Composing Quantum Protocols

Dominic Mayers
Université de Sherbrooke

Part I: Composability Theorem. Joint work with **Michael Ben-Or**

Part II: Composability of QKD. Work (in progress) with **Michael Ben-Or, Michal Horodecki, Debbie Leung and Jonathan Oppenheim**

Main Question I

Protocols often have subprotocols. For example, quantum key distribution (QKD) uses authentication as a subprotocol to make sure that (1) the classical messages sent in the protocol are not modified by Eve and (2) Eve cannot pretend to be Alice or Bob.

QKD

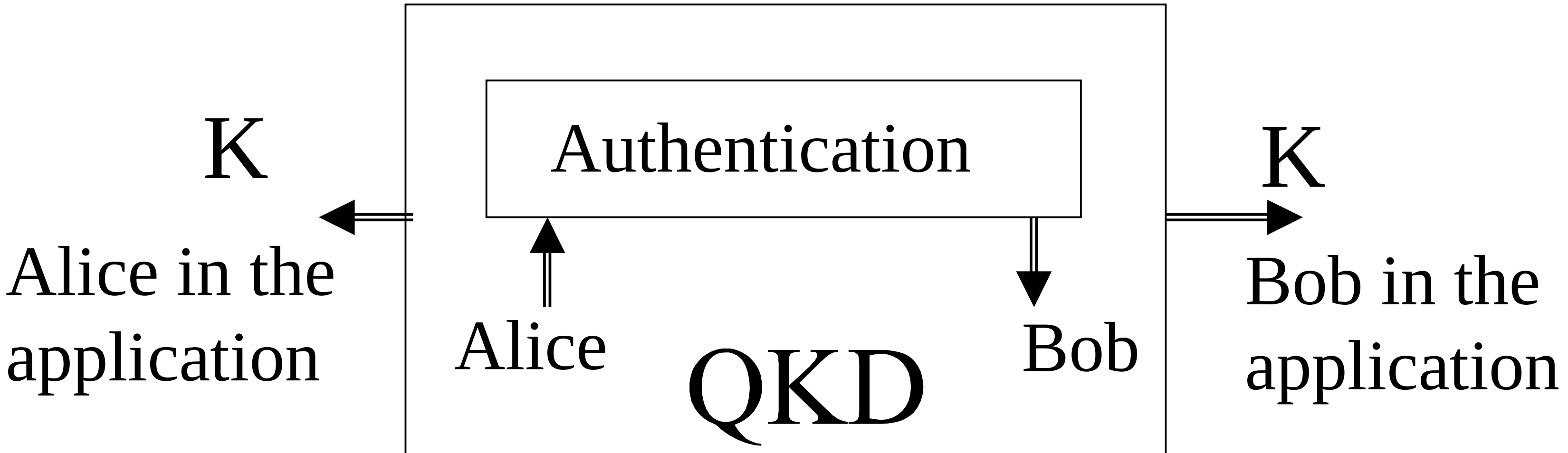
⋮

Alice announces the bases [using an authenticated channel].

⋮

Main Question II

To design a QKD protocol we need to know the ideal authentication task, but we should not need to know which authentication protocol is used. In general, the ideal task is the only thing that the designer of an application protocol should need to know about.



Main Question III

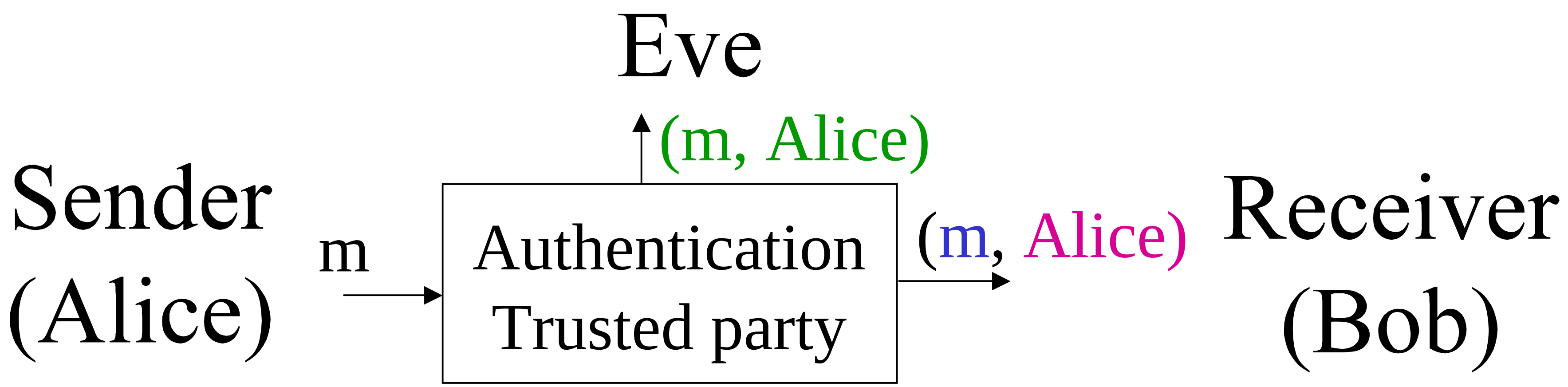
The main general question is how do we prove that a protocol provides what is promised by its associated ideal task? We want to make sure that the protocol can be used in all properly designed application protocols!

This is composability!

Introduction through an example

Authentication with QKD as a subprotocol

Ideal Authentication

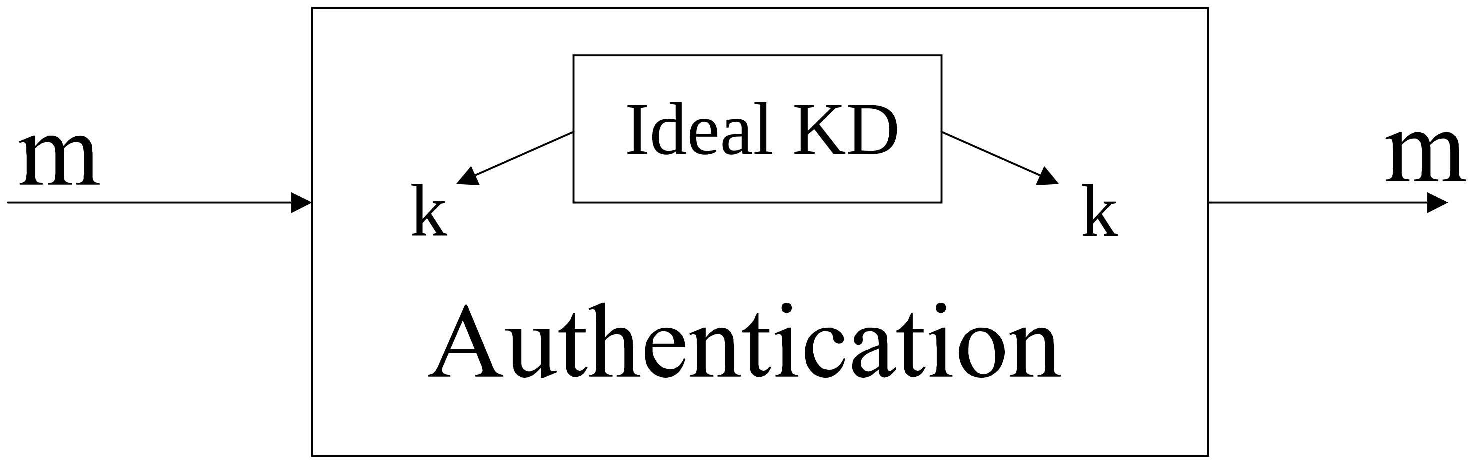


It is as if a trusted party took the message from the sender and delivered it unmodified to the receiver together with the sender identity. It also sends this information to anyone else who asks for it.

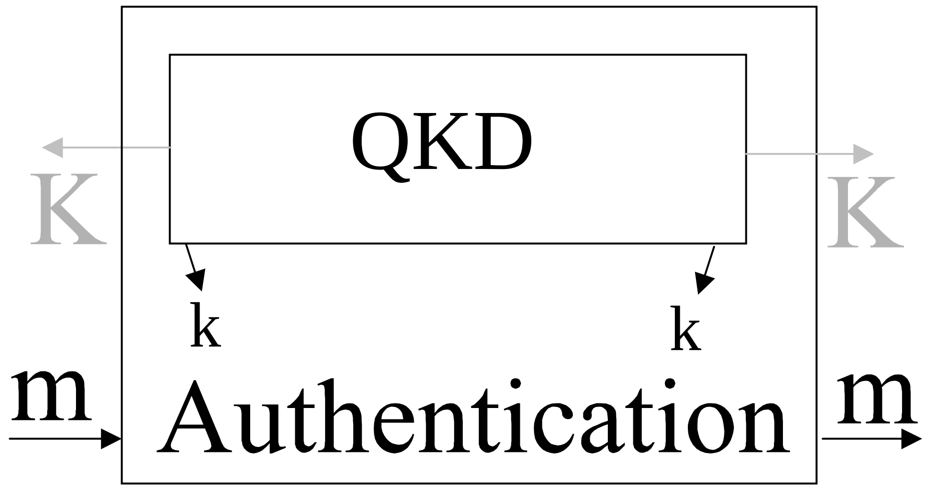
Real Authentication Protocols

There exists unconditionally secure classical protocols for classical messages. However, they require that Alice and Bob initially share a small private key k.

The small key is an internal resource, not an input. The Ideal KD which generates the small key is a subprotocol.



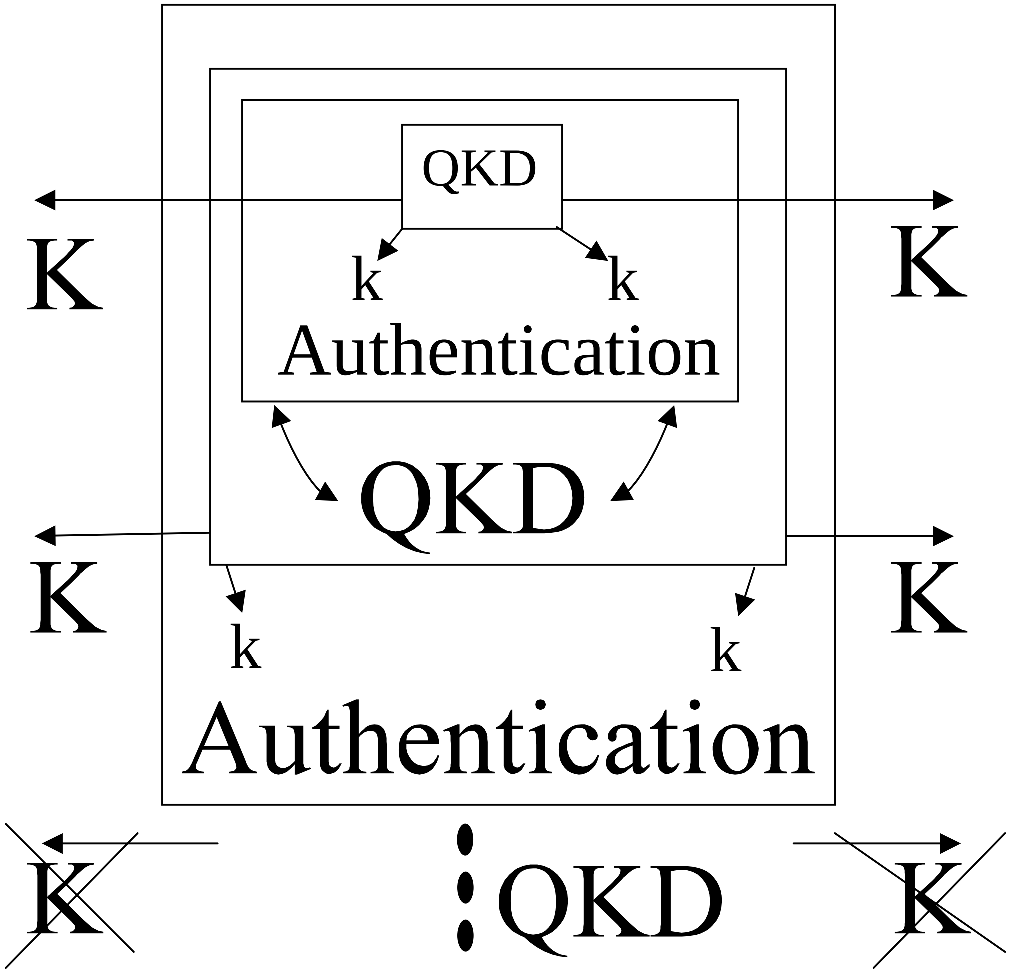
A composability question



What about the security of an authentication protocol when a real QKD protocol, not an ideal one, is used as a resource. Does the real QKD protocol provides what is promised?

Key Degradation

A negative answer could mean an important degradation of the key after a few repetitions.

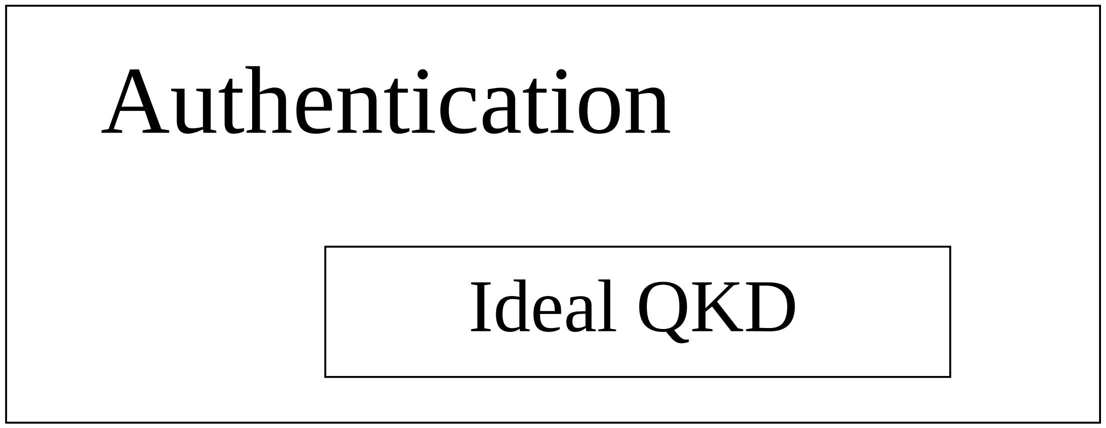


Ben-Or, Michal Horodecki, Leung, Mayers and Oppenheim (in progress)

Notation

G(A) denotes a protocol G with a subprotocol A.

Example: Authentication(Ideal QKD)



Modularity in Security Definition

$G(\beta)$ realises γ

B realises β

γ = Ideal Authentication, G = Authentication protocol.
β = Ideal QKD, B = QKD protocol,

The designers of the application protocol G should only worry about the definition of β and γ. The designer of the protocol B should only worry about the definition of β.

What is needed.

- A model for the ideal tasks β
- A model for the application protocols $G(\beta)$ and the subprotocols B
- A definition of the relation « B securely realises β » (simply noted « B s.r. β »).
- A general composability theorem:

  (a) B s.r. $\beta \wedge G(\beta)$ s.r. $\gamma \Rightarrow G(B)$ s.r. γ

  (b) B s.r. $\beta \Rightarrow B^{(m)}$ s.r. $\beta^{(m)}$

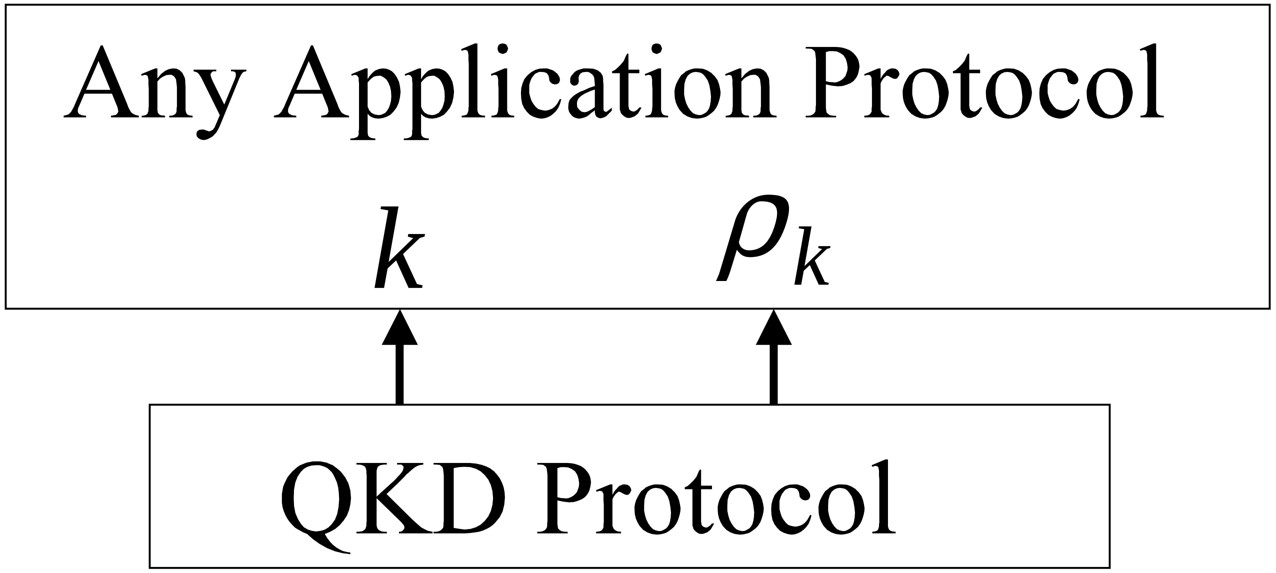
Back to the QKD example…

Alice’s key Vs Bob`s key

The security of QKD requires that Alice`s key and Bob’s key are almost always identical.

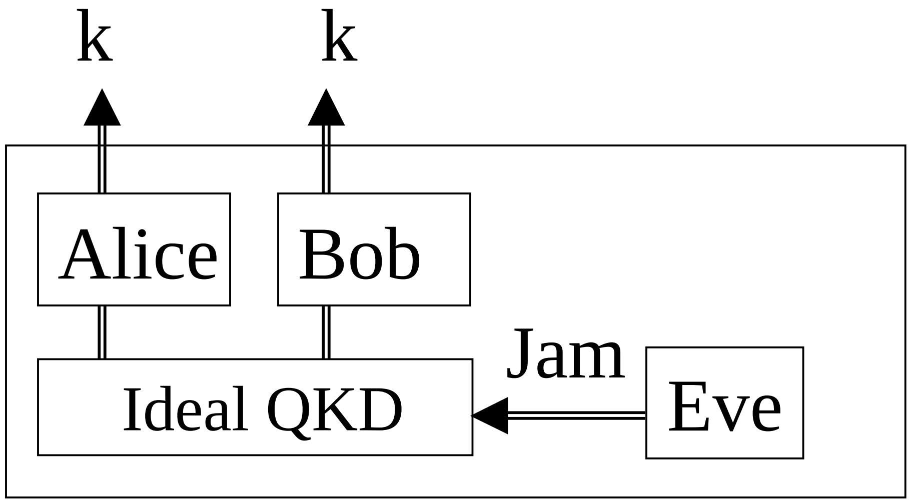
For simplicity, we will assume it is required that they are always identical.

QKD into an Application Protocol



Eve in the application protocol obtains a state ρ_k which provides a very small amount of information about the key k. The honest parties in the application protocol also receive the key.

Ideal QKD Protocol



If Jam = 1, k = fail.

If Jam = 0, $k \in_R \{0,1\}^m$

In an ideal QKD protocol, the participants interact directly with an ideal party which provides the key. The participants directly output this key to the application protocol.

Real versus Ideal QKD

$$\hat{\rho}_1 = \sum_k p(k) \left| k \right\rangle \left\langle k \right| \otimes \rho_k$$

Real QKD

$$\hat{\rho}_0 = \sum_k 2^{-m} \left| k \right\rangle \left\langle k \right| \otimes \rho$$

where $\rho = \sum_k 2^{-m} \rho_k$

Ideal Private QKD

$$k \in \{0,1\}^m \text{ Y } \{\text{fail}\}$$

The real protocol ε-securely realises the ideal private QKD if

$$SD(\hat{\rho}_0, \hat{\rho}_1) \leq \varepsilon$$

$$SD(\hat{\rho}_0, \hat{\rho}_1) = \mathrm{I}_{acc}(\mathrm{E}) \quad \text{where } \mathrm{E} = \{(\hat{\rho}_0, 1/2), (\hat{\rho}_1, 1/2)\}$$

Uniformity Vs Privacy

The security of QKD is not only a small mutual information. We must also require a priori uniformity, i.e., in the ideal case, for all k, $p(k \mid \text{Jam} = 0) = 2^{-m}$.

Universal Security Definition

Borrowing from Classical Composability.

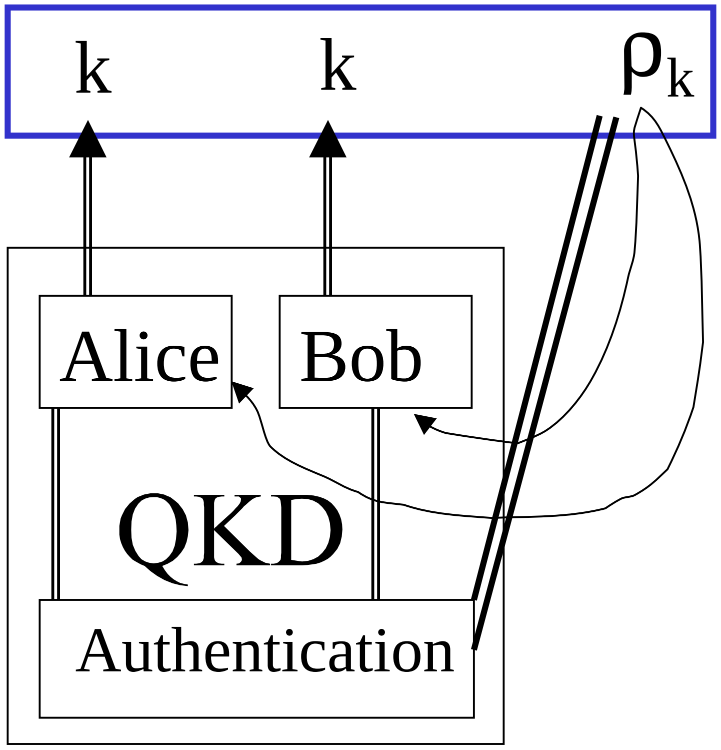
The issue of composability is important in standard cryptography and was progressively addressed in the last 10 years! The techniques currently used for classical composability can be useful to build a theory of quantum composability.

The Intuition behind the Universal Security Definition

- Intuitively, the fact that Eve cannot distinguish between the real and the ideal protocols should allow us to securely replace the ideal QKD protocol by the real QKD protocol in any application protocol.

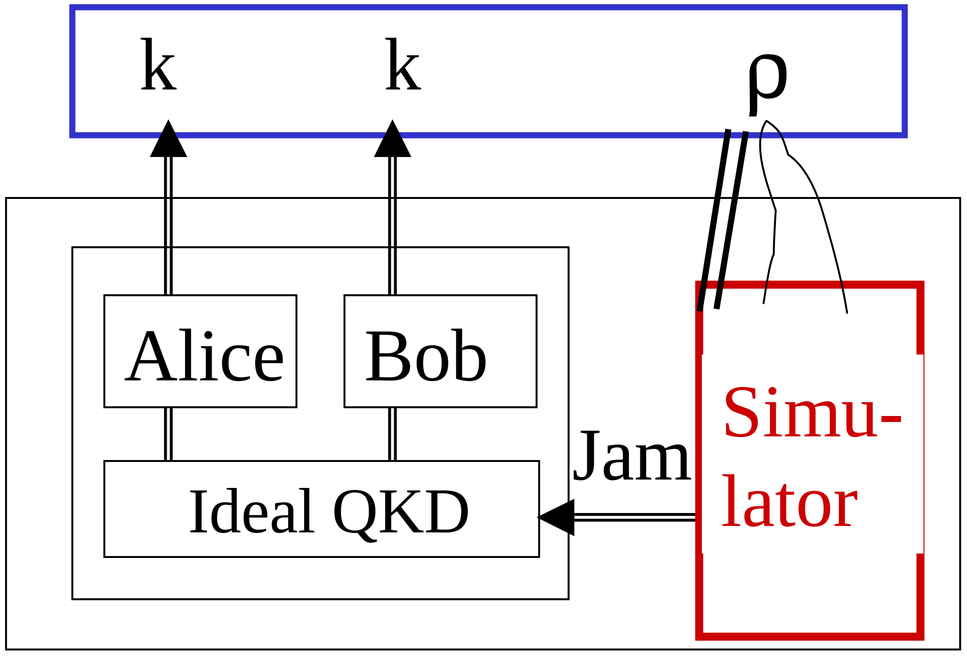
- What formal general security definition (for the application and the subprotocols) is suggested by this intuition?

Security of QKD in terms of Simulators and Environments



Real

$$k \in \{0,1\}^m \cup \{\text{fail}\}$$

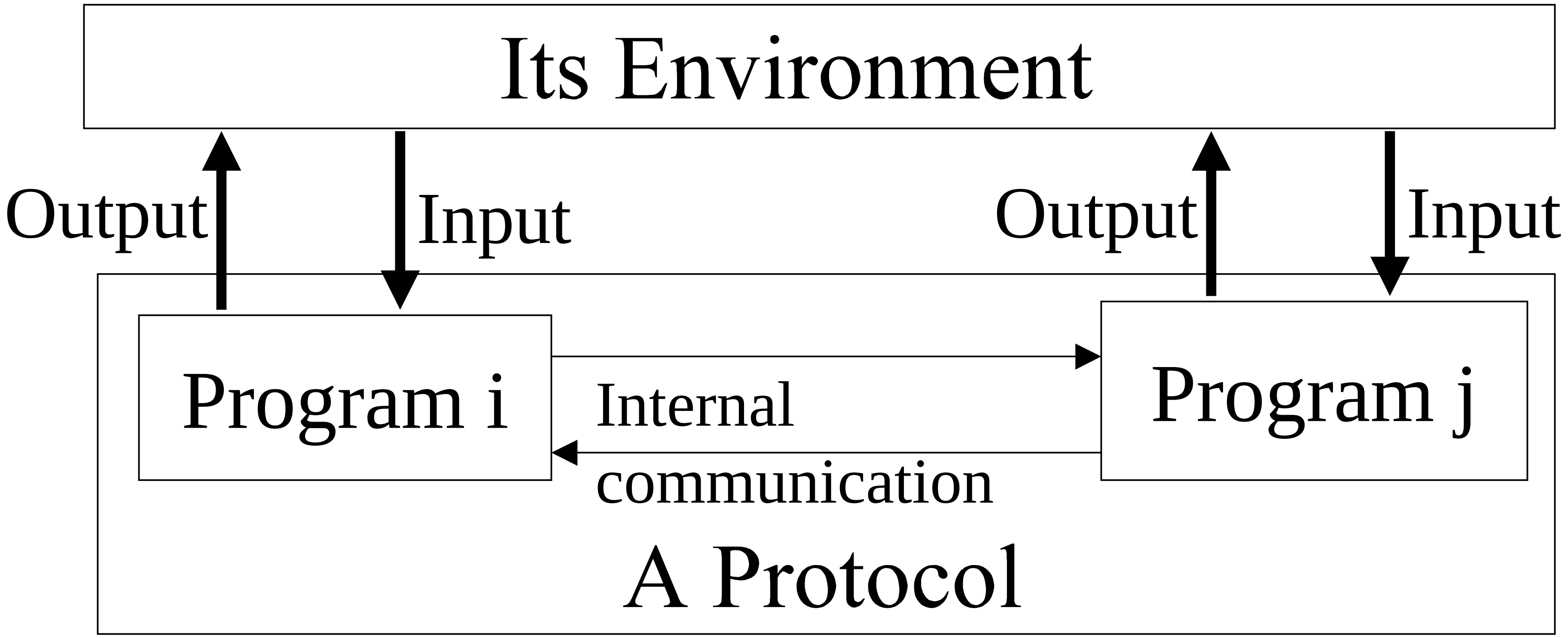


Ideal

$$\text{Jam} = 1 \Rightarrow k = \text{fail}$$

The models

A Protocol and its Environment



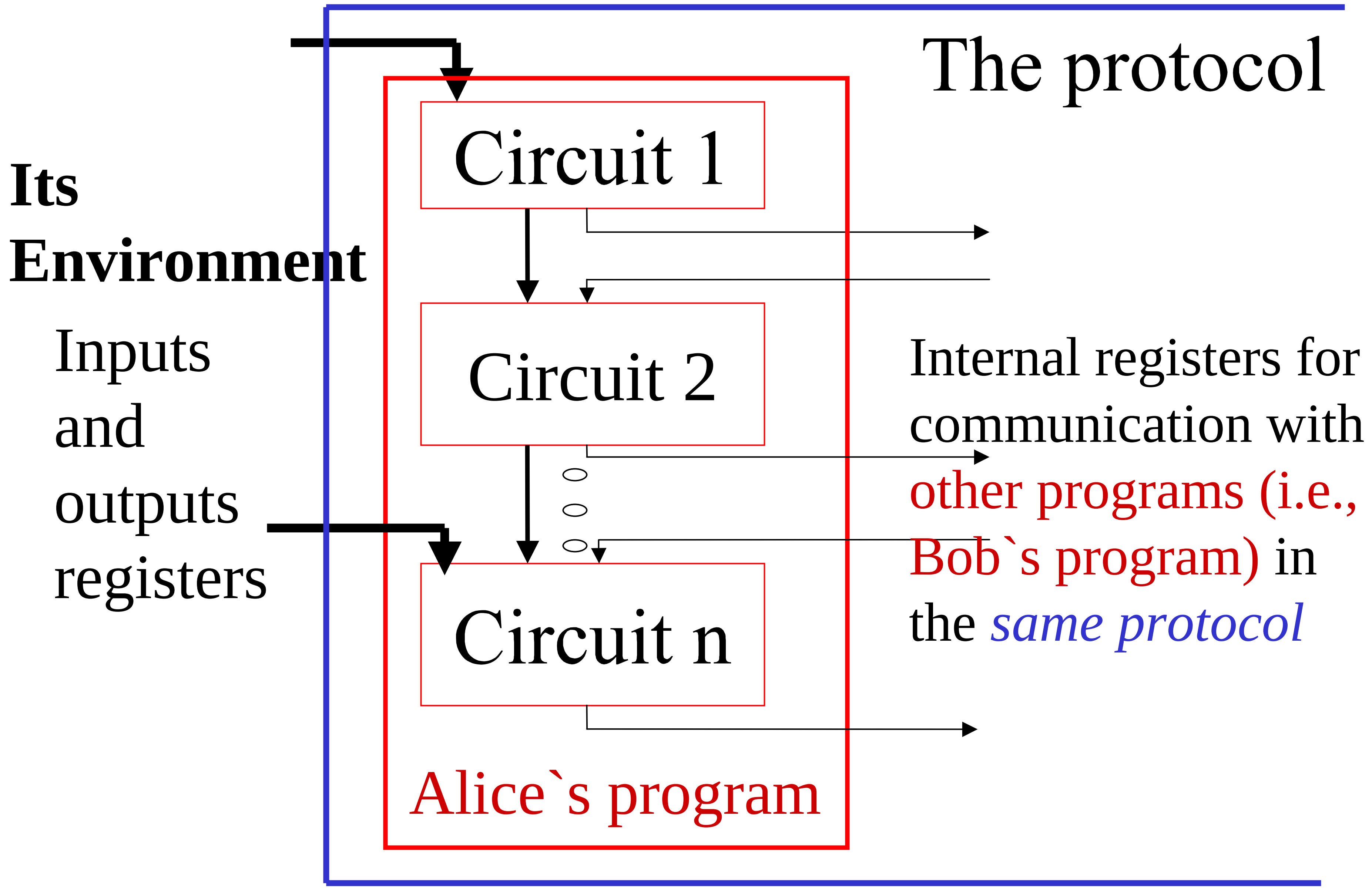
A quantum protocol is a collection of circuits regrouped in disjoint sets called programs together with channels for internal communication and for communication with the environment.

Functional and Internal Layers

The communication structure of a protocol determines two layers: the functional and the internal layers. The *functional layer* is defined by the relationship between the input and output registers of the protocol.

The *internal layer* is defined by the details of the program (the circuits) and the internal communication in the protocol. It`s the means by which the functionality layer is realised.

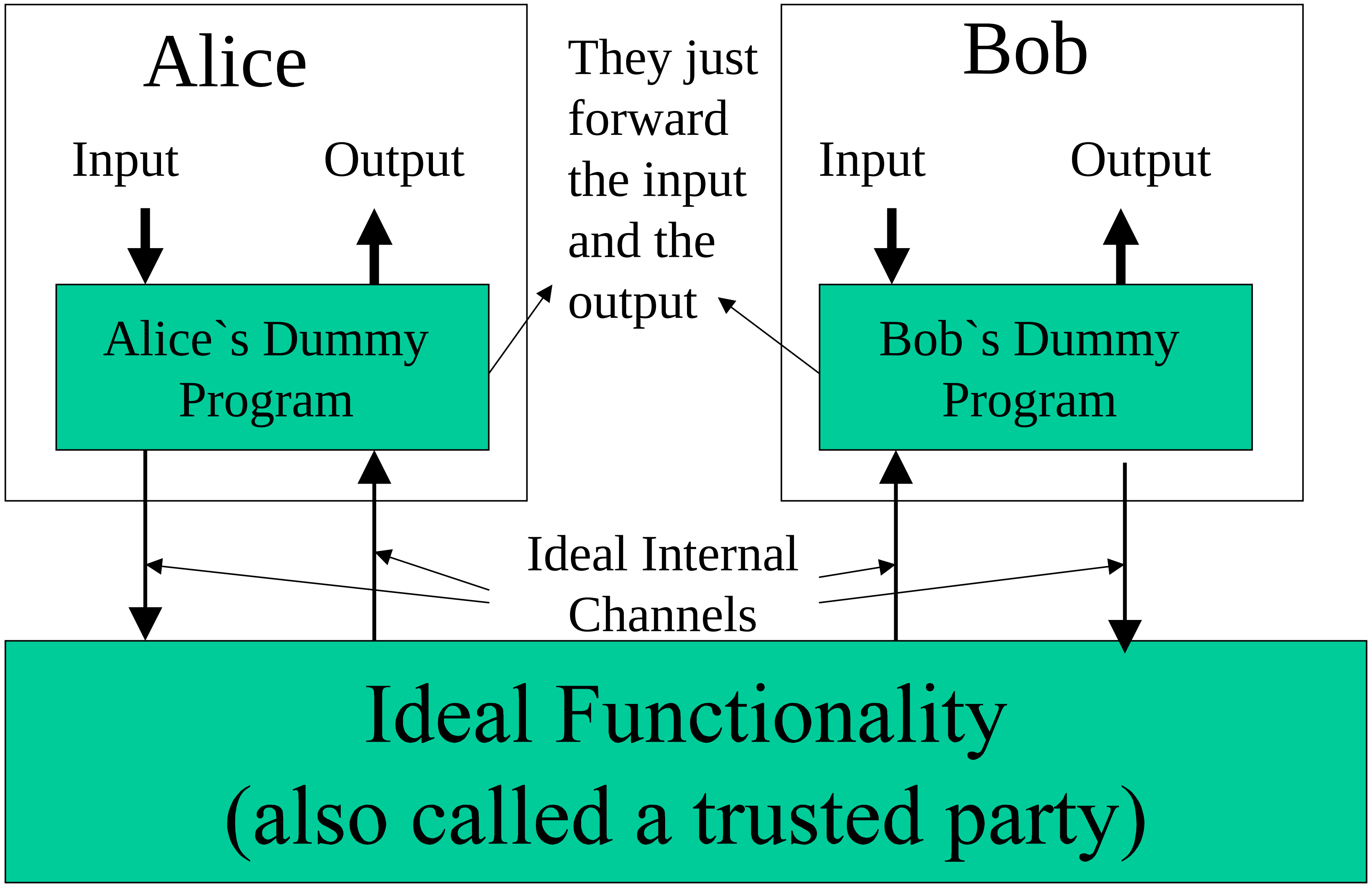
A Program in its context


The protocol
Circuit 1
Circuit 2
Circuit n
Its Environment
Inputs and outputs registers
Internal registers for communication with other programs (i.e., Bob`s program) in the *same protocol*
Alice`s program

Some Remarks

- Registers are sent through communication channels that respect assumptions (e.g., private or not, authenticated or not, etc.) We consider that these channels and assumptions are part of the definition of the protocol.
- The definition of the protocol also includes the access rule (e.g. how many participants can be corrupted).
- Circuits are automatically activated when all the required registers are received.
- All internal channels (between the programs) pass through the adversary
- Every program runs at a different location.

Format of an ideal protocol



A Trade off

A more secure ideal protocol provides a stronger (more secure) definition of security. The strongest definition will state that no party can be corrupted in the ideal protocol and it will use ideal channels with guarantee of delivery, etc. The problem, of course, is to find protocols that achieve this level of security.   A trade off is neccessary.

Example: the ideal internal channels offer no guarantee of delivery because the real channels can be jammed.

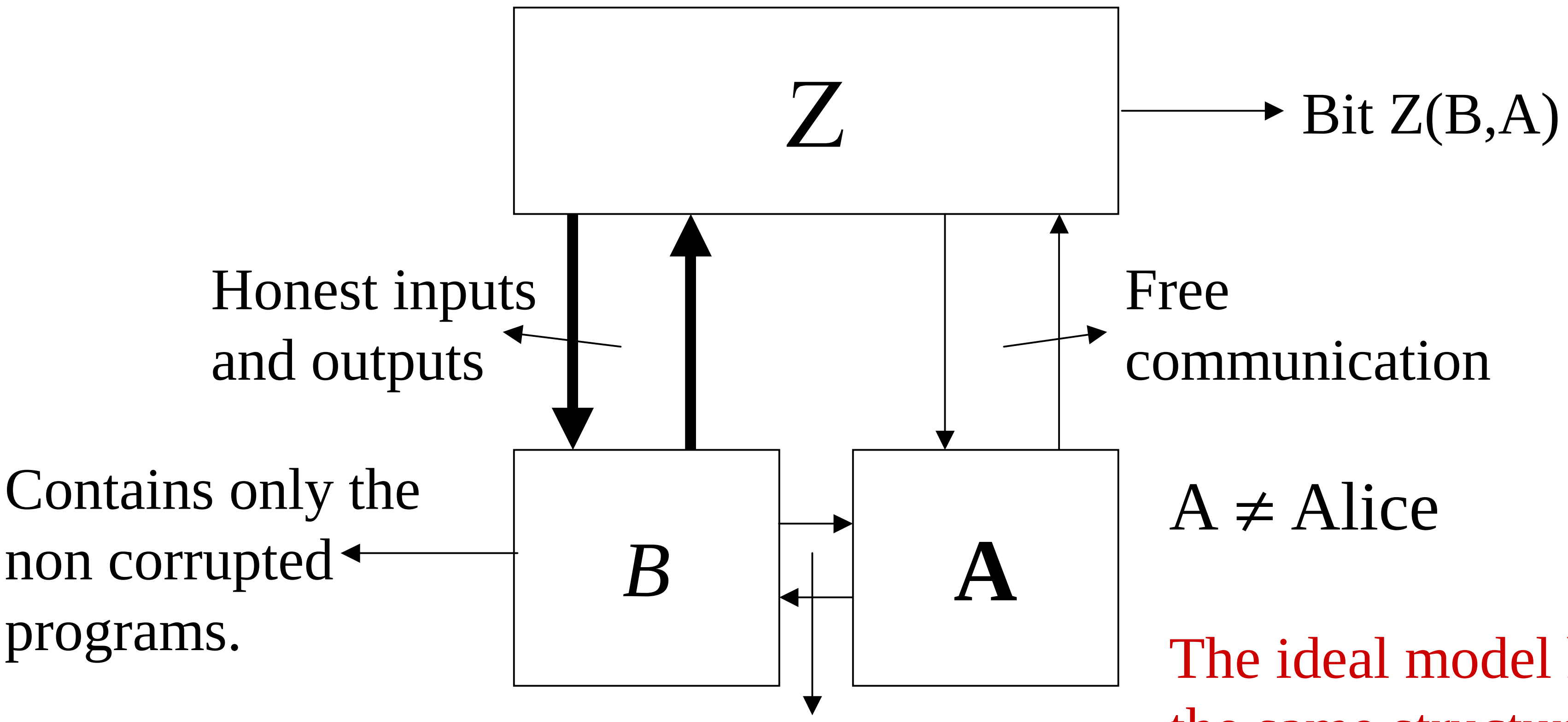
Ideal Protocol: Internal and functionality layers.

An ideal protocol is simply a special case of a real protocol.

The functionality layer is defined by the relationship between the inputs and the outputs of the dummy parties. This relationship is in turn determined by the internal layer which is the ideal functionality β and the communication between the dummy parties and β .

The General Security Definition

The overall Model


Z
Bit Z(B,A)
Honest inputs and outputs
Free communication
Contains only the non corrupted programs.
B
A

A ≠ Alice

The adversary **A** substitutes itself for the honest programs. Moreover, all internal communications in *B* pass through the adversary A.

The ideal model has the same structure, but *B* is replaced by β and **A** by **S**.

The worst real adversary

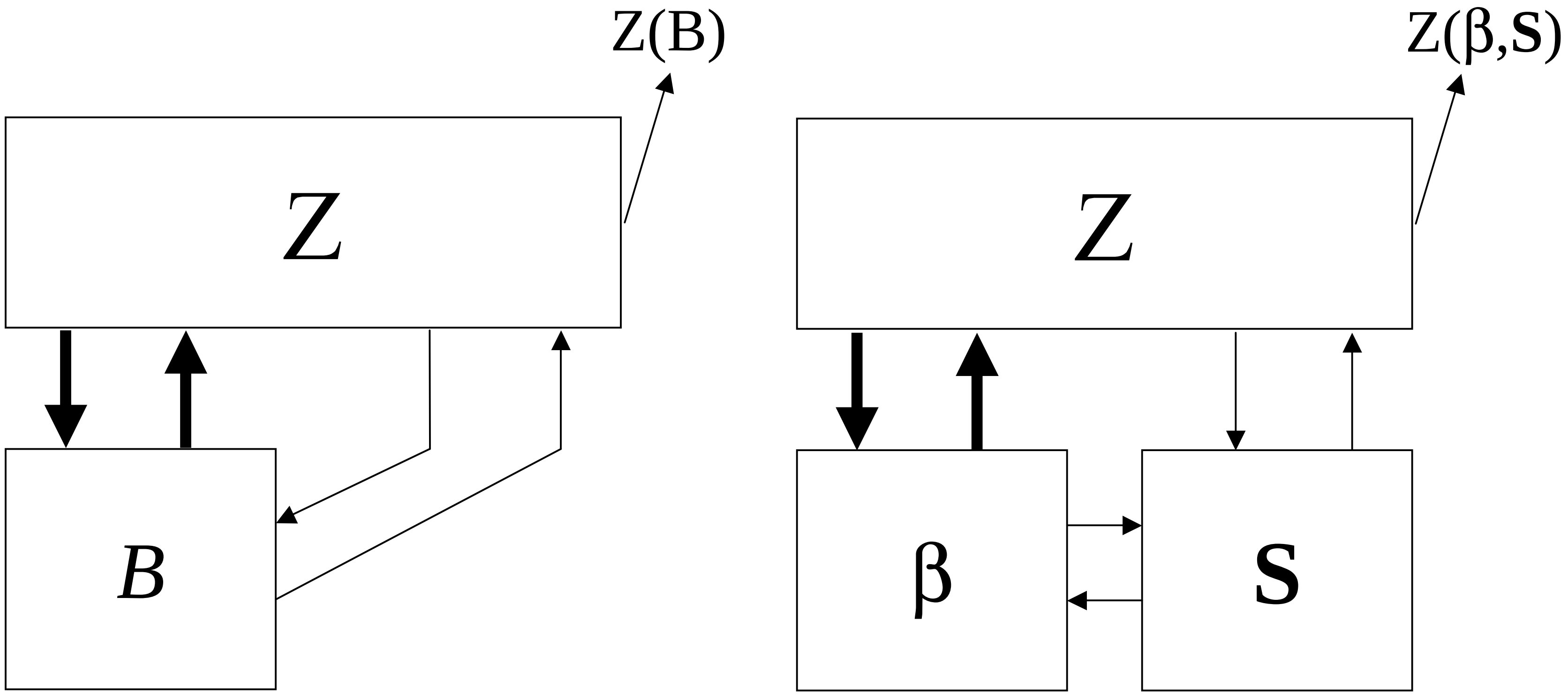
The worst case real adversary (to challenge the simulator **S**) is the dummy adversary which simply follows any request from the environment.

This can be proven, but this will only be useful to provide an intuition behind our definition. A definition does not require a proof.

The main point is that we can assume that the real adversary is part of the environment. The real adversary disappears from the picture!

Universal Security Definition

Basic Idea



For all environment Z, there must exist a simulator **S** such that Z(B) ≈ Z(β,**S**)

Quantum Universal Security Definition

For any two random binary variables Y, Y` let us write

$$Y \approx_e Y` \quad \text{if} \quad |\Pr( Y = 0 ) - \Pr( Y` = 0)| \leq e.$$

Let P be the set of all polynomial functions.

Definition. A protocol B for an ideal functionality β is secure, if for any environment Z there exists a simulator S such that $(\forall d \in P)\ (\exists\, n_0 \in \aleph)\ (\forall\, n > n_0)$

$$Z(B) \approx_e Z(\beta, S)$$

where e = 1/d(n).

Quantum Universal Security Definition

About the Computational Setting

The simulator S must have a polynomial complexity $c \in P$ that depends only on B (i.e. not on Z or n_0). Also, n_0 can only depend on d and on the respective polynomial complexity c, c\` of S and Z (not on their actual circuits). The actual circuit of S can depend on n.

For every $c \in P$, let T(c) be the set of programs of complexity c. Formally, the order for the quantifiers is:

$(\exists\, c \in \mathbf{P})(\forall c' \in \mathbf{P})(\forall d \in \mathbf{P})\, (\exists\, n_0 \in \aleph)(\forall\, n > n_0)$
$(\forall Z \in T(c'))(\exists\, S \in T(c))$

$$Z(B) \approx_e Z(\beta, S)$$

where $e = 1/d(n)$.

Proof of part (a) of the composabity theorem

There are three steps in the proof.

(1) Essentially, we must construct a simulator S(G(B)) for G(B) given the simulators S(G(β)) for G(β) and S(B) for B.

(2) We must show that the size of the simulator S(G(B)) is a polynome that depends only on the protocols G(β) and B.

(3) We must show that the lower bound n_0 for n depends only on the polynome d (for the indistinguishability) and on the complexity of the circuits Z and S, not on the actual circuits.

All in terms of a single bit

All the security properties of the protocol considered are encapsulated into the non distinguishability of a single bit returned by the environment. For example, both privacy and uniformity are included in the case of QKD.

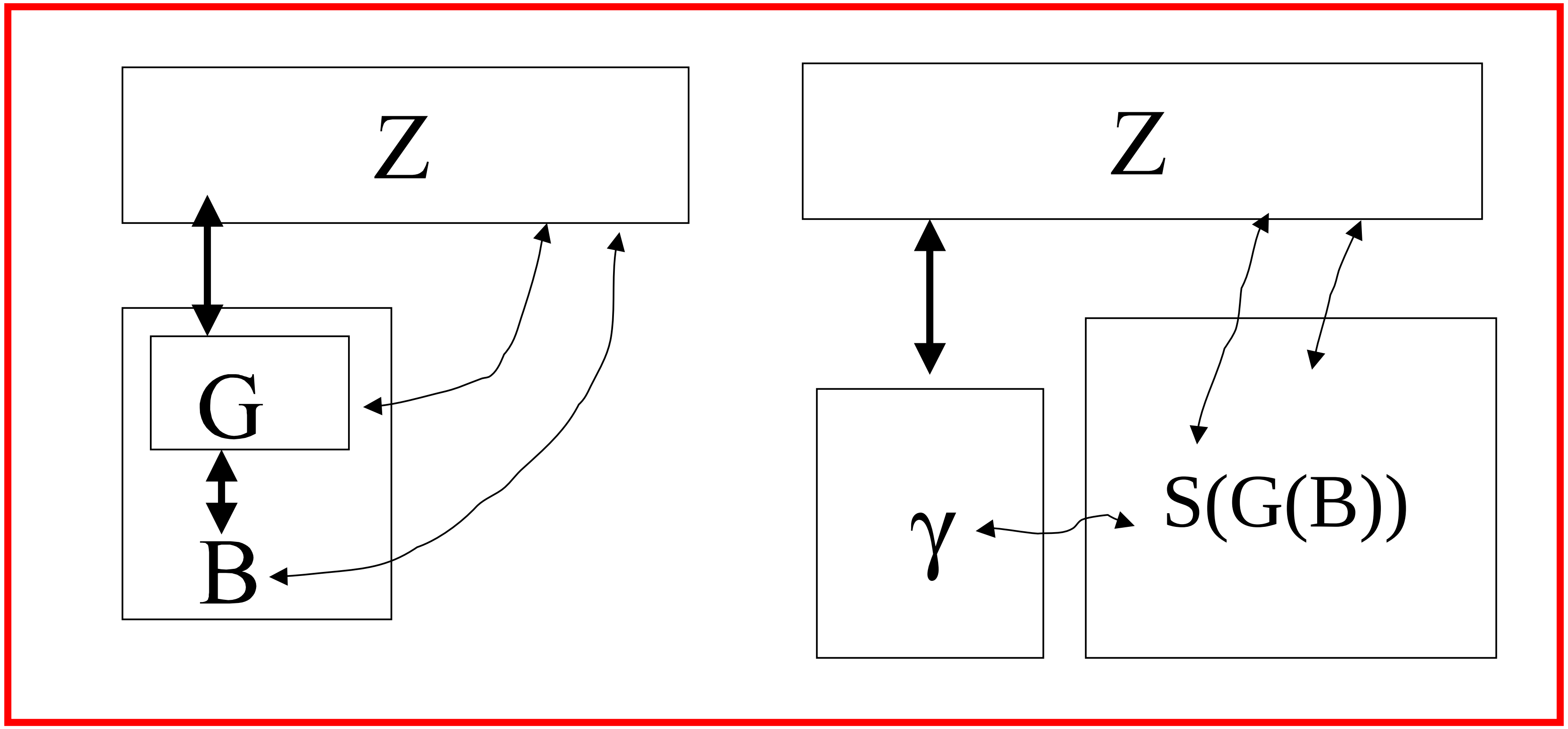
This approach can be used to define the security of any protocol and this is very powerful.

The Composability Theorem

How does it work?

Recall that we want to prove

$$B \text{ s.r. } \beta \wedge G(\beta) \text{ s.r. } \gamma \Rightarrow G(B) \text{ s.r. } \gamma$$



s.r. = securely realises

The role of the environment

The subcircuit G of the protocol G(B) is a part of the environment for B.

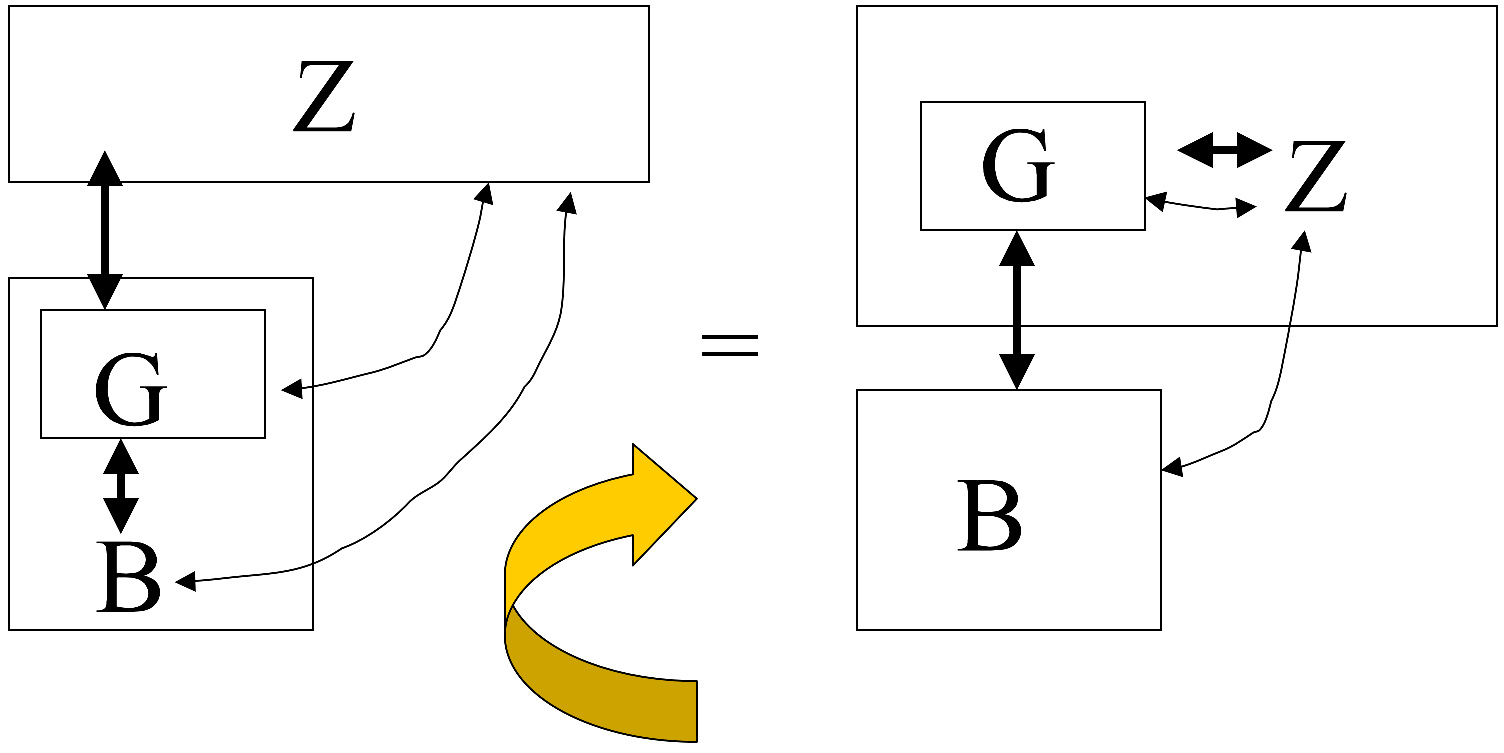


Diagram 1 Diagram 2

So, we can use the security of B,

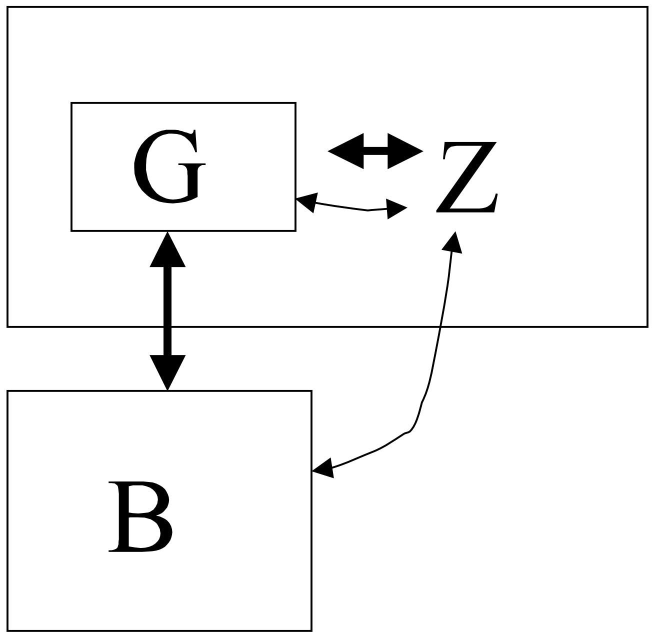


Diagram 2

≈

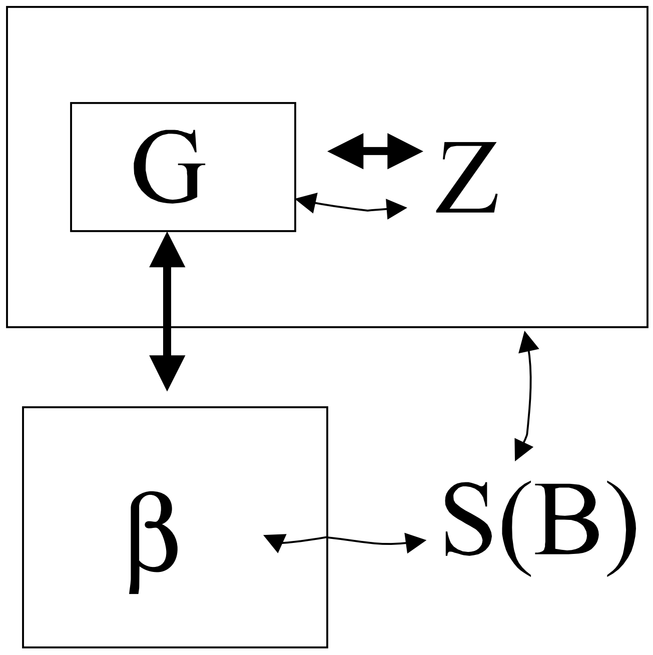


Diagram 3

and take back G from the environment

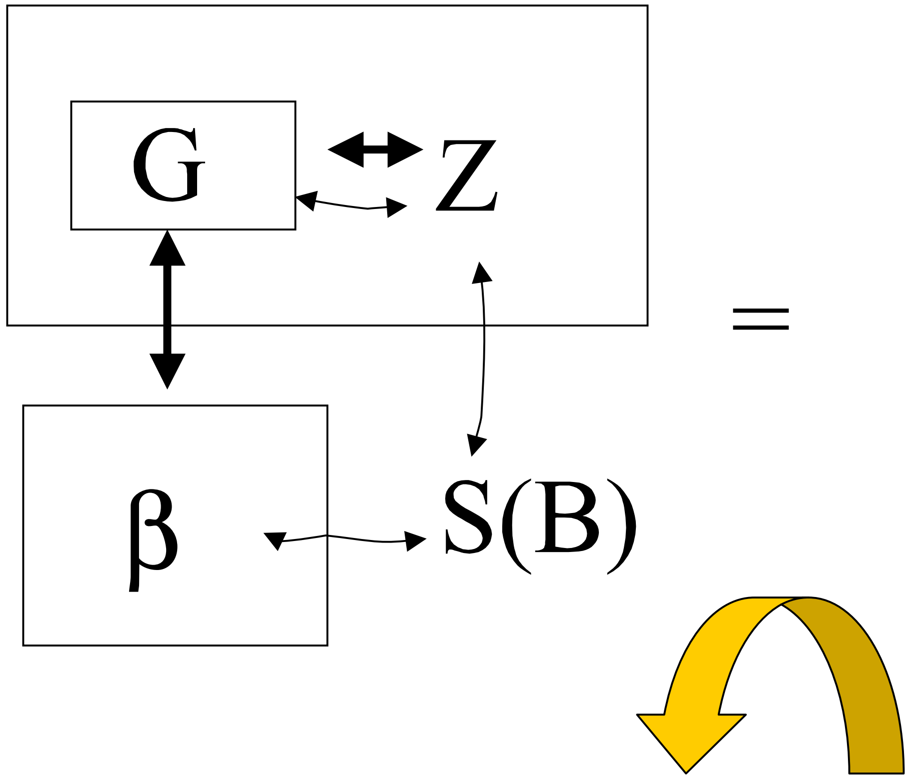


Diagram 3

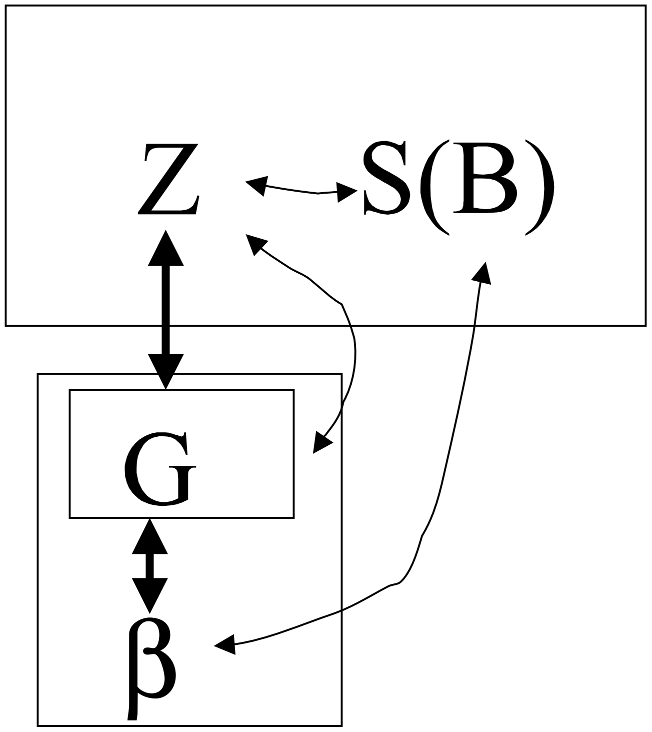


Diagram 4

and, finally, use the security of G(β).

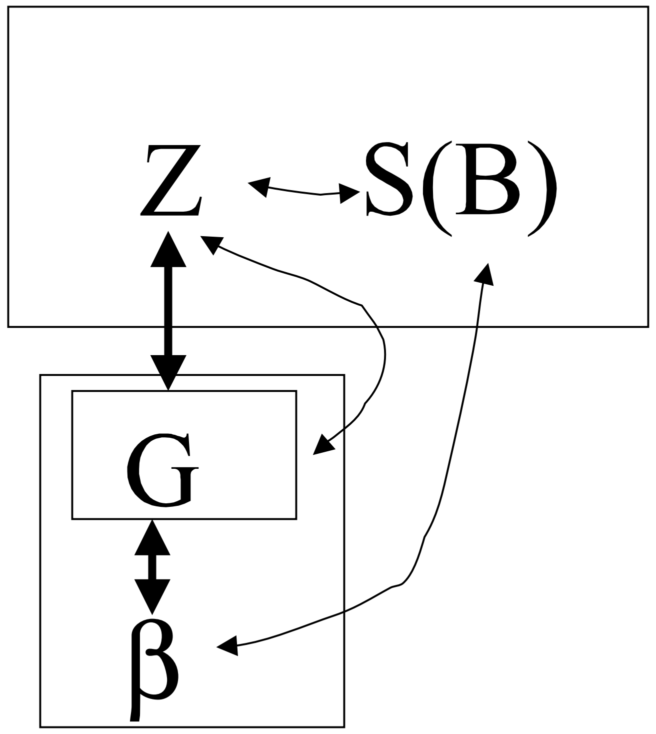


Diagram 4

$\approx$

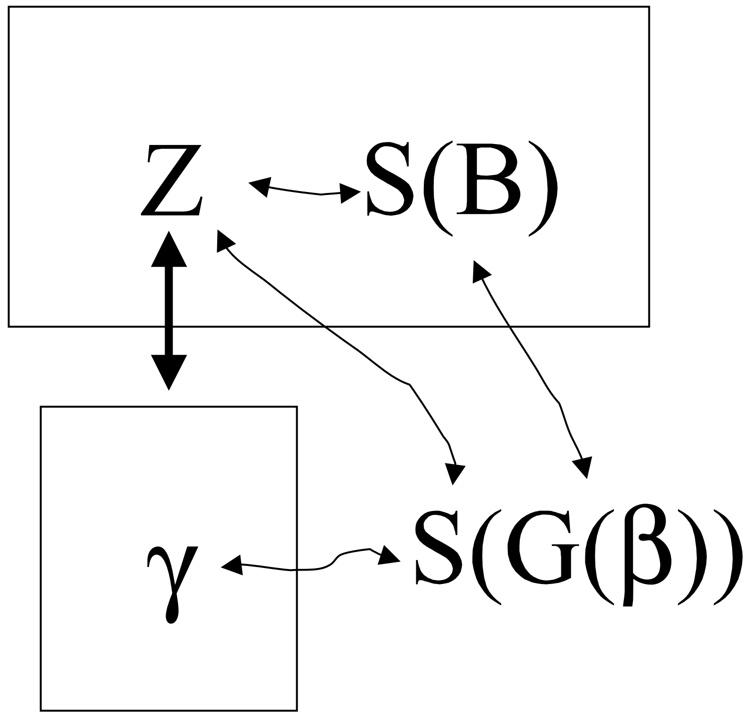
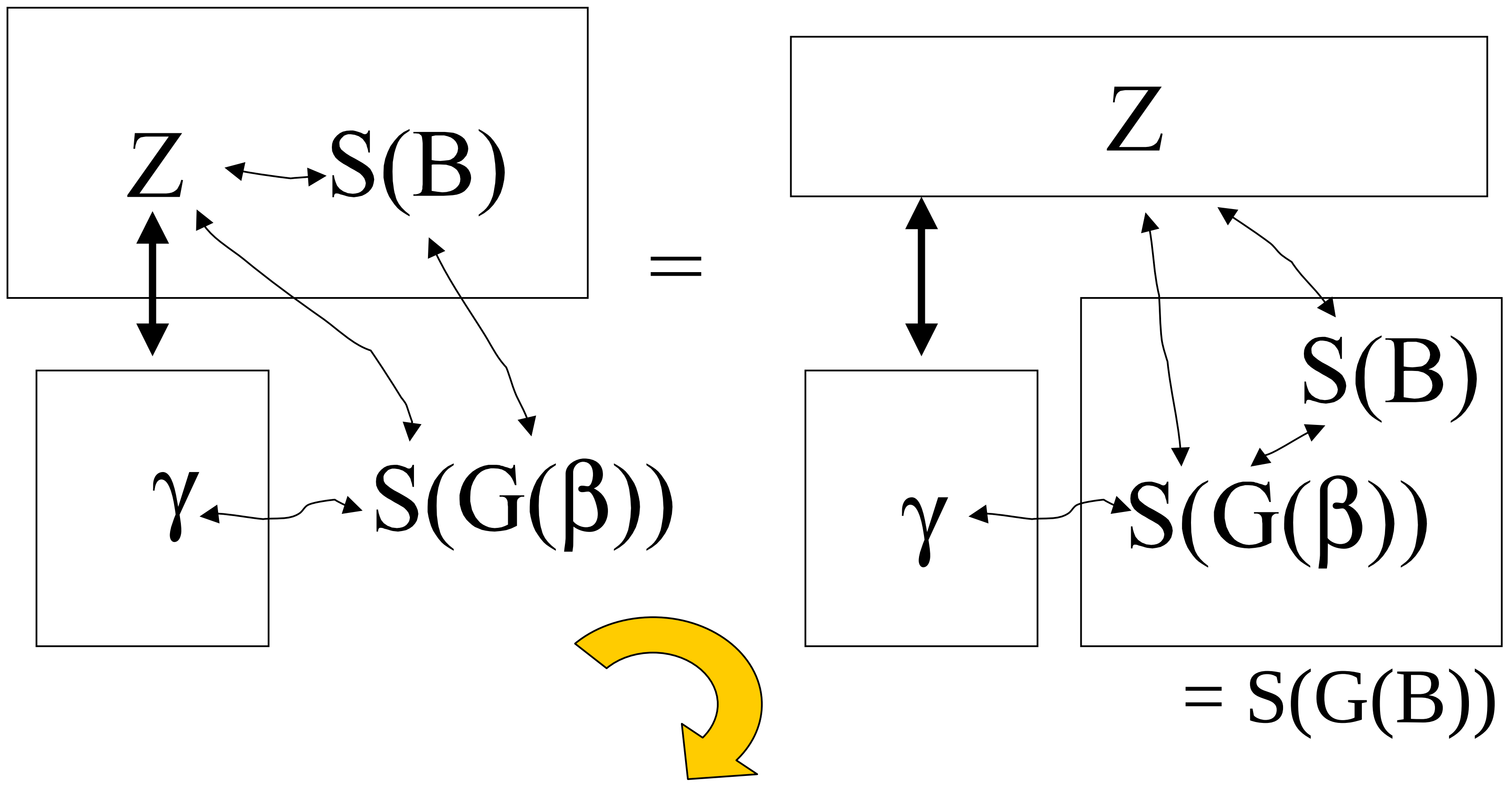


Diagram 5

So we have a simulator for G(B)



Part II: Composability of QKD Joint Work (in progress) with

Michael Ben-Or, Michal Horodecki, Debbie Leung and Jonathan Oppenheim

The « standard » QKD criteria

$$\text{Uniformity: } \Pr(\text{jam}=0)(m - H(k \mid \text{jam}=0)) \le \alpha$$

$$\text{Privacy: } I(k;Y \mid \text{jam}) \le \alpha$$

Y = Eve`s optimal measurement outcome.

Are Uniformity and Privacy
enough for composability?

Recall: Composability means there exists a simulator so that the ideal case is indistinguishable from the real case.

Using Uniformity…

we obtain that

key k with probability 2^{-m} (ideal) and
key k with probability $p(k \mid jam = 0)$ (near ideal)

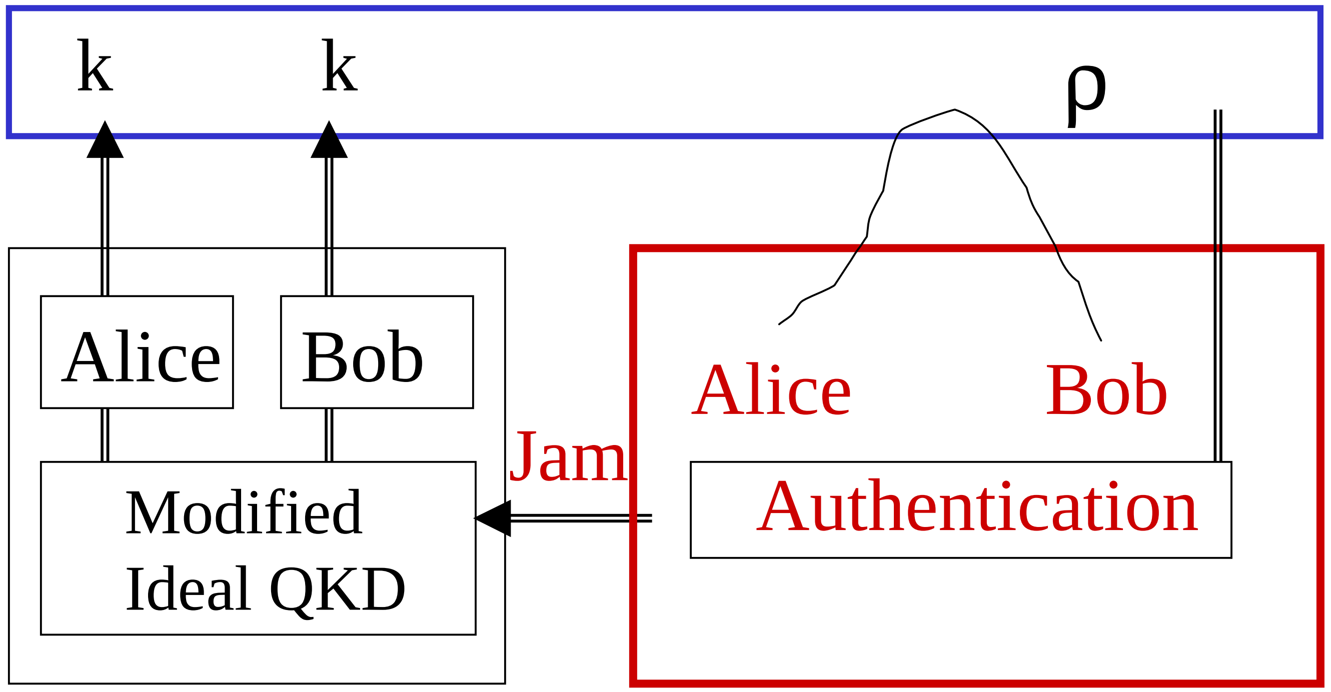
are indistinguishable.

So, it is enough to show that the real QKD protocol and a near Ideal QKD protocol which outputs k with probability $p(k \mid jam = 0)$ are indistinguishable.

Recall: Composability means there exists a simulator so that the ideal case is indistinguishable from the real case.

The simulator

$k \in \{0,1\}^m \cup \{\text{fail}\}$



The simulator runs the real QKD protocol while interacting with the environment. This produces ρ. If the simulated key k^* = fail, it sets jam = 1. In all cases, it throws away the simulated k^*.

Using Privacy

We can show that

$$SD(\hat{\rho}_0, \hat{\rho}_1) \leq 2^m \mathbf{I}_{acc}(E \,|\, \mathrm{Jam}) \leq 2^m \alpha$$

where $I_{acc}(E| \mathrm{Jam}) = \max I(k;Y| \mathrm{Jam})$ and m is the length of the key. (We omit the proof here).

The large factor 2^m looks bad, but actually it is not so bad because the bound α on I_{acc} respects

$$\alpha \leq 2^{-\xi n}$$

where n can be taken arbitrarily large, independently of m.

What about the known QKD protocols

- Mayers and Shor-Preskill security proofs can be adapted for composability without the large factor 2^m.

- We do not know if B92 is composable without this large factor (since there is no security proof).

Open Questions

- The composability of QKD is useful in application protocols that also respect this universal security definition. Multiparty secure computation and other tasks respect this universal security definition. (Not yet formally proven).
- The large factor 2^m might not be so bad, but still it makes a difference. We have no reason to believe that it is neccessary!
- Obtain a more flexible « universal » security definition because the one we have now is not so easy to achieve. We already have one proposal.

History of this security definition.

Simulator that interacts with an ideal protocol for specific protocols (1985).

Universal composability came after Goldreich, Micali, Wigderson in 1987.

The idea of using the environment with an output bit as a distinguisher was introduced in 2000 (Canetti). This takes care of concurrent composability.

Proved for quantum protocols in 2002 (Ben-Or, Mayers). At the same time, a modified and simpler (but perhaps equivalent) model was proposed which allowed to extend the result. This is the model used here.

Computational Setting Issues (I)

The simulator for Π^F and ρ have polynomial size $c(\Pi^F)$ and $c(\rho)$, respectively. The simulator for Π^ρ which we will construct consists of a constant size manager that forwards the requests from the environment to one of these two simulators. So, it has polynomial size $c \in \mathbb{P}$.

Consider any $c``, d \in \mathbb{P}$. We pick for $k_0$ the maximum of $k_0(\Pi^F, c, c`` + c(\rho), 2d)$ and $k_0(\rho, c, c`` + |\Pi|, 2d)$.

Let $k > k_0$ and $Z \in T(c``(k))$. We need to find $S \in T(c(k))$ such that $Z(\Pi^\rho) \approx_{d(k)} Z(G^S)$. We have $|Z| = c``$ and $|\Pi| = c$. So $|Z + \Pi| = c`` + |\Pi|$ is a polynome. The definition of k_0 and the security of ρ gives us $S(\rho)$ such that

$$[Z + \Pi](\rho) \approx_{1/(2d(k))} [Z + \Pi](F^{S(\rho)}) \qquad (1)$$

Computational Setting Issues (II)

We also have $| Z + S(\rho) | = c`` + c(\rho)$. So the security of Π^F and the definition of k_0 guarantees the existence of a simulator $S(\Pi^F)$ such that

$$[Z + S(\rho)](\Pi^F) \approx_{1/(2d(k))} [Z + S(\rho)](G^{S(\Pi^F)}) \qquad (2)$$

The environment Z (i.e., the dummy adversary Ã) does not see the (honest) I/O communication between Π and ρ. It sees Π and ρ as if they were two independent protocols. Note that the simulator $S(\Pi^F)$ provides the I/O interface to F that is needed by $S(\rho)$. This is possible because $S(\Pi^F)$ corrupts the dummy parties of F as requested by the environment Z and allowed by the access rule of F. So, the simulator $S(\rho)$ interacts with $S(\Pi^F)$ to access F.