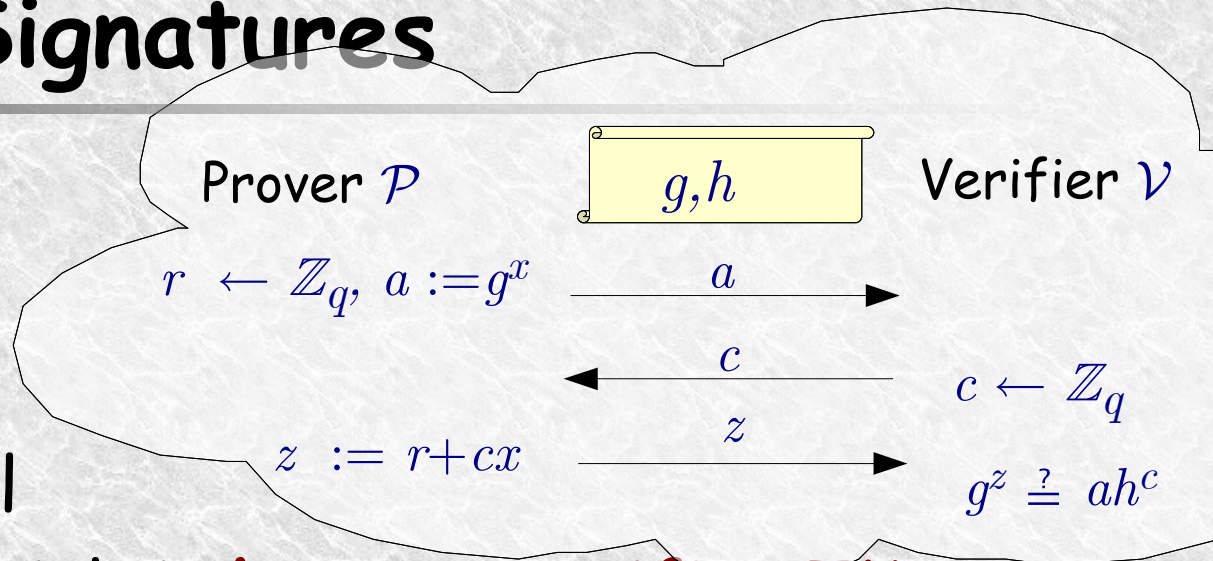
In order to **identify** himself to U_j , user U_i can do a
ZK proof-of-knowledge (that he knows $sk_i = x_i$).

ZK $\Rightarrow U_i$ gives away **no info at all** (in particular not on x_i),
 U_j does not even get a convincing "receipt"

PoK \Rightarrow prover must know x_i , and thus must be U_i .

Application II: Signatures

Consider e.g. Schnorr's DL-proof with $C = \mathbb{Z}_q$, which is a **PoK** with small soundness error ϵ , and which is **honest-verifier ZK**.



The following is a **heuristically-secure signature scheme**: public and secret key are $pk = h = g^x$ and $sk = x$, and $sig_x(m) = (a, c, z)$ such that $g^z = ah^c$ for $c = Hash(a, m)$.

Heuristic: $c = Hash(a, m)$ "behaves" like a random, so that

PoK \Rightarrow signer knows $sk = x$,

ZK \Rightarrow signatures give away no info

\Rightarrow security against chosen message attack

Achievability Results

Theorem [Sha92]: The class **IP** of languages that have a (not necessarily ZK) **interactive proof** equals

$$\mathbf{IP} = \mathbf{PSPACE}.$$

Note: Only special languages have unconditionally-secure zero-knowledge proofs, but if we allow computational assumptions, then

Theorem [GMW91]: Every language in **NP** has a **computationally-secure ZK proof** (of knowledge).

Content

Part I: Interactive proofs

- Interactive proofs
- (Interactive) proofs of knowledge
- Zero-knowledge (interactive) proofs
- Applications

Part II: A methodology for defining security

- The ideal/real-world paradigm
- An example

Defining (and Proving) Security

"What is the right way to **define** security for a cryptographic scheme?"

and thus

"... to **prove** a cryptographic scheme secure?"

The Property-Based Approach

List the security properties you require:

Encryption:

- **Connectness** ("decryptability")
- **Commitment** ("commitment hides message")
- **Privacy** ("ciphertext gives no info on plaintext")
- **Blinding** ("commitment hides message")

Oblivious Transfer:

- **Binding** ("unique opening")
- **Obliviousness** ("receiver gets only one bit")

Multi-Party Computation:

- **Correctness** ("output is correct")
- **Privacy** ("adversary learns nothin beyond output")
- **Input-independence**
- ...

Problems with this Approach

Problem 1:

Do we have all security properties (that we need)?

Problem 2:

What happens if the protocol is composed?

The Ideal/Real-World Approach

Typically, one has some **idealized version** in mind, like an **honest party** that faithfully executes the task:

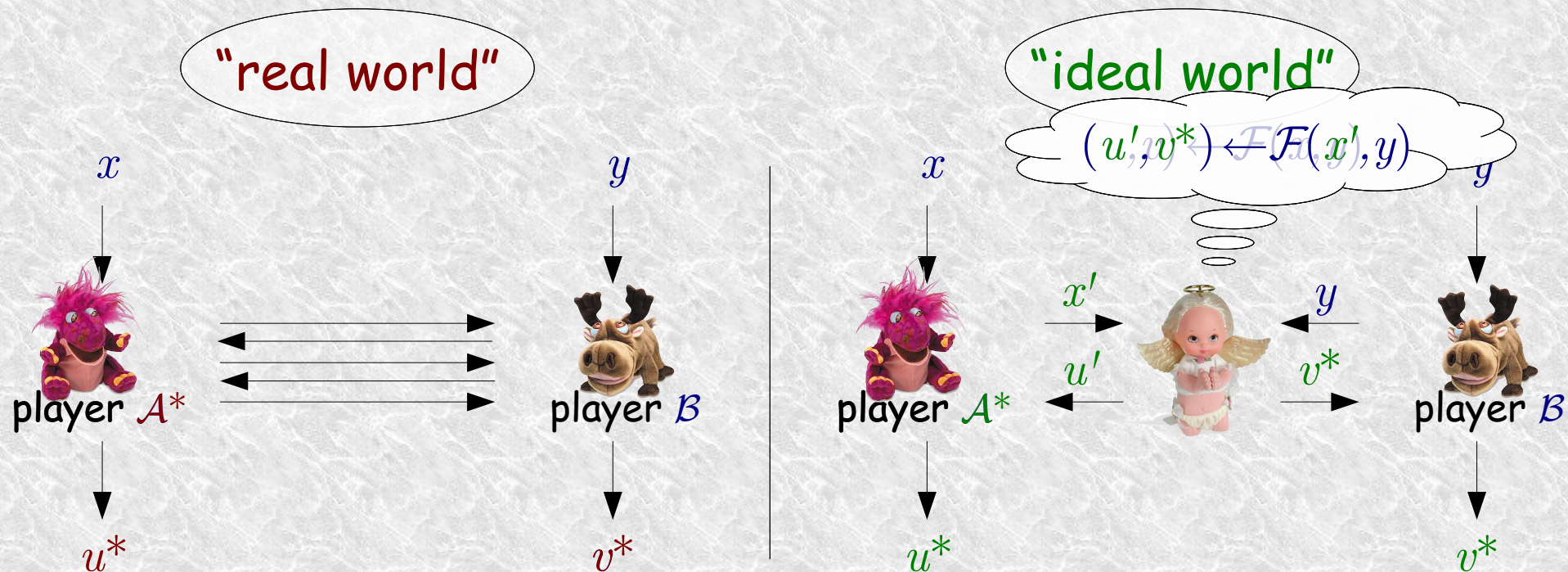
- Encryption: delivers message privately from A to B.
- Commitment: stores message and reveals it on committer's request.
- OT: forwards the one bit the receiver asks for.
- Multi-Party Computation: computes the function.

Idea [Bea, Gol, Can,...]:

Require the scheme to be:

as secure as the idealized version.

Formally: Real World vs Ideal World



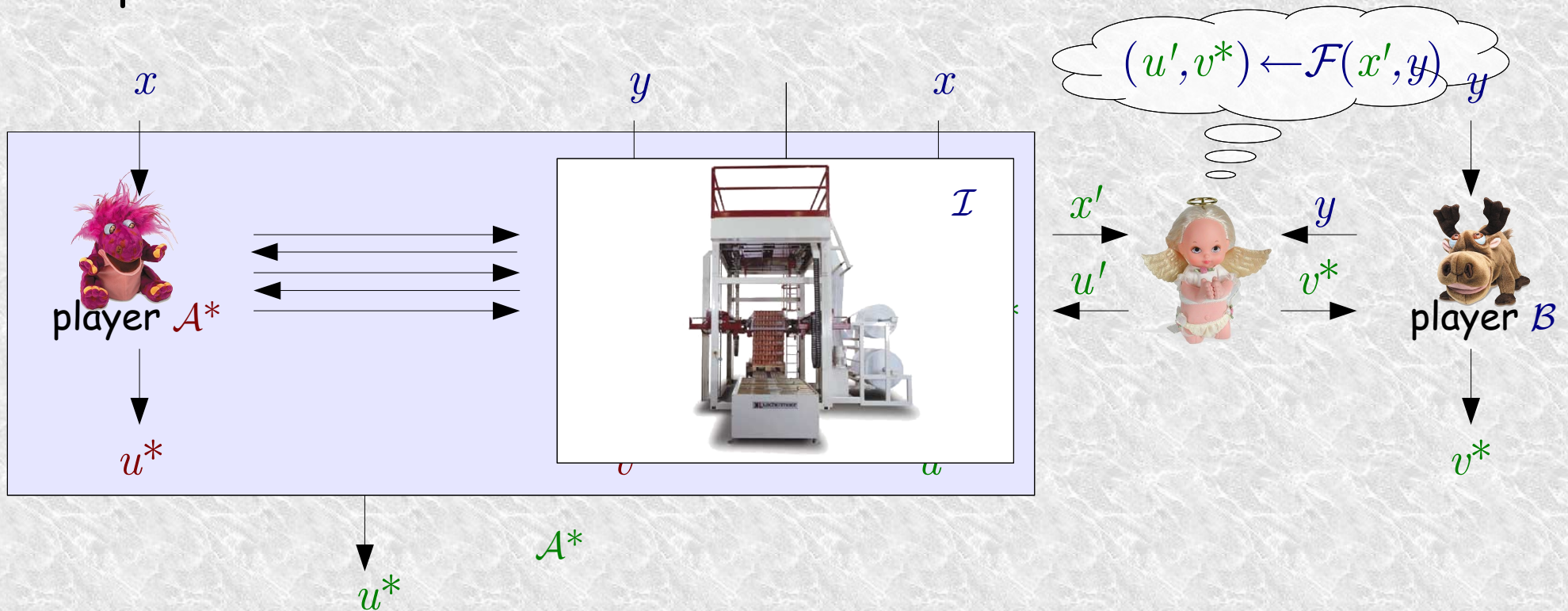
Require:

$$\forall A^*_{real} \exists A^*_{ideal} \forall x, y : (u^*_{real}, v^*_{real}) \approx (u^*_{ideal}, v^*_{ideal})$$

perfectly/statistically/computationally **indistinguishable**

How to Actually **Prove** Security

To prove: **real-world attack** \Rightarrow **ideal-world attack**



Show: \exists interface \mathcal{I} between A^*_{real} and the **ideal world**, which

1. extracts x' from A^*_{real} so that $v^*_{real} \approx v^*_{ideal}$,
2. simulates A^*_{real} 's view, so that $u^*_{real} \approx u^*_{ideal}$.

Some Properties

- **Applicable** to (m)any cryptographic task
- Intuitive very **meaningful**
- Captures **all security properties** of the task
- Implies security under **composition**:
 - **sequential** for the definition as given,
 - **universal** for an even stronger formulation.

Example: Oblivious Transfer (OT)



Properties:

- $\exists c^*$ such that B^* has no info on b_{1-c^*} , given b_{c^*} .
- A^* has no info on c .

A Candidate OT Scheme

Set-up: B knows sk of a homomorphic public-key bit-encryption scheme, i.e., $\lambda \cdot E(b) + E(b') = E(\lambda b \oplus b')$.

Security against dishonest B^* :
 Whatever bit c^* is encrypted by C , B^* gets a random encryption $E(b_c^*)$ of b_c^* , and thus no info on b_c^* .

Security against dishonest A^* :
 The only info A^* has on c is from $C = E(c)$. Thus, c is computationally hidden from A^* .

player A player B

\xleftarrow{C}

$C := E(c)$

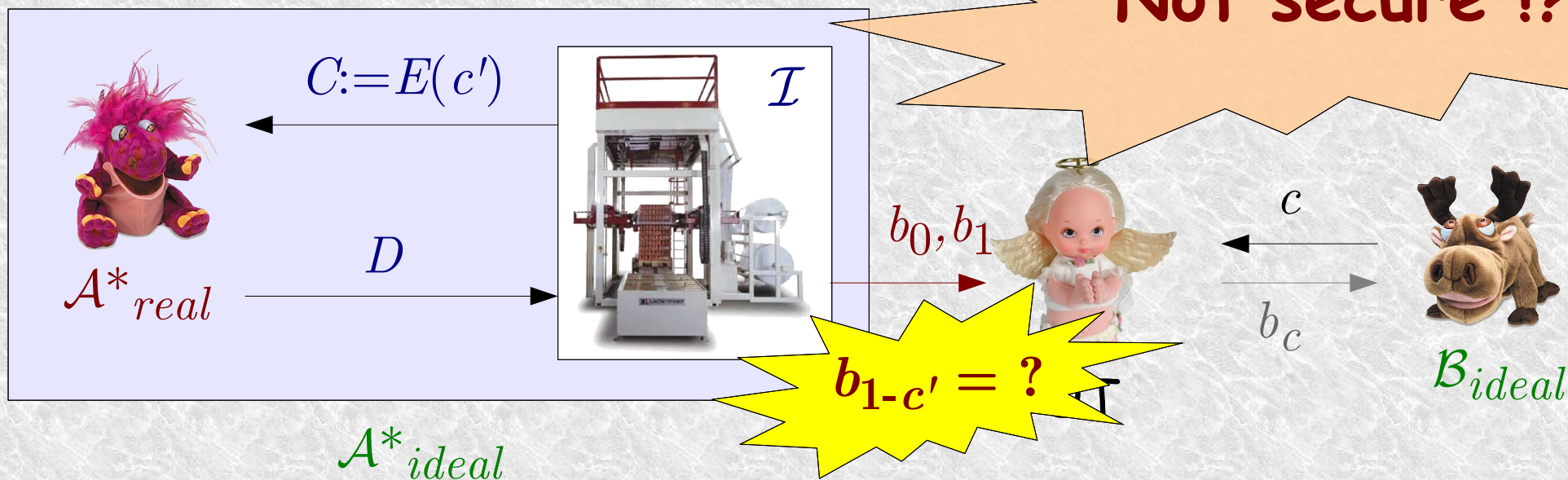
$D := b_0 \cdot (E(1) + C) + b_1 \cdot C + E(0)$ \xrightarrow{D}

$$= E(b_0(1 \oplus c) \oplus b_1 c) = E(b_c)$$

Decrypt D and output the result, b_c .

What about Ideal/Real-World Security?

Let's try:



Problem: \mathcal{I} needs to provide OT with b_0 and b_1 , such that B_{ideal} outputs the same as B_{real} would, if they choose the same c , but \mathcal{I} only learns one of the two.

Rewinding and asking for both bits does not help, as A^*_{real} could choose b_0 and b_1 depending on C .

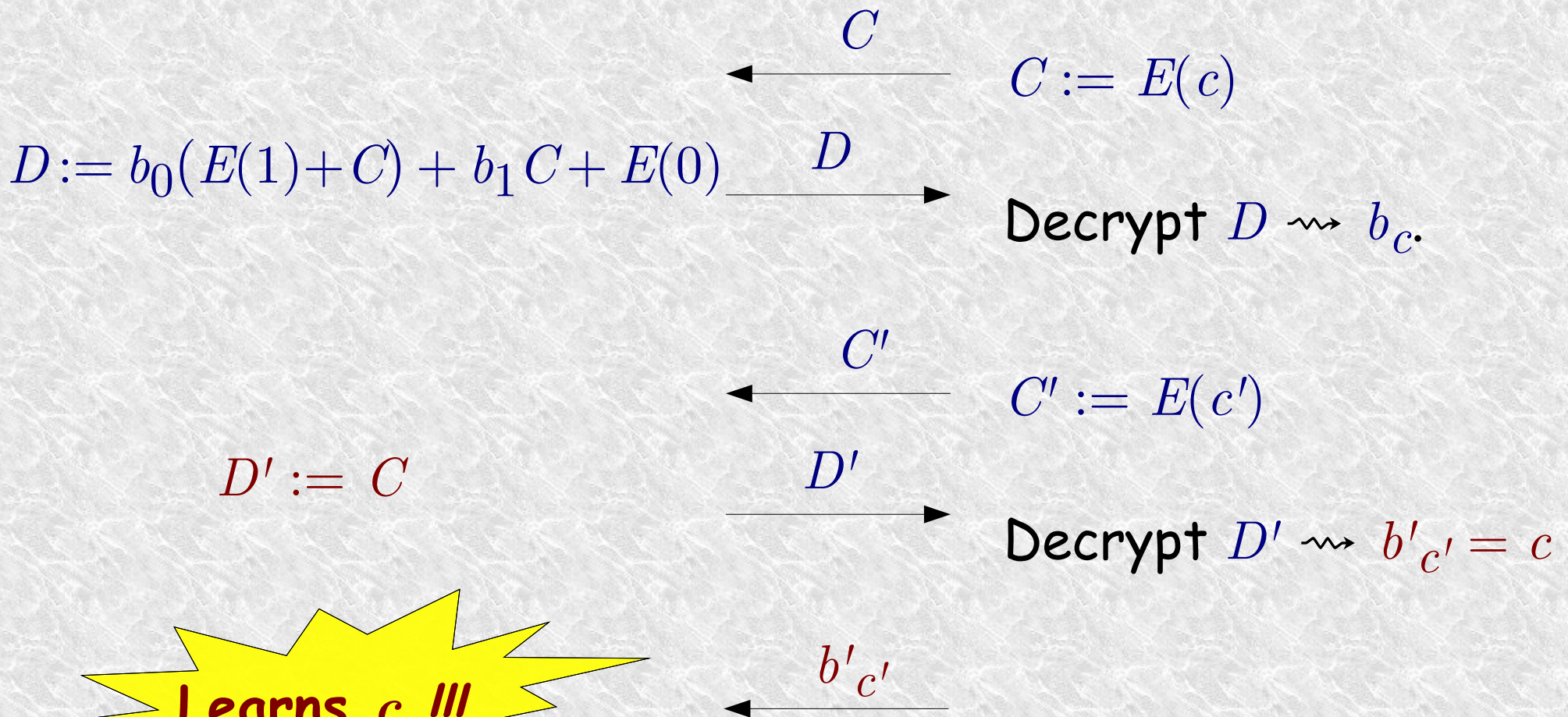
A Security Problem



A^*



B



Learns c !!!

Fixing the Scheme



player A

$b_0, b_1 \in \{0,1\}$

Zero-knowledge proof of knowledge of c .



player B



C

$$C := E(c)$$

$$D := b_0(E(1) + C) + b_1 C + E(0)$$



D

Decrypt $D \rightsquigarrow b_c$.
Allows \mathcal{I} to **extract** b_0, b_1 .

Zero-knowledge proof of knowledge of b_0, b_1

such that $D = b_0(E(1) + C) + b_1 C + E(0)$.

(Actually witness-indistinguishable proof suffices.)

Moral: Security \neq Security

- Security of a scheme is **wrt. a specific definition**.
- Ideal/real-world paradigm gives "good" security definition.
- **Failure** of being secure in that sense: indicator of possible unexpected (insecure) behavior \Rightarrow **use with care!**
- **Being** secure: behaves as expected (**plug'n'play**), remains **secure under composition** (sequential/universal).
- **BUT:** The stronger the security definition, the harder to construct schemes that satisfy them.