Lecture 13

Secure Multi-Party Computation Protocols

Stefan Dziembowski

www.crypto.edu.pl/Dziembowski

University of Warsaw

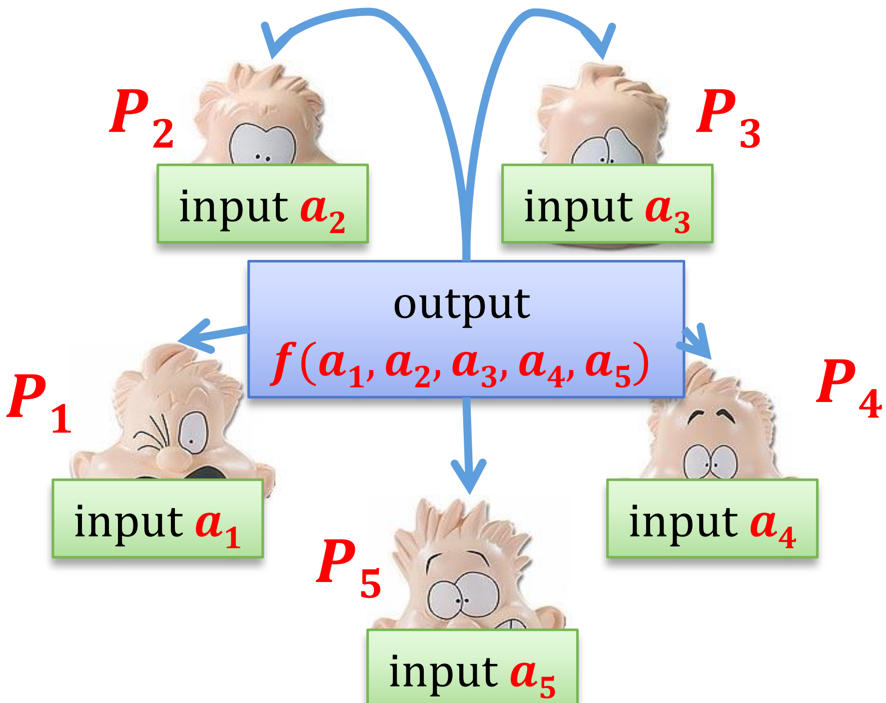
9.01.19 **version 1.0**

Plan

1. Definitions and motivation
2. Security against the threshold adversaries
    1. overview of the results
    2. overview of the constructions
3. General adversary structures
4. Applications

Multi-party computations (MPC)

a group of parties:



they want to compute some value

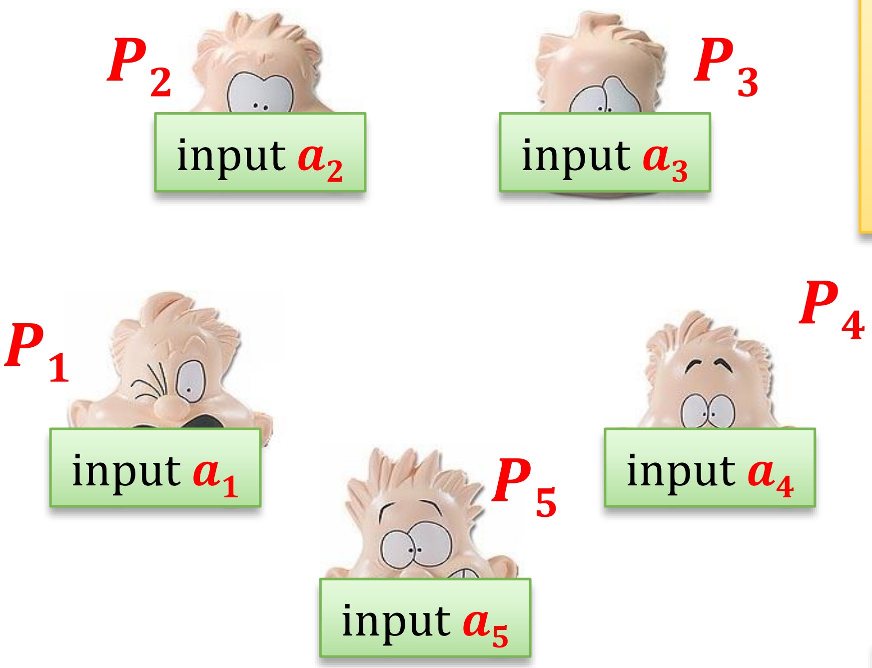
$$f(a_1, a_2, a_3, a_4, a_5)$$

for a publicly-known f.

Before we considered this problem for $n = 2$ parties.

Now, we are interested in arbitrary groups of n parties.

Examples

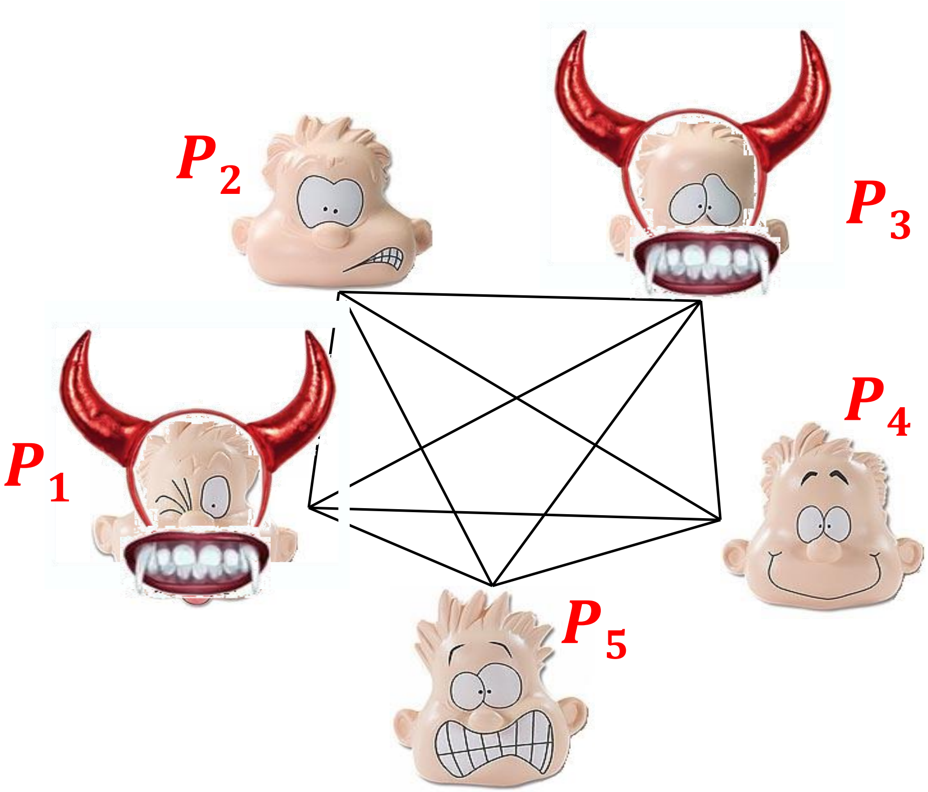


A group of millionaires wants to compute how much money they own **together**.

$$f(a_1, a_2, a_3, a_4, a_5) = a_1 + a_2 + a_3 + a_4 + a_5$$

Another example: **voting**

The general settings



Each pair of parties is connected by a **secure channel**.

(assume also that the **network is synchronous**)

Some parties may be **corrupted**.

The corrupted parties may **act in coalition**.

How to model the coalitions of the corrupted parties?

We assume that there exists one adversary that can **corrupt** several parties.

Once a parity is corrupted the adversary "takes control over it".

what it means depends on the settings

Threshold adversaries

In the **two-party case** we considered an adversary that could corrupt one of the players.

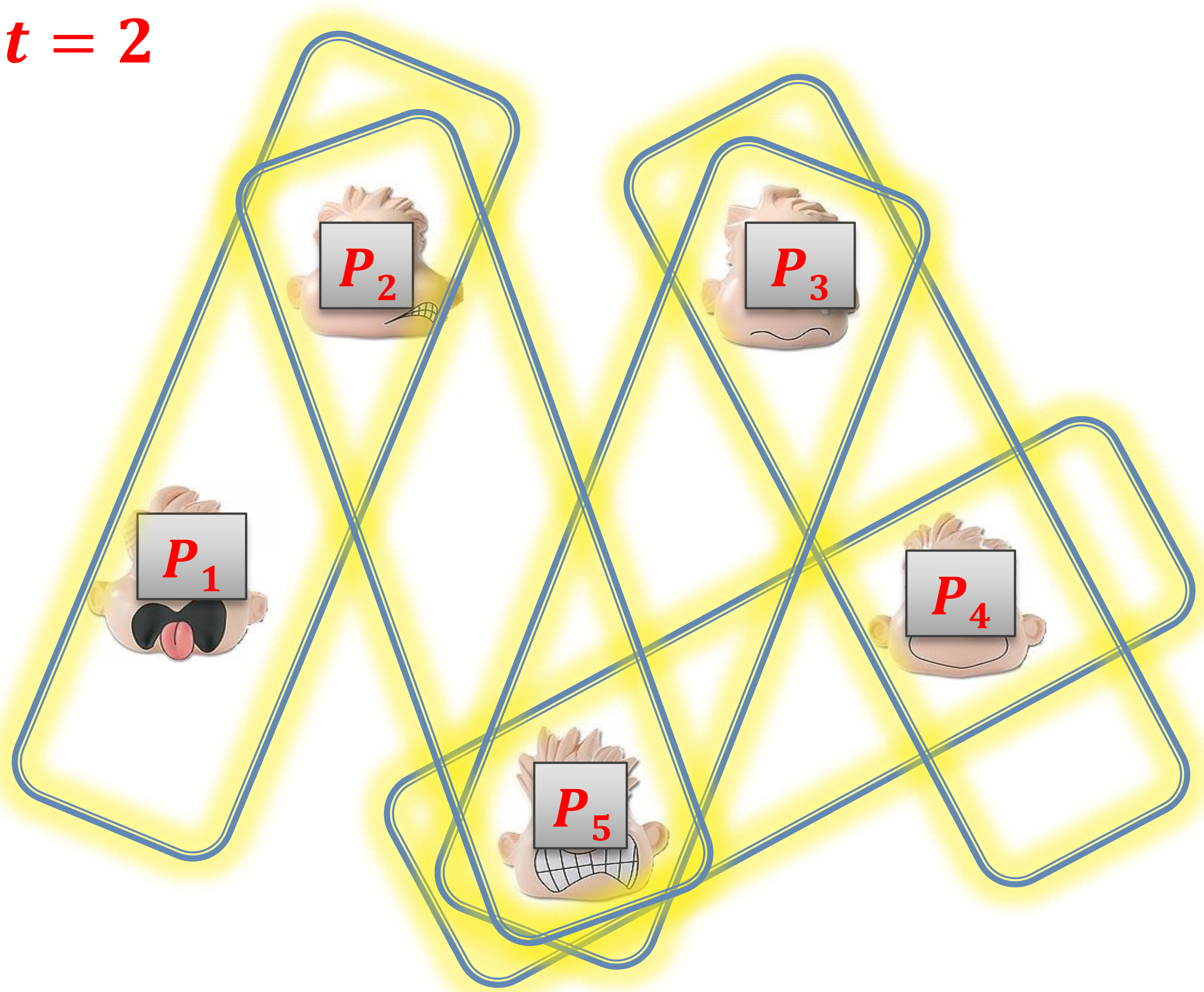
Now, we assume that the adversary can corrupt **some subset of the players**.

The simplest case:

set some threshold $t < n$ and allow the adversary to corrupt **up to t players**.

Example

$n = 5, t = 2$



Types of adversaries

As before, the adversary can be:

- **computationally bounded**, or
- **infinitely powerful**,

and

- **passive**
- **active**

These choices are orthogonal!

	computationally bounded	**infinitely powerful**
passive		
active		

all those choices make sense!

Adaptivness

In addition to it the adversary may be

- **adaptive** – he may decide whom to corrupt **during the execution of the protocol**, or
- **non-adaptive** – he has to decide whom to corrupt, **before the execution starts**.

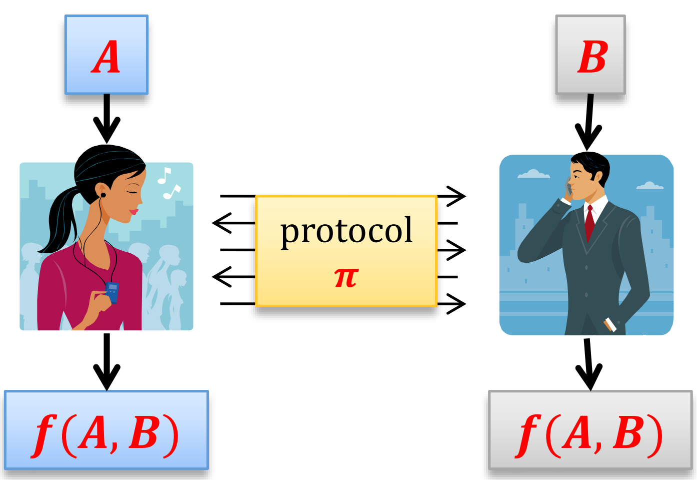
The security definition

The security definition is complicated and we do not present it here.

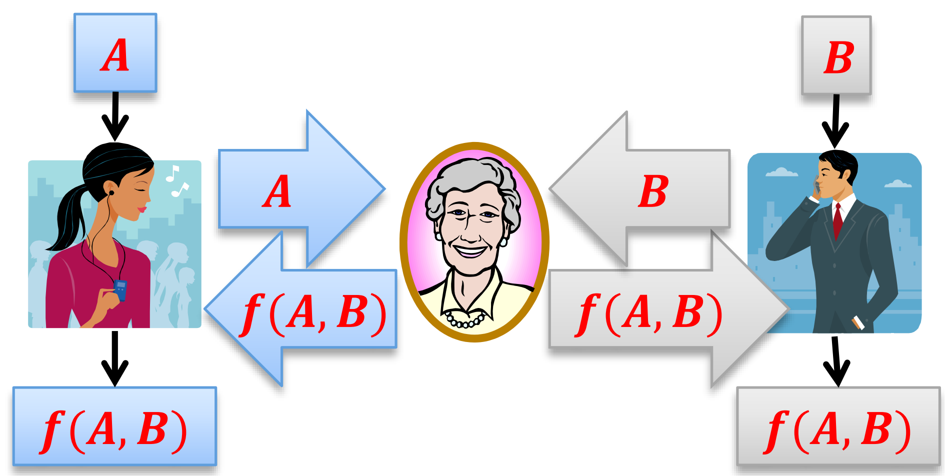
Main intuition: the adversary should not be able to do more damage in the **"real" scenario** than he can in the **"ideal" scenario**.

Remember the **two-party case**?

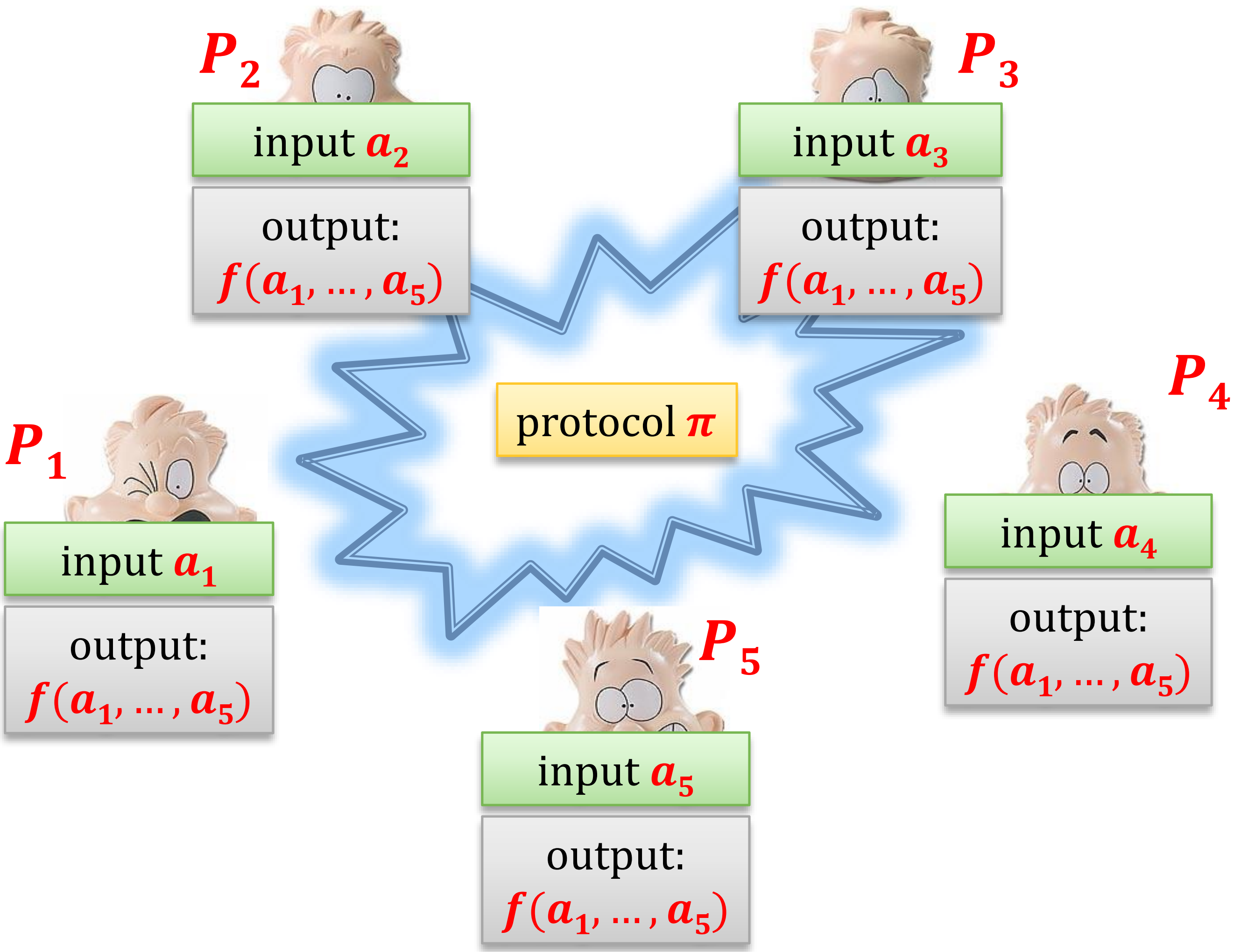
"real" scenario



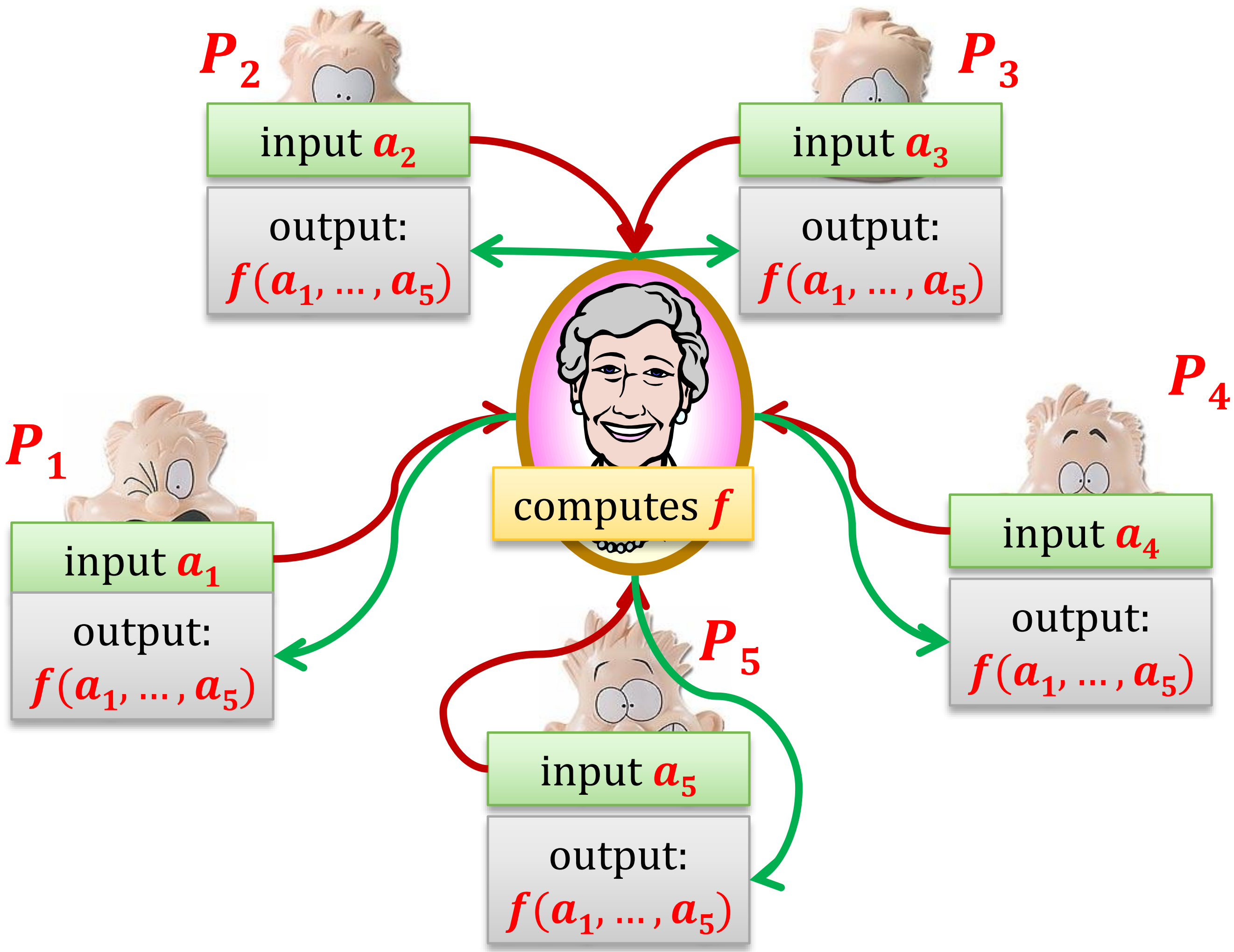
"ideal" scenario



The “real scenario”



The "ideal" scenario



Plan

1. Definitions and motivation
2. Security against the threshold adversaries
   1. overview of the results
   2. overview of the constructions
3. General adversary structures
4. Applications

Classical results

Question:

For which values of the parameter t multi-party computations are possible (for every poly-time computable function f)?

n – the number of players

setting	adversary type	conditio n
computational	passive	$t < n$
computational	active	$t < n/2$
information-theoretic	passive	$t < n/2$
information-theoretic	active	$t < n/3$

this can be improved to
$t < n$
if we give up "fairness"

(these are tight bounds)

this can be improved to
$t < n/2$
if we add a "broadcast channel"

(Turns out that the adaptivness doesn't matter)

Example of a lower bound

information-theoretic, passive: $\boldsymbol{t < n/2}$

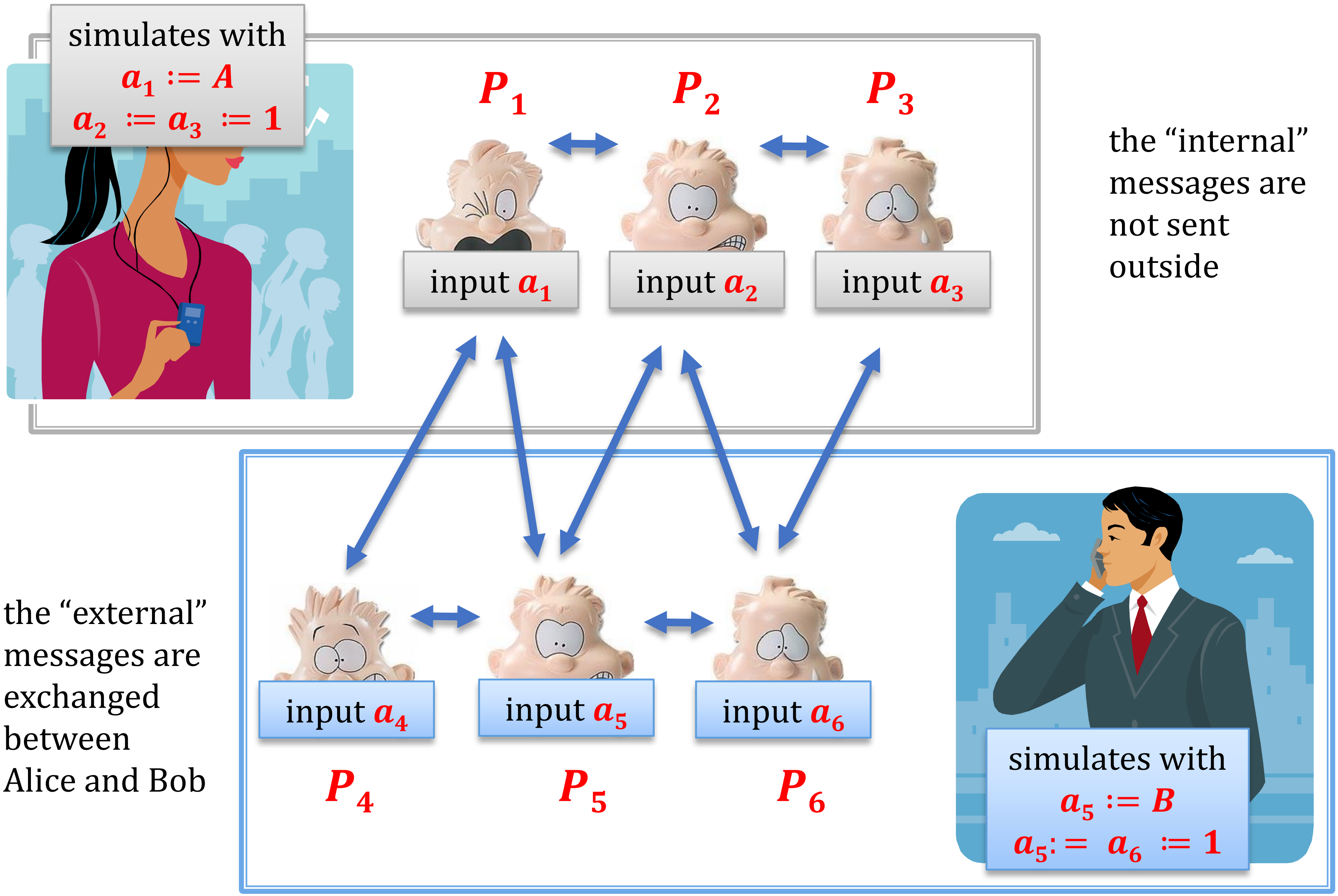
Suppose $\boldsymbol{n = 6}$ and $\boldsymbol{t = 3}$

Suppose we have a protocol for computing
$\boldsymbol{f(a_1, a_2, a_3, a_4, a_5, a_6) = a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge a_6}$

We show an information-theoretically secure 2-party protocol for computing

$$\boldsymbol{F(A, B) = A \wedge B}$$

After showing this we will be done, since we know it's impossible!



simulates with
$a_1 := A$
$a_2 := a_3 := 1$
P_1
P_2
P_3
input a_1
input a_2
input a_3
the “internal” messages are not sent outside
the “external” messages are exchanged between Alice and Bob
input a_4
input a_5
input a_6
P_4
P_5
P_6
simulates with
$a_5 := B$
$a_5 := a_6 := 1$

Correctness?

At the end of the execution of the simulated protocol **Alice** and **Bob** know

$$\boldsymbol{f(A, 1, 1, B, 1, 1) = A \wedge B}$$

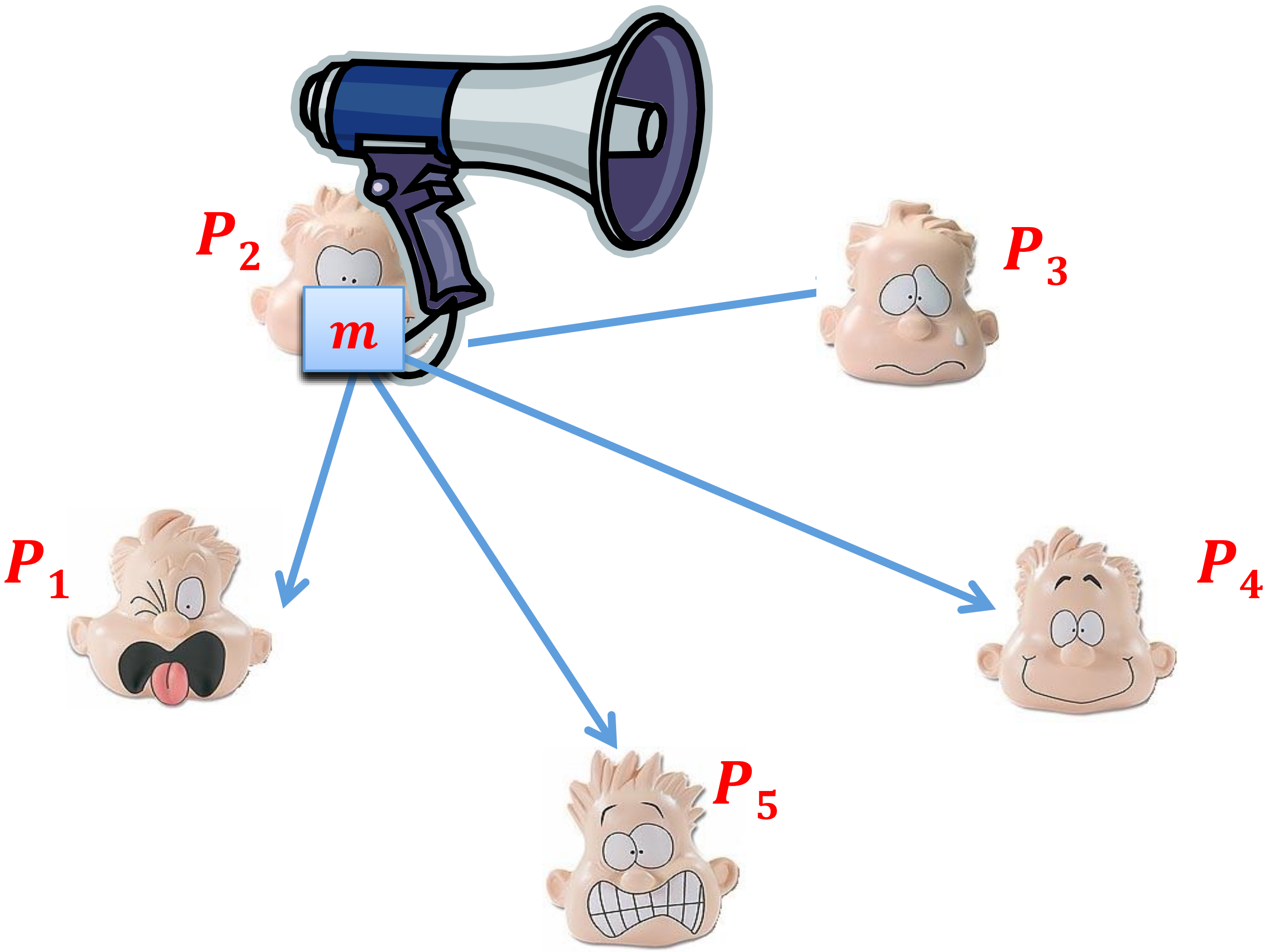
So they have computed **F**.

Why is this protocol secure?

If the adversary corrupted **Alice** or **Bob** then he "corrupted" exactly $t = 3$ parties.

From the security of the **MPC** protocol the "new" **2**-party protocol is also secure!

A broadcast channel

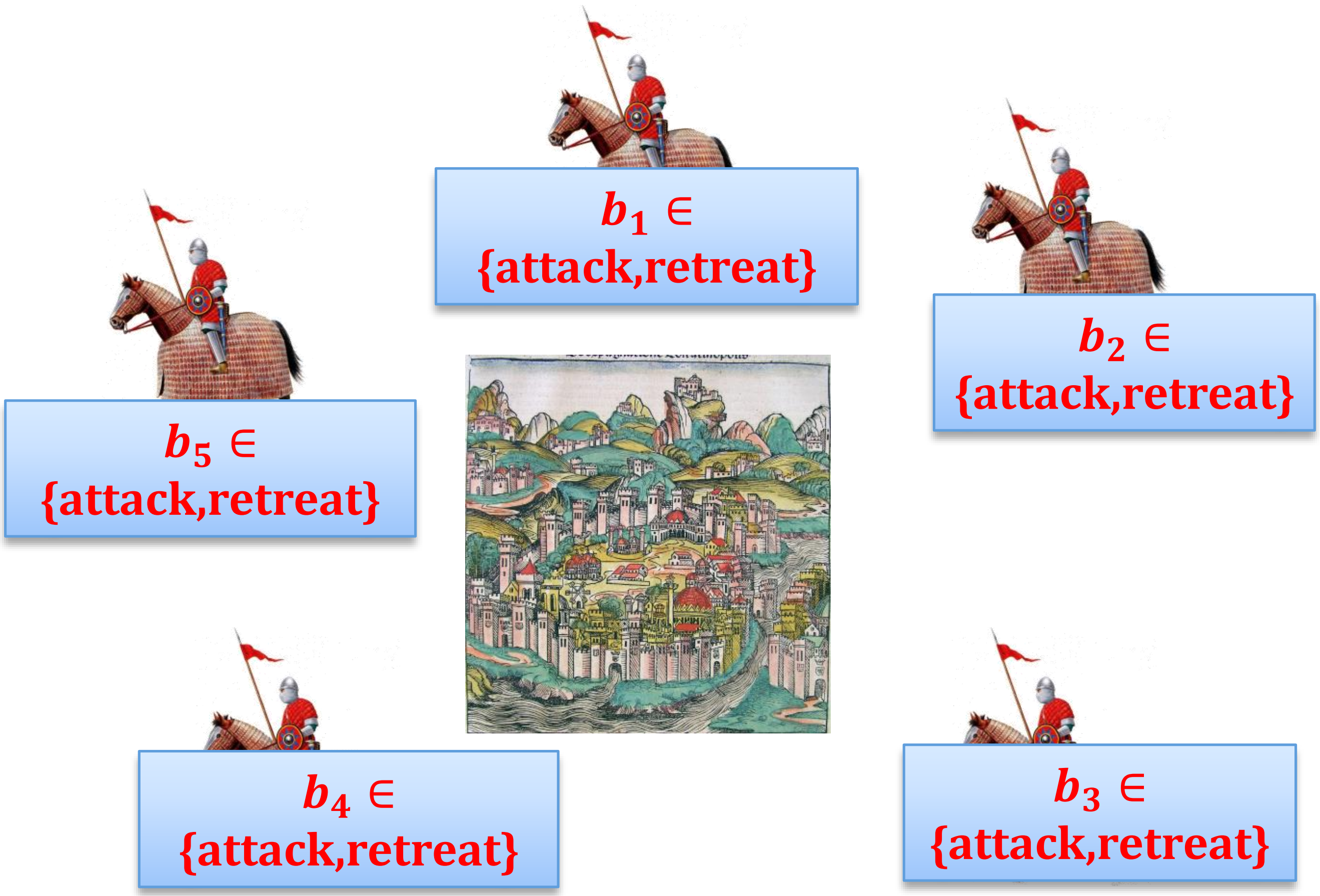


Every player receives **the same** message (even if the sender is malicious).

Byzantine Agreement

A classical problem in distributed computing **[Lamport, Shostak, Pease, 1982]**:

- n generals (connected with private channels) want to reach a consensus
- there may be t traitors among them



Formally

We have the following requirements

- **Non-triviality**: If all loyal generals have the same input bit ***b*** then, the only possible decision value of the loyal generals is ***b***.
- **Agreement**: The loyal generals should agree on the decision.
- **Limited bureaucracy**: The protocol must terminate

A classical result

Byzantine agreement is possible if and only if

$$\boldsymbol{t < n/3}$$

Broadcast channel vs. byzantine agreement

If the **broadcast channel** is available then the **byzantine agreement** can be achieved as follows:

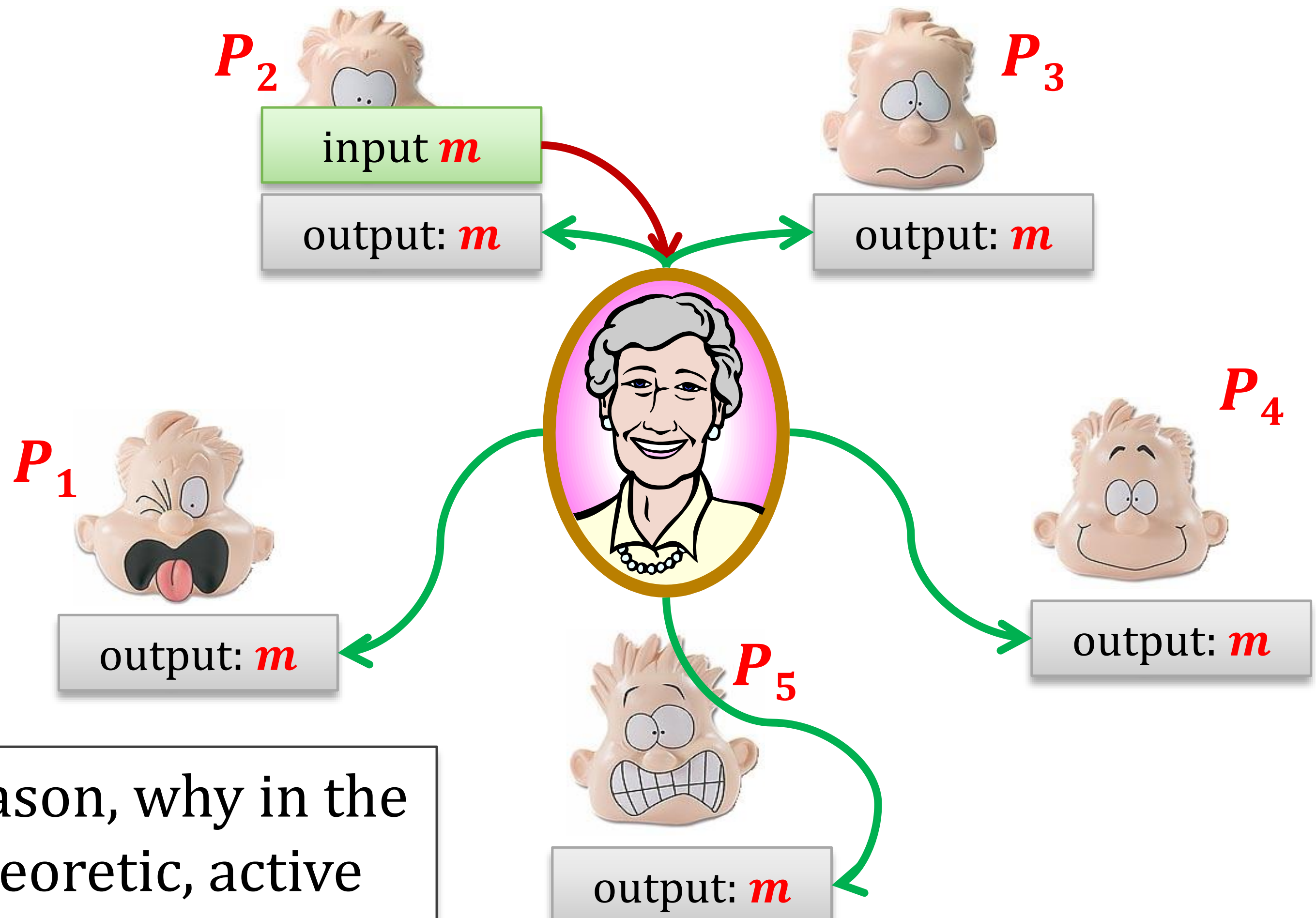
1. every party P_i broadcasts her input s_i
2. the majority of the broadcasted values is the agreed value.

Fact

In the **information-theoretic settings**:

a broadcast channel can be "emulated" by a multiparty protocol.

Emulation



So, this is the reason, why in the information theoretic, active case we have $t < n/3$

Idea

Allow the parties to use a broadcast channel.

We get:

setting	**adversary type**	**condition**
information-theoretic	passive	$t < n/2$
information-theoretic	active	$t < n/3$
information-theoretic (with broadcast)	**active**	$t < n/2$

Plan

1. Definitions and motivation
2. Security against the threshold adversaries
    1. overview of the results
    2. overview of the constructions
3. General adversary structures
4. Applications

How to construct such protocols?

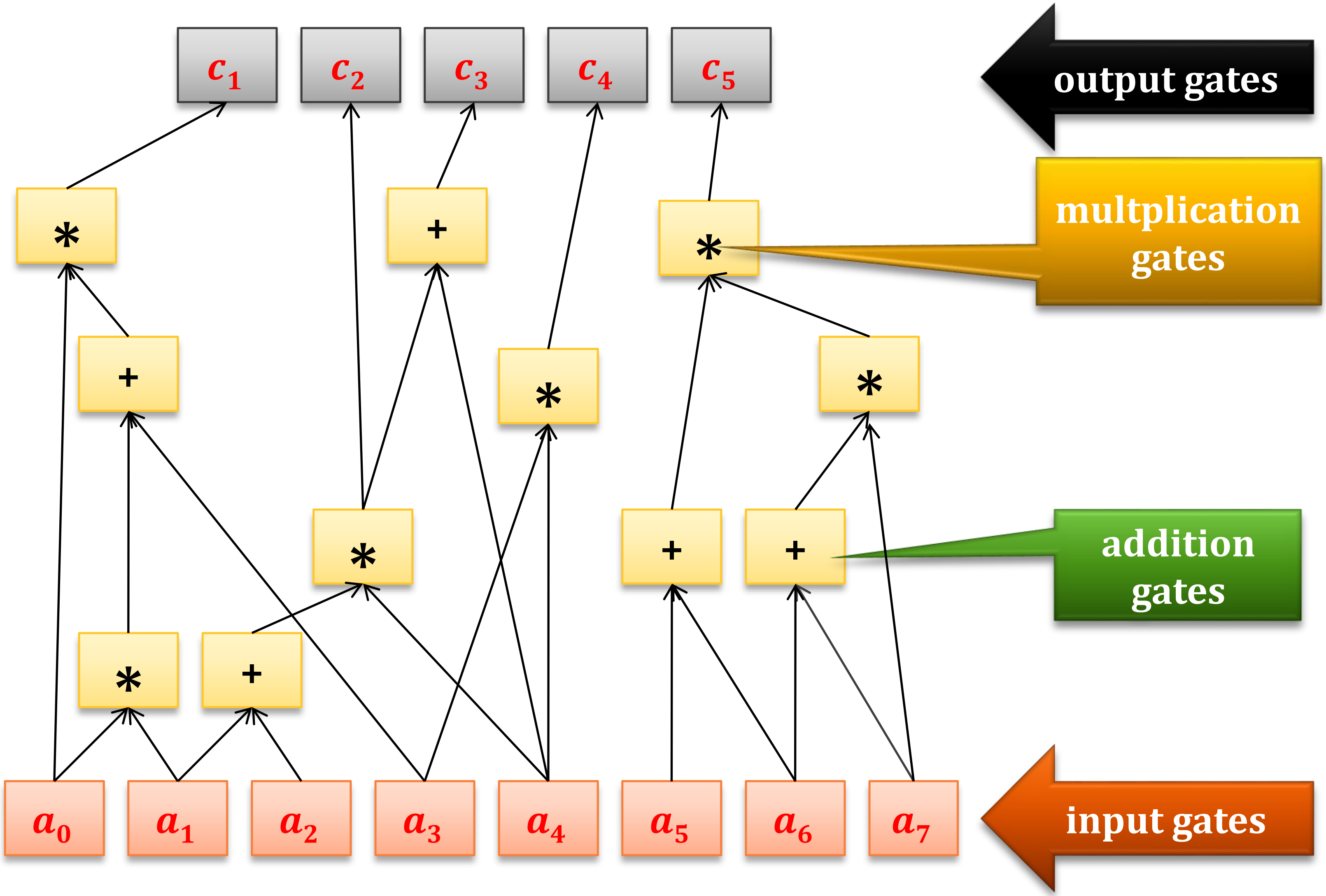
The general scheme is like in the two-party case:

1. Represent the **function as a circuit**.

   usually: arithmetic circuit over some field

2. Let **every party "share" her input** with the other parties.
3. **Evaluate the circuit gate-by-gate** (maintaining the invariant that the values of the intermediary gates are shared between the parties)
4. **Reconstruct the output**.

Arithmetic circuits (over a field **F**)



How to share a secret?

Informally:

We want to share a secret $\boldsymbol{S}$ between a group of parties, in such a way that:

1. any set of up to $\boldsymbol{t}$ corrupted parties has no information on $\boldsymbol{S}$, and
2. if $\boldsymbol{t+1}$ parties cooperate then they can reconstruct the secret $\boldsymbol{S}$.

m-out-of-n secret sharing

$$\boxed{m = t + 1}$$

dealer's secret S

$(n = 5)$

S_1 S_2 S_3 S_4 S_5

1. Every set of at least m players can **reconstruct S**.
2. Any set of less than m players has **no information about S**.

<u>**note**</u>: this primitive assumes that the adversary is **passive**

m-out-of-n secret sharing – more formally

Every secret sharing protocol consists of

- a **sharing** procedure: $(S_1, \ldots, S_n) \coloneqq \mathbf{share}(S)$
- a **reconstruction** procedure:
  for any distinct $i_1, \ldots, i_m$ we have $S \coloneqq \mathbf{reconstruct}(S_{i_1}, \ldots, S_{i_m})$

matching

- a **security condition**:
  for every S, S' and every $i_1, \ldots, i_{m-1}$:

  $\left(S_{i_1}, \ldots, S_{i_{m-1}}\right)$ and $\left(S'_{i_1}, \ldots, S'_{i_{m-1}}\right)$ are distributed identically,

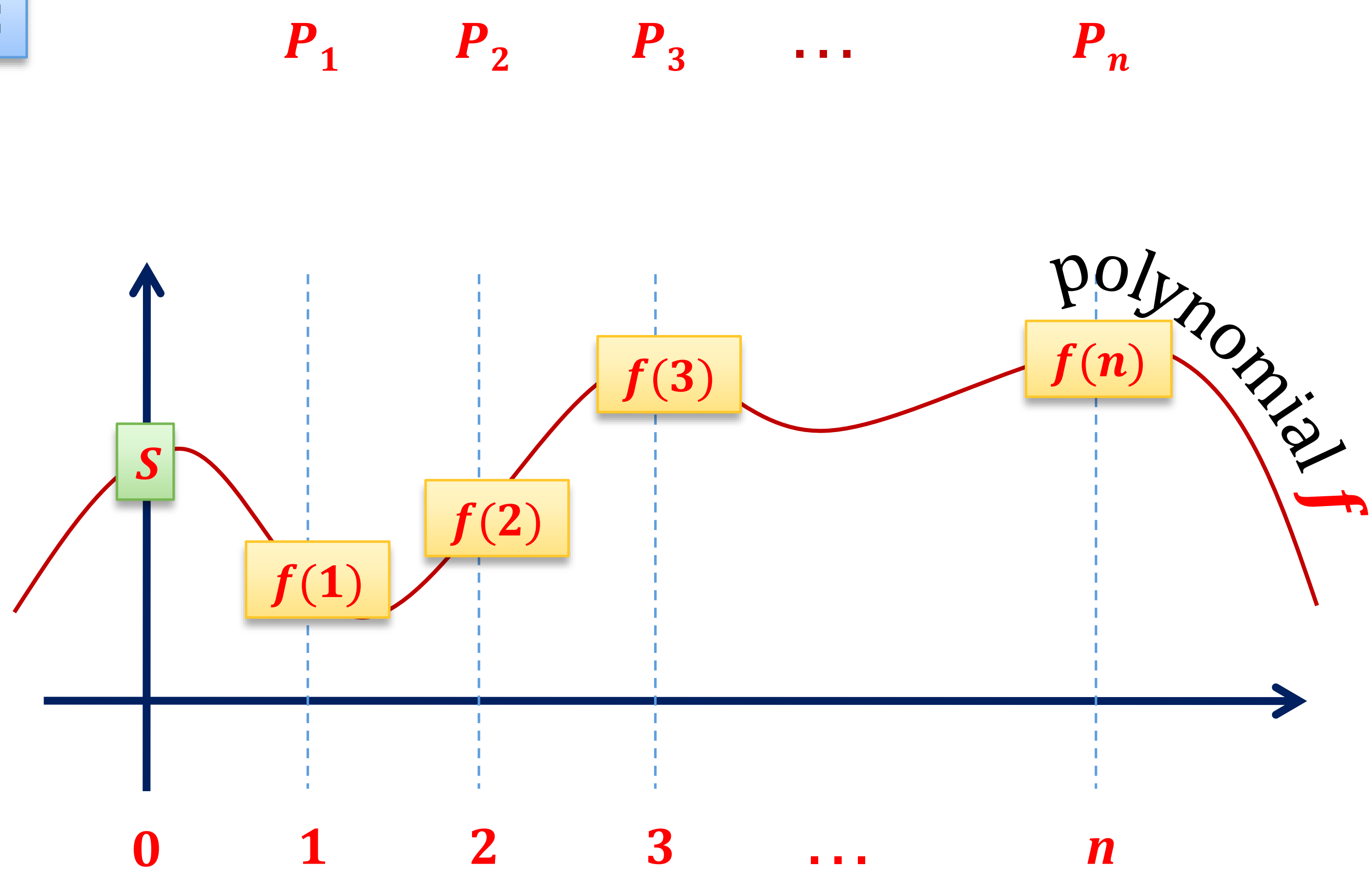
  where:

  $$(S_1, \ldots, S_n) \coloneqq \mathbf{share}(S) \text{ and } (S'_1, \ldots, S'_n) \coloneqq \mathbf{share}(S')$$

Shamir's secret sharing [1/2]

Suppose that S is an element of some finite field $\mathbf{F}$, such that $|\mathbf{F}| > n$
f – a random polynomial of degree $m-1$ over $\mathbf{F}$ such that $f(0) = S$

sharing:



Shamir's secret sharing [2/2]

reconstruction:

Given $\boldsymbol{f(i_1), \ldots, f(i_m)}$ one can interpolate the polynomial $\boldsymbol{f}$ in point $\mathbf{0}$.

security:

One can show that $\boldsymbol{f(i_1), \ldots, f(i_{m-1})}$ are independent from $\boldsymbol{f(0)}$.

How to construct a MPC protocol on top of Shamir's secret sharing?

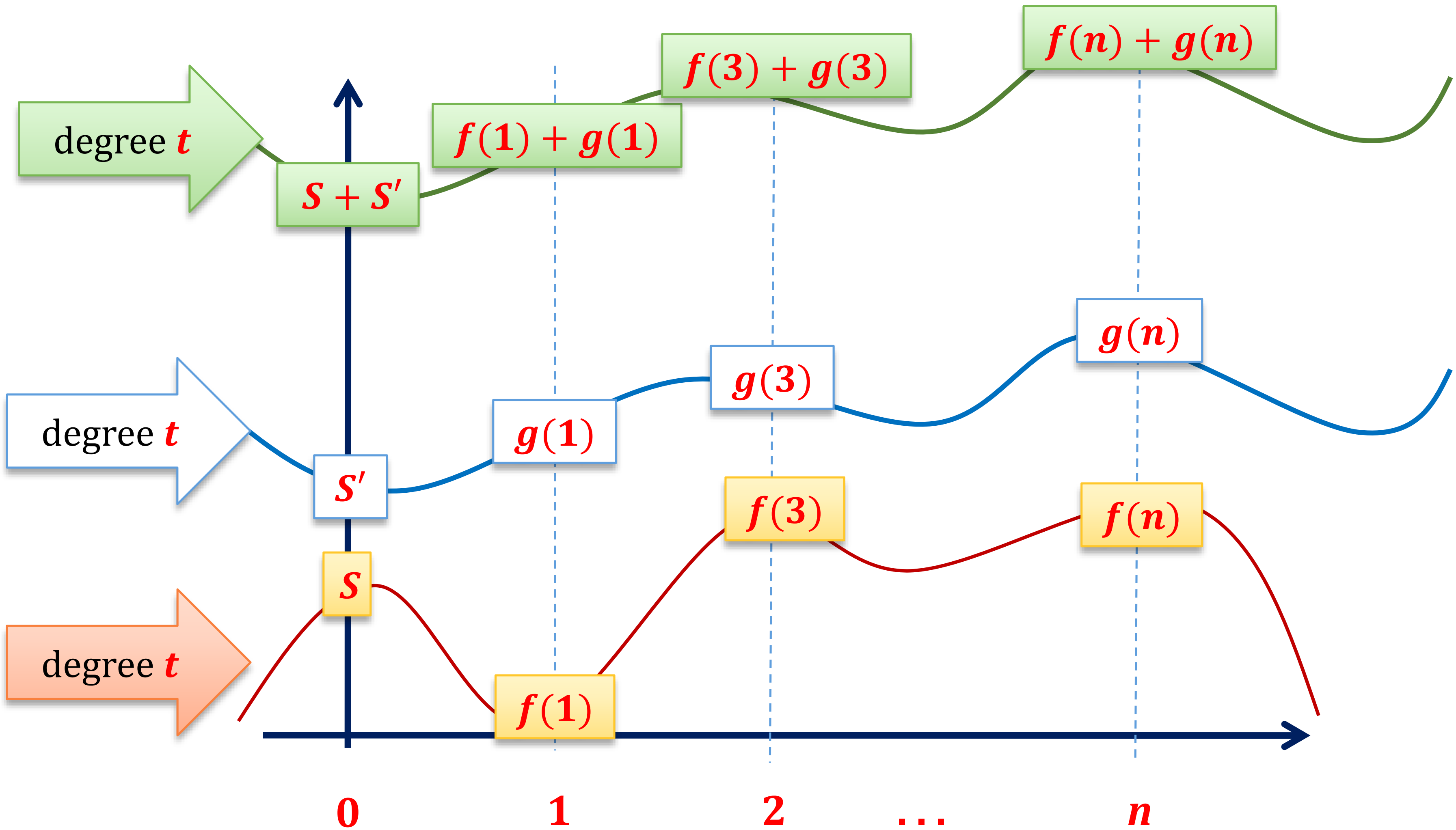
Observation

Addition is easy...

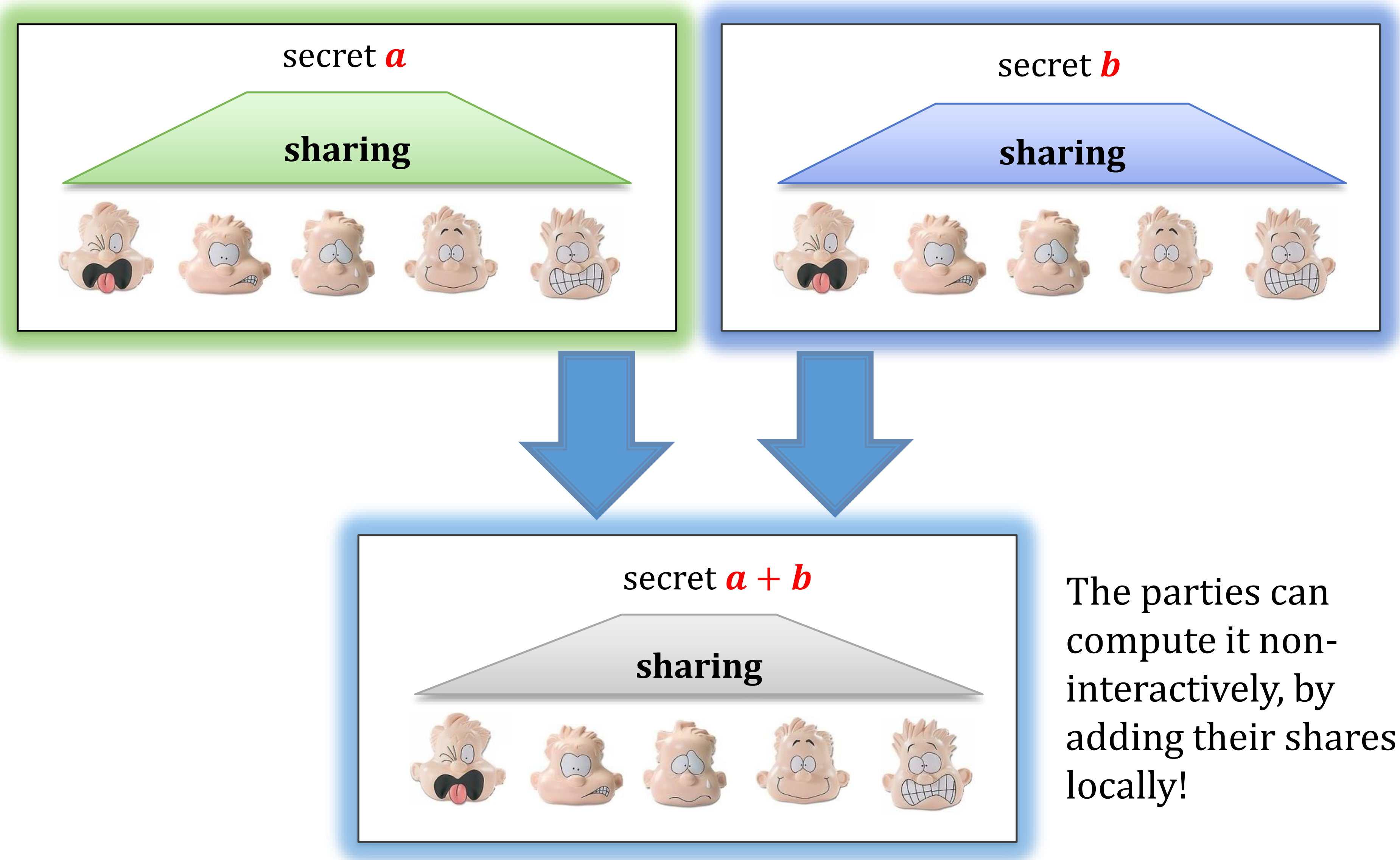
Why?

Because polynomials are homomorphic with respect to addition.

Polynomials are homomorphic with respect to addition



Addition



The parties can compute it non-interactively, by adding their shares locally!

How can we use it?

We can construct a protocol for computing

$$\boldsymbol{f}(\boldsymbol{a_1}, \ldots, \boldsymbol{a_n}) := \boldsymbol{a_1} + \cdots + \boldsymbol{a_n}$$

This protocol will be secure against an adversary that

- corrupts up to $\boldsymbol{t}$ parties and is
- **passive**, and
- **information-theoretic**.

A protocol for computing

$$f(a_1, \dots, a_n) \coloneqq a_1 + \cdots + a_n$$

1. Each party P_i shares her input using a $(t+1)$-out-of-n Shamir's secret sharing.
   Let $a_i^1, \dots, a_i^n$ be the shares.
   Therefore at the end we have quadratic number of shares

a_1		a_i		a_n
a_1^1	$\cdots$	a_i^1	$\cdots$	a_n^1
$\vdots$		$\vdots$		$\vdots$
a_1^j	$\cdots$	a_i^j	$\cdots$	a_n^j
$\vdots$		$\vdots$		$\vdots$
a_1^n	$\cdots$	a_i^n	$\cdots$	a_n^n

2. Each $\boldsymbol{P_j}$ computes a sum of the shares that he received

this is what $\boldsymbol{P_j}$ received in **Step 1**

a_1^1	$\cdots$	a_i^1	$\cdots$	a_n^1
$\vdots$		$\vdots$		$\vdots$
a_1^j	$\cdots$	a_i^j	$\cdots$	a_n^j
$\vdots$		$\vdots$		$\vdots$
a_1^n	$\cdots$	a_i^n	$\cdots$	a_n^n

$$b^1 := \sum_i a_i^1$$

$$\vdots$$

$$b^j := \sum_i a_i^j$$

$$\vdots$$

$$b^n := \sum_i a_i^n$$

The final steps:

3. Each party P^j broadcasts b^j
4. Every party can now reconstruct

$$f(a_1, \ldots, a_n) := a_1 + \cdots + an$$

by interpolating the shares $b^1, \ldots, b^n$

It can be shown that no coalition of up to t parties can break the security of the protocol.

(Even if they are infinitely-powerful)

How to construct a protocol for any function

Polynomials are homomorphic also with respect to multiplication.

Problem

The degree gets doubled...

Hence, the construction of such protocols is not-trivial.

But it is possible! **[exercise]**

Plan

1. Definitions and motivation
2. Security against the threshold adversaries
   1. overview of the results
   2. overview of the constructions
3. General adversary structures
4. Applications

General adversary structures

Sometimes assuming that the adversary can corrupt up to ***t*** parties is not general enough.

It is better to consider arbitrary **<u>coalitions</u>** of the sets of parties that can be corrupted.

Example of coalitions

Adversary structures

$\boldsymbol{\Delta}$ is an **adversary structure over the set of players** $\{P_1, \dots, P_n\}$ if:

$$\boldsymbol{\Delta \subseteq 2^{\{P_1, \dots, P_n\}}}$$

and for every $\boldsymbol{A \in \Delta}$

$$\text{if } \boldsymbol{B \subseteq A} \text{ then } \boldsymbol{B \in \Delta}$$

This property is called **monotonicity**.
Because of this, to specify $\boldsymbol{\Delta}$ it is enough to specify a set $\boldsymbol{M}$ of its **maximal sets**.
We will also say that $\boldsymbol{M}$ **induces** $\boldsymbol{\Delta}$.

Q2 and Q3 structures

We say that A is a Δ-**adversary** if he can corrupt only the sets in Δ.

How to generalize the condition that $t < n/2$?
We say that a structure Δ is **Q2** if

$$\forall_{A,B\in\Delta}\, A \cup B \neq \{P_1, \dots, P_n\}$$

What about "$t < n/3$"?
We say that a structure Δ is **Q3** if

$$\forall_{A,B,C\in\Delta}\, A \cup B \cup C \neq \{P_1, \dots, P_n\}$$

A generalization of the classical results

[Martin Hirt, Ueli M. Maurer: Player Simulation and General Adversary Structures in Perfect Multiparty Computation. J. Cryptology, 2000]

setting	adversary type	condition	generalized condition
information-theoretic	passive	$t < n/2$	**Q2**
information-theoretic	active	$t < n/3$	**Q3**
information-theoretic with broadcast	active	$t < n/2$	**Q2**

There is one problem, though...

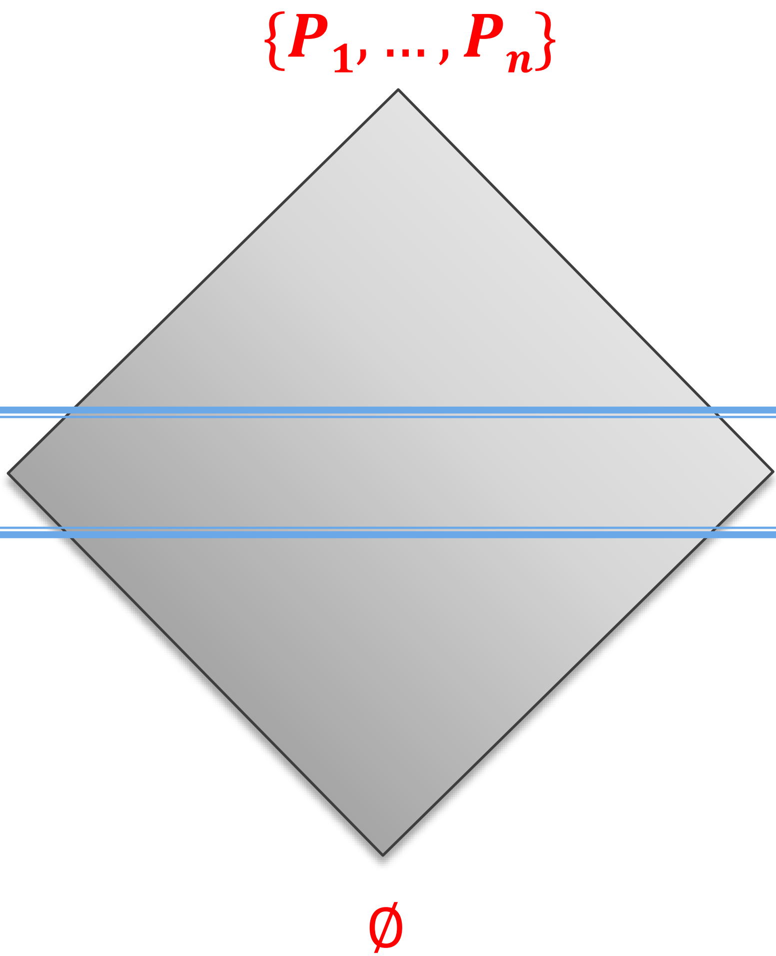
What is the **total** number of possible adversary structures?

Fact

It is **doubly-exponential** in the number of players.

Why?

inclusion is a partial order on the set of subsets of $\{P_1, \dots, P_n\}$



(suppose n is even)

X := family of sets of cardinality $n/2$

$$|X| = \binom{n}{n/2} \geq 2^{n/2}$$

On the other hand...

($\boldsymbol{X}$:= family of sets of cardinality $\boldsymbol{n/2}$)

Every subset of $\boldsymbol{X}$ induces a different adversary structure.

Hence the set of all adversary structures has cardinality at least:

$$\mathbf{2}^{|\boldsymbol{X}|} \geq \mathbf{2}^{\mathbf{2}^{\boldsymbol{n}/\mathbf{2}}}$$

So, we have a problem, because

On the other hand

The number of poly-time protocols is just exponential in the size of the input.

Hence

If the number of players is super-logarithmic, we cannot hope to have a poly-time protocol for every adversary structure.

What to do?

Consider only those adversary structure that "can be represented in polynomial space".

For example see:

Ronald Cramer, Ivan Damgård, Ueli M. Maurer: **General Secure Multi-party Computation from any Linear Secret-Sharing Scheme.** EUROCRYPT 2000

Plan

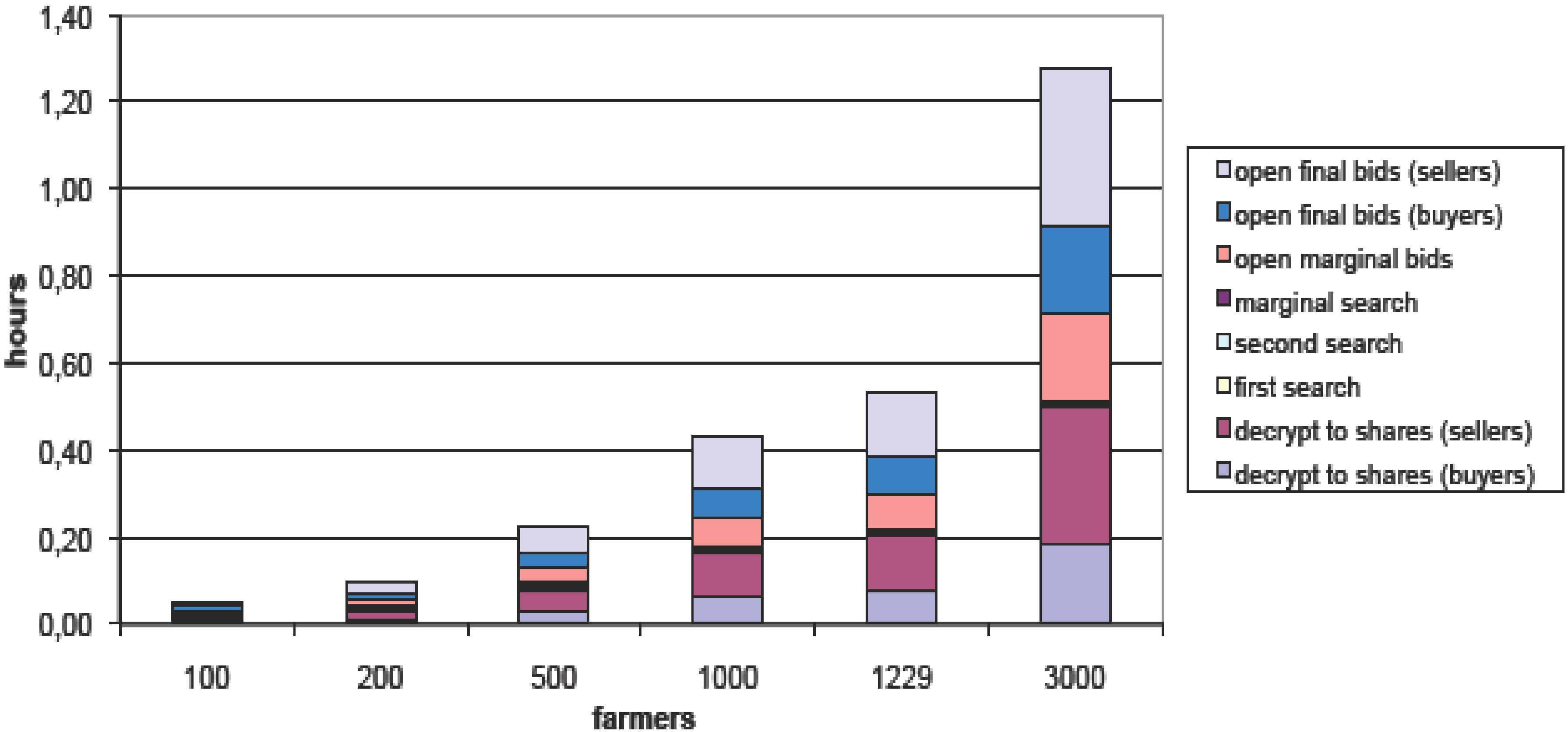
1. Definitions and motivation
2. Security against the threshold adversaries
   1. overview of the results
   2. overview of the constructions
3. General adversary structures
4. Applications

Practical implementation

Peter Bogetoft et al. **Multiparty Computation Goes Live.** 2009

The Danish farmers can now bet in a secure way for the contracts to deliver sugar beets.

Efficiency



1,40
1,20
1,00
0,80
0,60
0,40
0,20
0,00
hours
100
200
500
1000
1229
3000
farmers
open final bids (sellers)
open final bids (buyers)
open marginal bids
marginal search
second search
first search
decrypt to shares (sellers)
decrypt to shares (buyers)

Other applications

Distributed cryptography is also used in the following way.

Suppose we have a secret key $\boldsymbol{sk}$ (for a signature scheme) and we do not wan to store it on on machine.

Solution:

1. share $\boldsymbol{sk}$ between $\boldsymbol{n}$ machines $\boldsymbol{P_1, \ldots, P_n}$
2. "sign" in a distributed way (without reconstructing $\boldsymbol{sk}$)

see e.g.:

Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin: **Robust Threshold DSS Signatures.** EUROCRYPT 1996