Perfectly-Secure Multiplication for Any t<n/3

Gilad Asharov — **Bar-Ilan University, Israel**

Yehuda Lindell — Bar-Ilan University, Israel

Tal Rabin — IBM Research, New York

Secure Multiparty Computation

- A set of parties with private inputs wish to compute some joint function of their inputs
- Parties wish to preserve some security properties. E.g., privacy and correctness
  - Example: secure election protocol
- Security must be preserved in the face of adversarial behavior by some of the participants, or by an external party

The BGW Protocol [STOC 1988]

- Michael **B**en-Or, Shafi **G**oldwasser and Avi **W**igderson

- A protocol for general multiparty computation
  - Perfectly secure
  - Adaptively secure
  - Concurrently secure

- Elegant and beautiful construction
- A huge impact on our field

Our Results

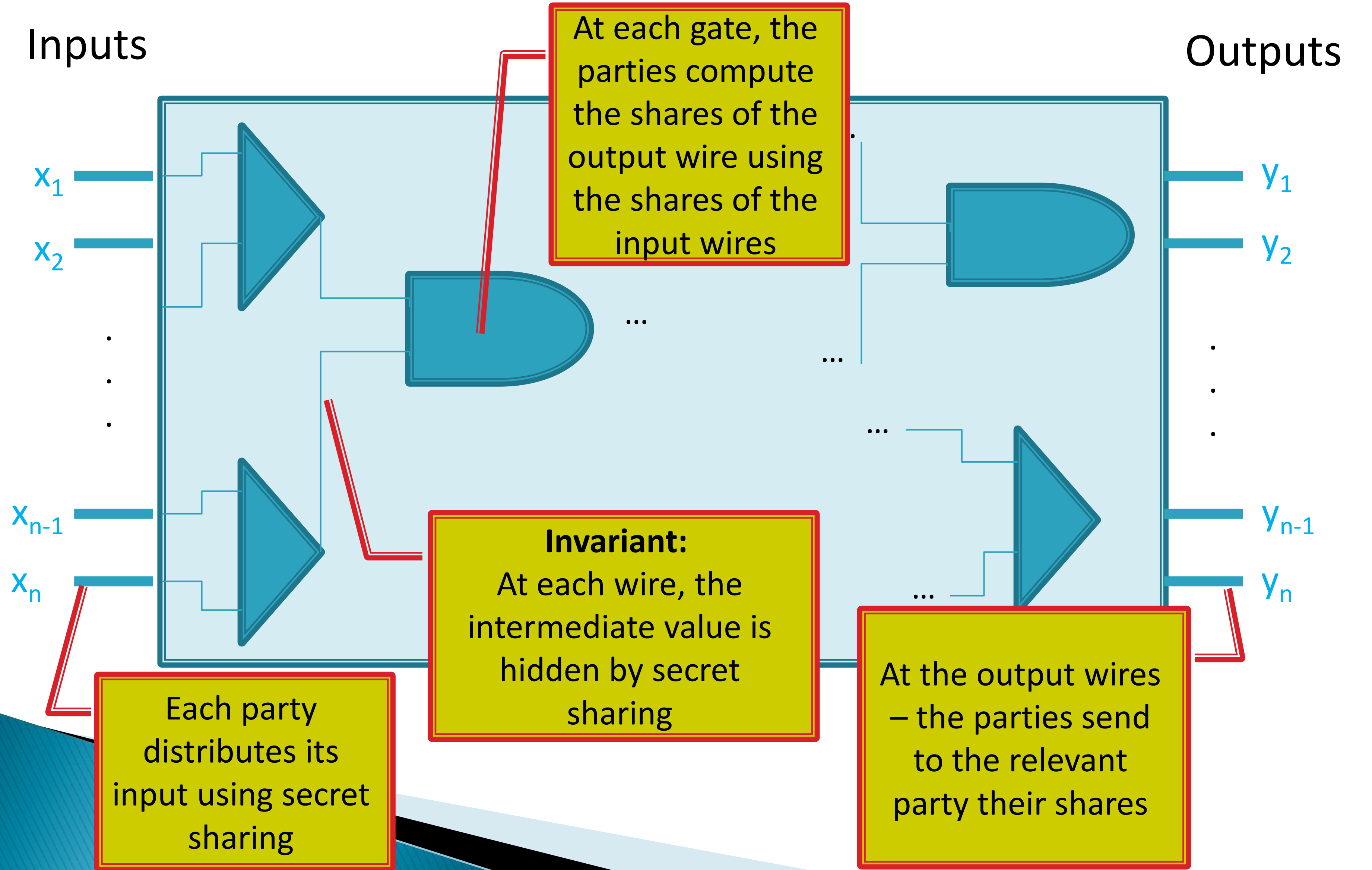
- A full specification of the BGW multiplication protocol
  - The protocol requires a new step for the case of $n/4 \leq t < n/3$
  - A full proof of security
- A new multiplication protocol
  - More efficient
  - Simpler
  - Constant round per multiplication (as BGW)

Related Work

- Perfect multiplication based on homomorphic secret sharing
  - [Cramer, Damgard, Maurer 00]

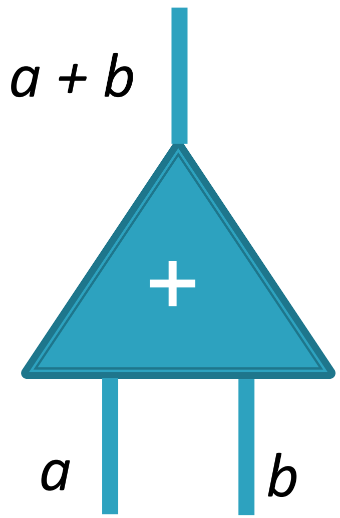
- Efficiency of perfect multiplication
  - Player elimination technique [Hirt, Maurer, Przydatek 00] [Hirt, Maurer 01], [Beerliova-Trubıniova, Hirt 06] [Hirt, Nielsen 06] [Damgard, Nielsen 07] [Trubıniova, Hirt 08]
  - Very efficient protocols
  - The round complexity per multiplication depends on the number of parties

The BGW Protocol

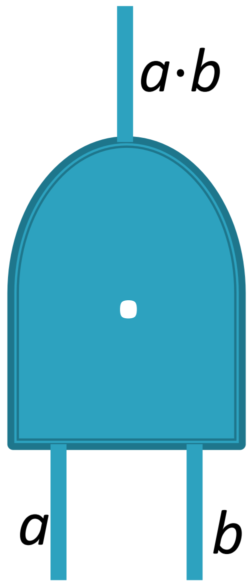


The Computation Stage

- The invariant:
  - Each party holds shares of *a* and *b*
- Addition Gate:
  - Each party locally adds its shares
    - The result is a share of a random polynomial of degree-t that hides *a+b*

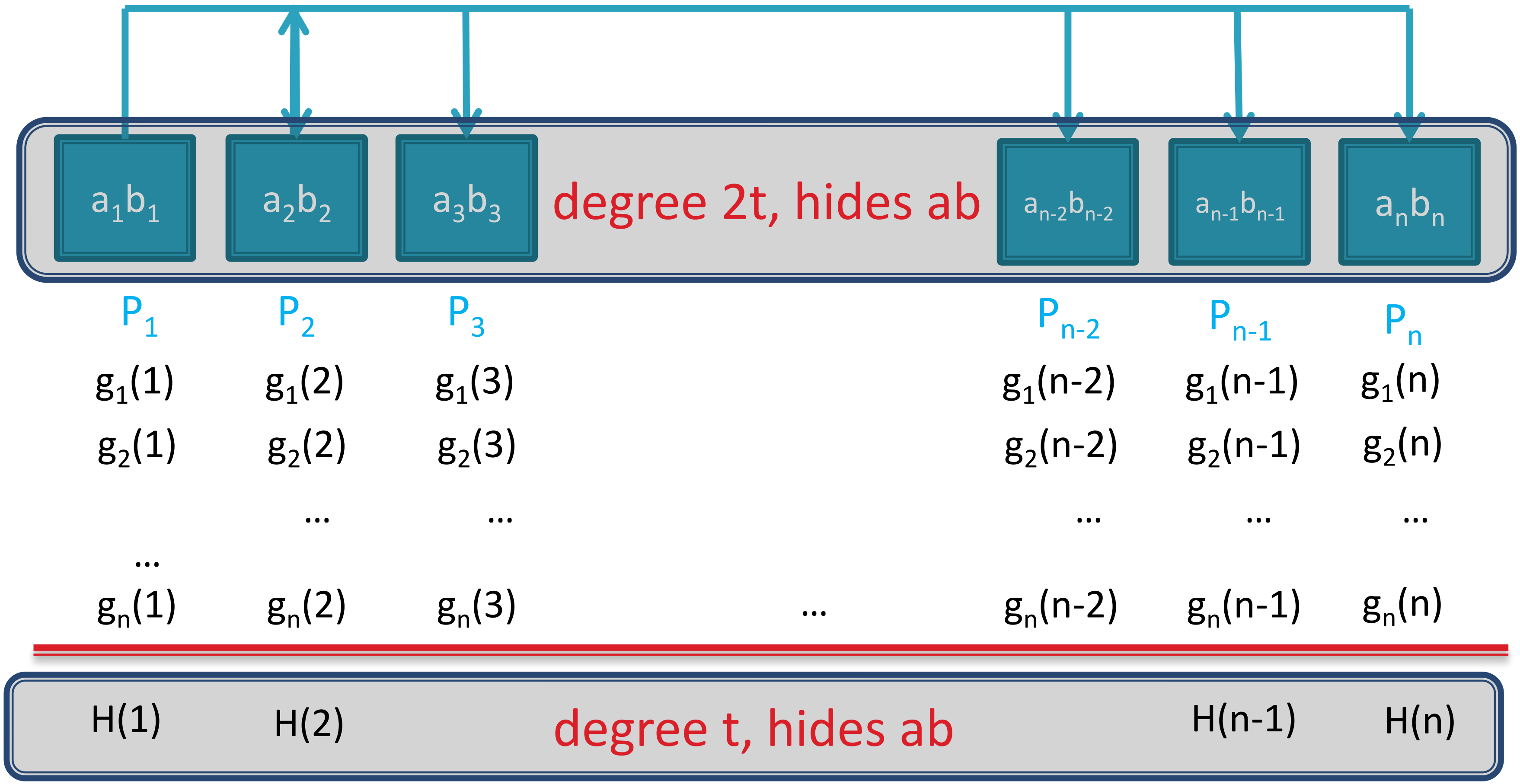


The Computation Stage



- The invariant:
  - Each party holds shares of *a* and *b*
- Addition Gate:
  - Each party locally adds its shares
    - The result is a share of a random polynomial of degree-t that hides *a+b*
- Multiplication Gate:
  - Each party locally multiplies its shares
    - Result is a share of a poly of degree-2t that hides $a \cdot b$
    - Run an interactive protocol to reduce the degree

The Multiplication Protocol (simplification according to [GRR98])



Possible whenever at least 2t+1 shares were sub-shared correctly

Moving to the Malicious* – Problem

a_1b_1 a_2b_2 a_3b_3 degree 2t, hides ab $a_{n-2}b_{n-2}$ $a_{n-1}b_{n-1}$ a_nb_n

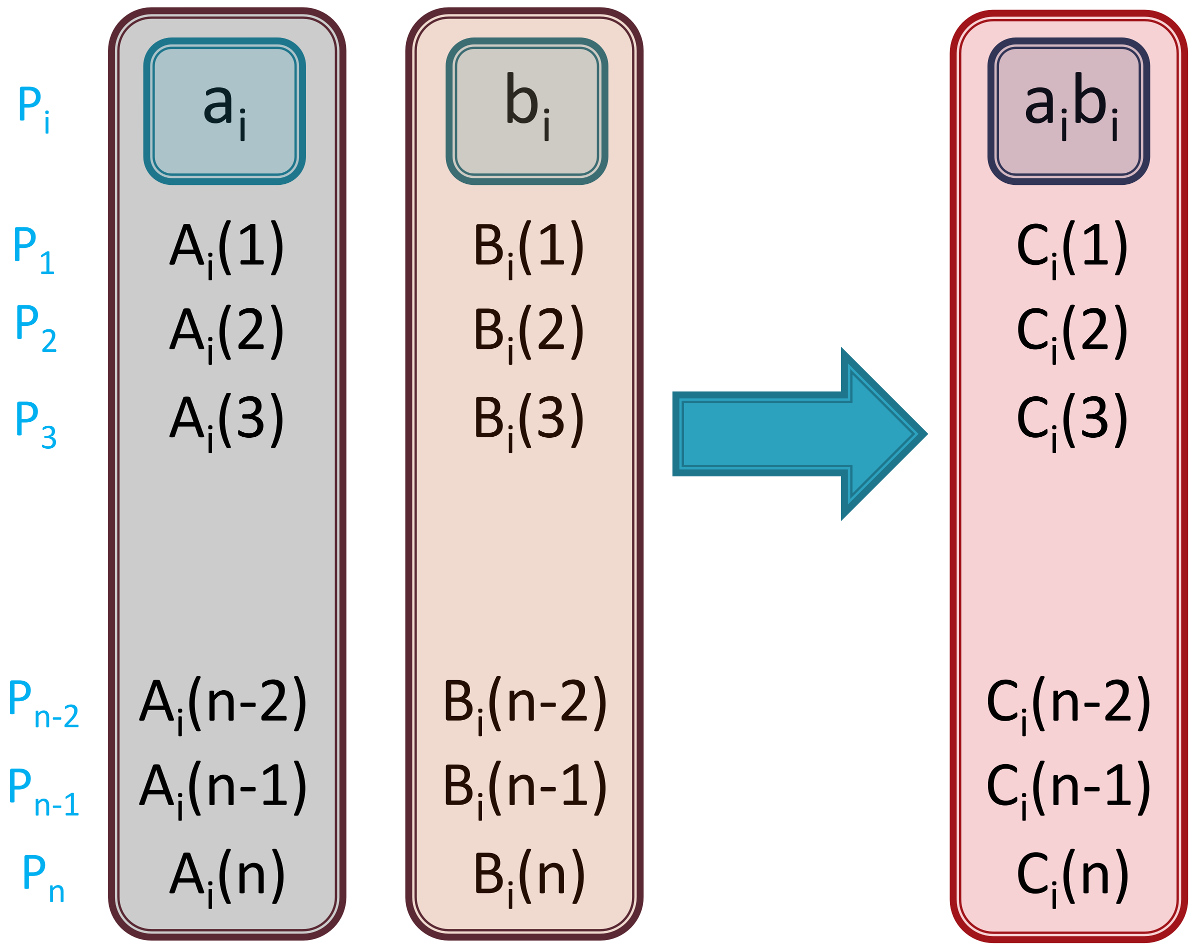
The honest parties need to identify the incorrect shares

*we assume:
at **least** *2t+1* honest parties
at **most** *t* corrupted parties

First BGW Tool: Robust Sub-Sharing

$f(1)$	$f(2)$	$f(3)$	degree-t	$f(n-2)$	$f(n-1)$	$f(n)$
P_1	P_2	P_3		P_{n-2}	P_{n-1}	P_n
$g_1(1)$	$g_1(2)$	$g_1(3)$	…	$g_1(n-2)$	$g_1(n-1)$	$g_1(n)$
$g_2(1)$	$g_2(2)$	$g_2(3)$	…	$g_2(n-2)$	$g_2(n-1)$	$g_2(n)$
$g_3(1)$	$g_3(2)$	$g_3(3)$	…	$g_3(n-2)$	$g_3(n-1)$	$g_3(n)$
…	…	…		…	…	…
$g_n(1)$	$g_n(2)$	$g_n(3)$	…	$g_n(n-2)$	$g_n(n-1)$	$g_n(n)$

Second BGW Tool: Verifying Product



Multiplication - Overview

a_1 a_2 a_3 hides a a_{n-2} a_{n-1} a_n

$A_1(1)$ $A_1(2)$ $A_1(3)$ … hides a_1 $A_1(n-2)$ $A_1(n-1)$ $A_1(n)$

$A_2(1)$ $A_2(2)$ $A_2(3)$ … hides a_2 $A_2(n-2)$ $A_2(n-1)$ $A_2(n)$

… … … …

b_1 b_2 b_3 hides b b_{n-2} b_{n-1} b_n

$B_1(1)$ $B_1(2)$ $B_1(3)$ … hides b_1 $B_1(n-2)$ $B_1(n-1)$ $B_1(n)$

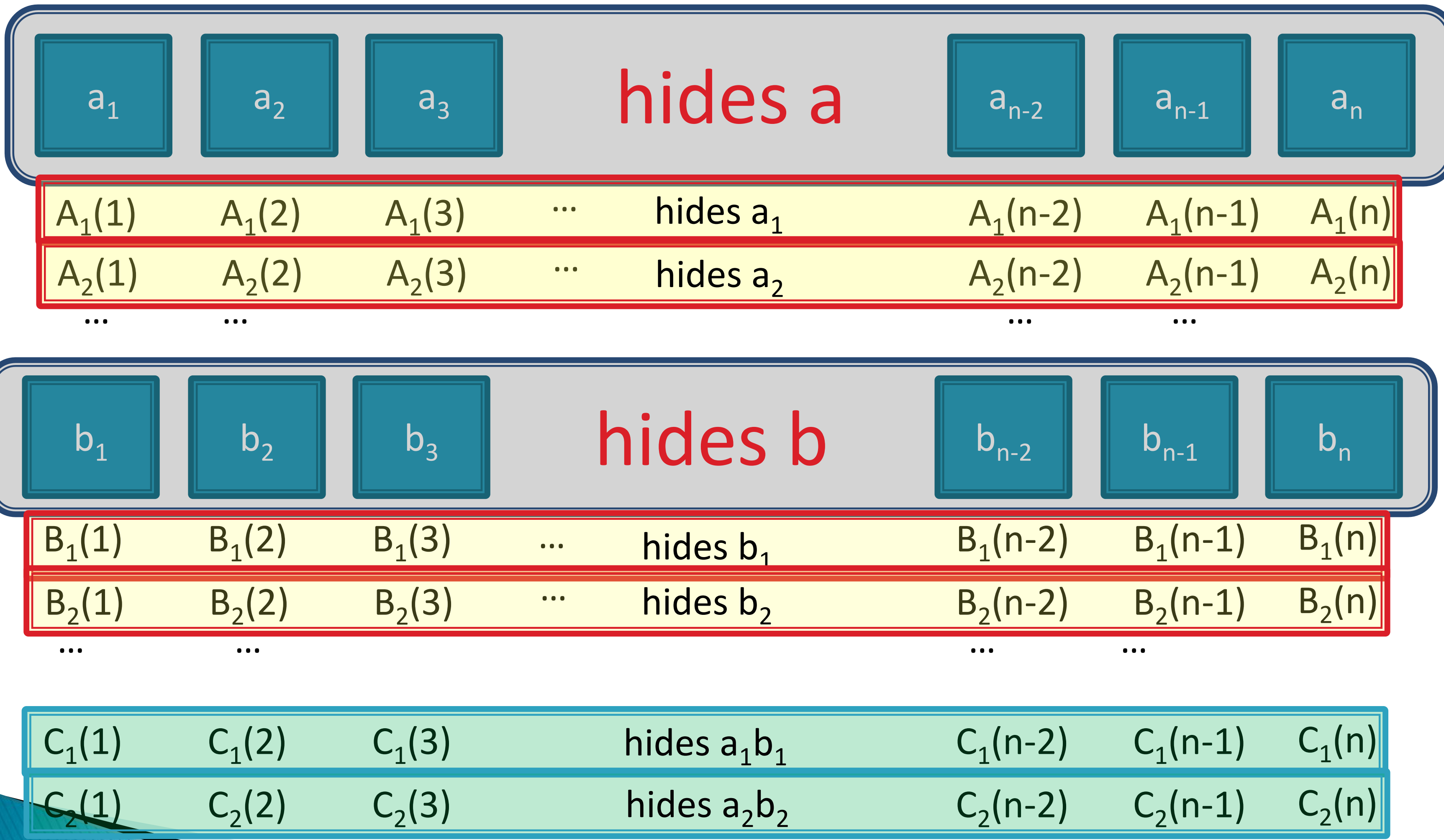
$B_2(1)$ $B_2(2)$ $B_2(3)$ … hides b_2 $B_2(n-2)$ $B_2(n-1)$ $B_2(n)$

… … … …

$C_1(1)$ $C_1(2)$ $C_1(3)$ hides a_1b_1 $C_1(n-2)$ $C_1(n-1)$ $C_1(n)$

$C_2(1)$ $C_2(2)$ $C_2(3)$ hides a_2b_2 $C_2(n-2)$ $C_2(n-1)$ $C_2(n)$

Multiplication - Overview



The Second Tool: Proving that $c_i=a_ib_i$

- **Inputs:**

The parties need to verify that $C_i(x)$ is of degree-t

  - The free coefficient of $C_i(x)$ is always $A_i(0)B_i(0) = a_ib_i$
  - Choosing $D_1,\ldots,D_t$ inappropriately can end up with a polynomial of degree higher than t

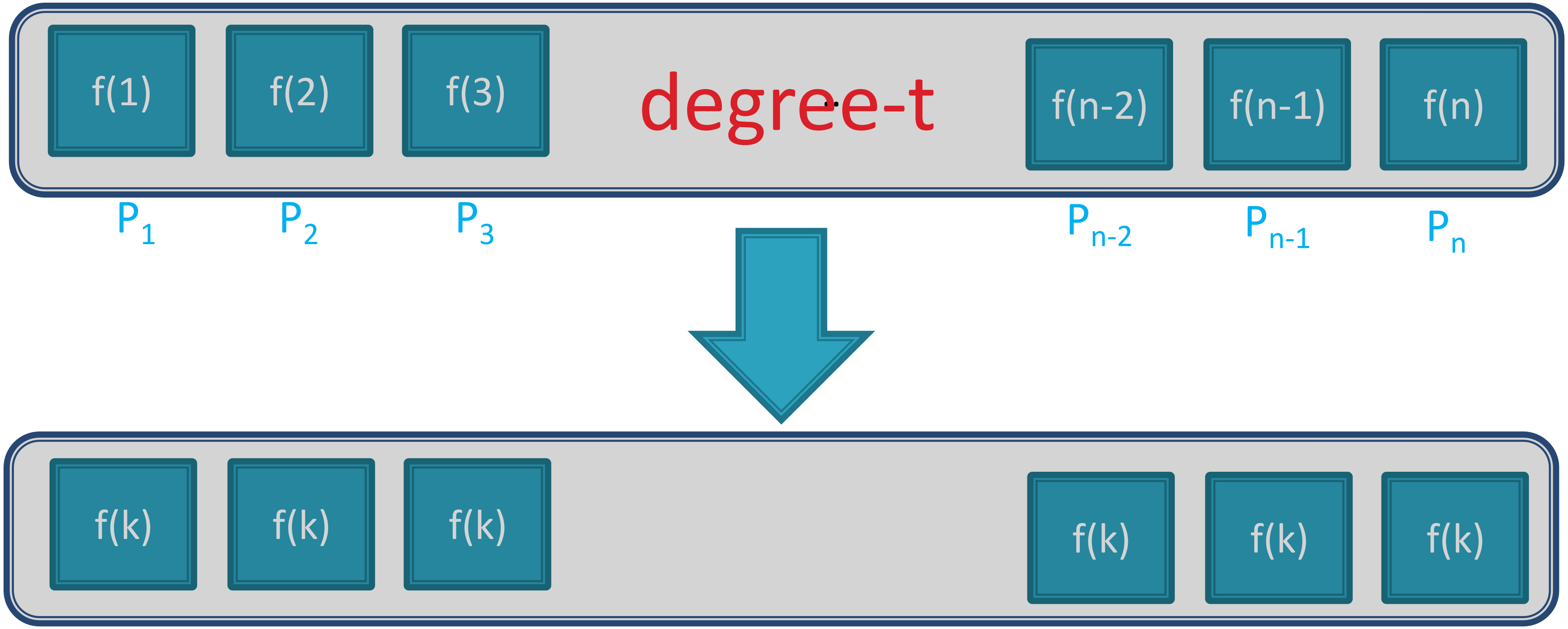
Verifying the Degree

- Parties have shares of $C_i(x)$ and want to check that it is of degree-t
- P_i distributes $C'_i(x)$ using VSS (guarantees degree-t) and claims that $C'_i(x) = C_i(x)$
  - $C_i(0)$ has the correct free coefficient, but unknown degree
  - $C'_i(x)$ is of degree-t, not necessarily the correct free coefficient
- Each party P_j checks that $C'_i(j) = C_i(j)$
  - If $C'_i(j) \neq C_i(j)$ – it broadcasts a "**complaint**"
- If number of complaints > t : "reject"
  - need more than *t* complaints, since the adversary may complain about an honest dealer

A Subtle Attack on this Solution

- The dealer creates $D_1(x),\ldots,D_t(x)$ **not** according to the protocol and so $C_i(x)$ is of degree higher than *t*
- It chooses $C'_i(x)$ of degree-t such that $C'_i(j) = C_i(j)$ for t+1 honest parties, but $C'_i(0) \neq a_i b_i$
- The corrupted parties do not complain
- Result:
  - t+1 honest parties **do not** complain
  - t corrupted parties **do not** complain
  - t honest parties complain
- **The polynomial is accepted**

Our Solution: F^{eval}



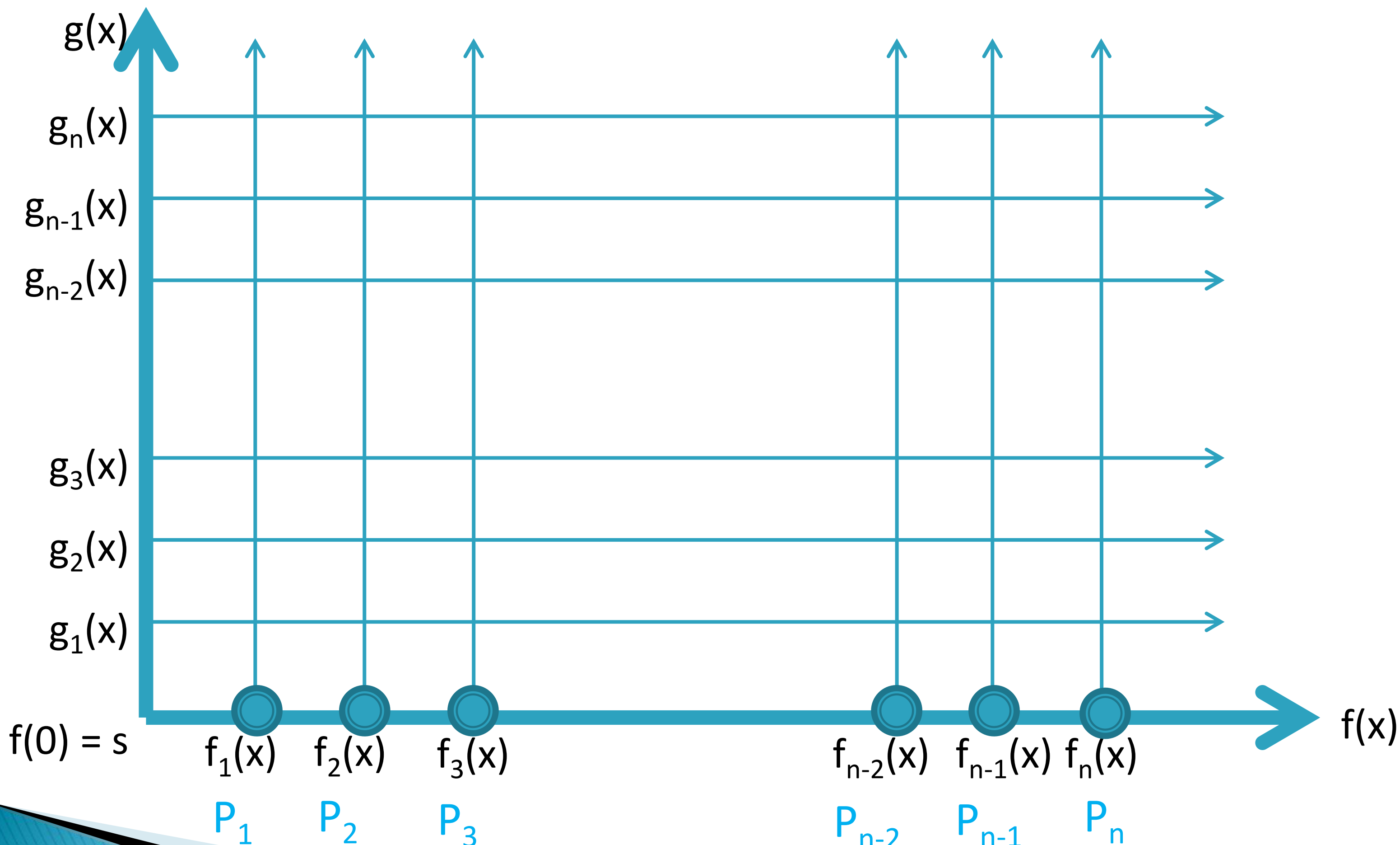
Verifying the Degree

- For each complaining party P_k – the parties check if its complaint is fake or legitimate:
  - Invoke f^{eval} on the shares of $A_i(x)$ and receive $A_i(k)$
  - Invoke f^{eval} on the shares of $B_i(x)$ and receive $B_i(k)$
  - …
  - The values $C'_i(k)$, $A_i(k)$, $B_i(k)$, $D_1(k)$, …, $D_t(k)$ become public
  - The parties compute $C_i(k)$, and compare it to $C_i'(k)$
    - If $C_i(k) = C_i'(k)$: the complaint is fake
    - If $C_i(k) \neq C_i'(k)$: the complaint is legitimate
- If there is one legitimate complaint – reject

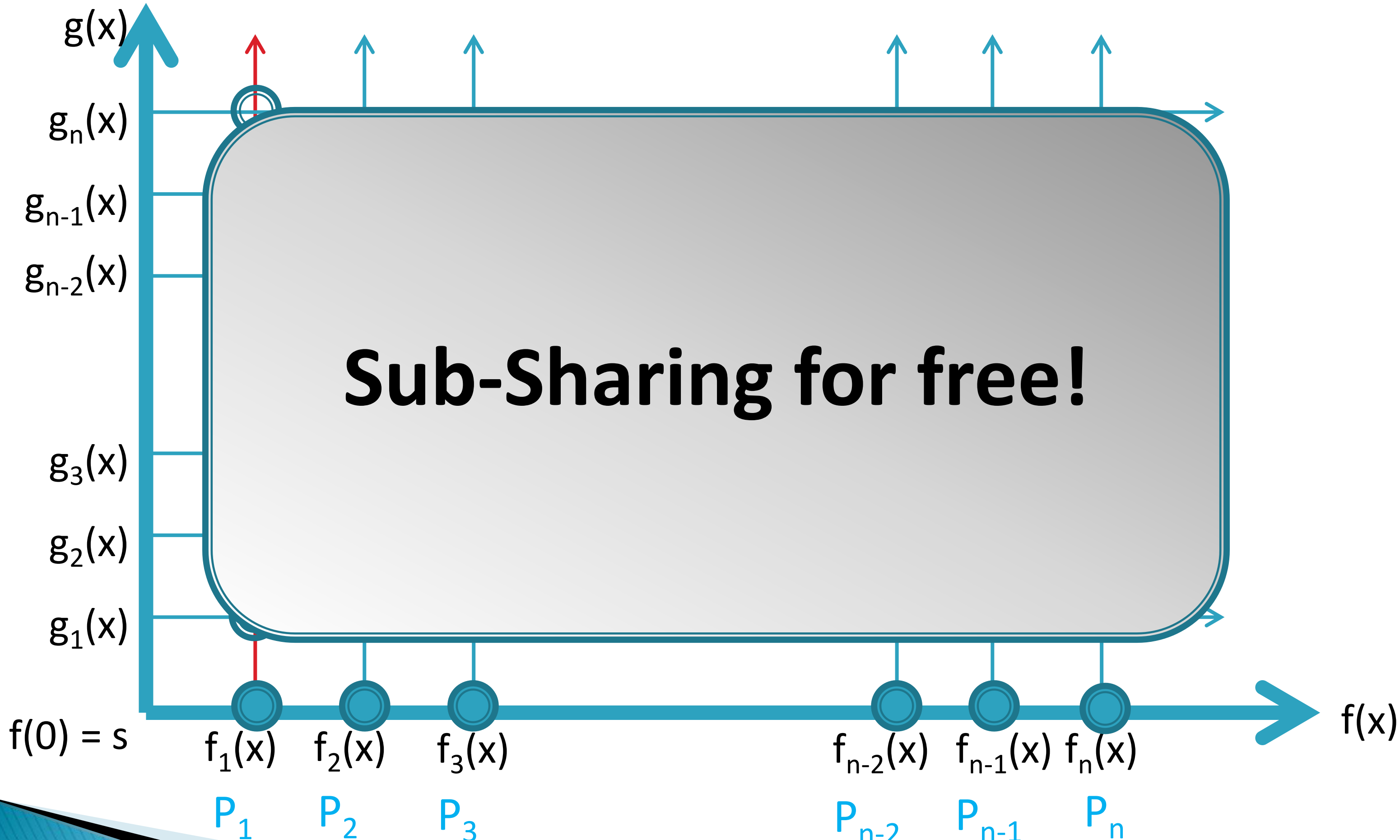
A New Constant-Round Multiplication Protocol

Utilizing Bivariate Sharing for Simplicity and Efficiency

Verifiable Secret Sharing



But…



g(x)
$g_n(x)$
$g_{n-1}(x)$
$g_{n-2}(x)$
$g_3(x)$
$g_2(x)$
$g_1(x)$
f(0) = s
Sub-Sharing for free!
f(x)
$f_1(x)$ $f_2(x)$ $f_3(x)$ $f_{n-2}(x)$ $f_{n-1}(x)$ $f_n(x)$
P_1 P_2 P_3 P_{n-2} P_{n-1} P_n

Simpler Construction

- The invariant is changed: univariate --> bivariate
- Sub-sharing for free – no need for robust sub-sharing
- f^{eval} and other tools are much more efficient and simpler
  - All the constructions become simpler
  - including the proof of security
- But maintaining the invariant requires some work
- Reduced the communication complexity of BGW by quadratic factor
  - Best constant-round multiplication protocol (by a linear factor)
  - Incomparable to player elimination techniques that have lower communication complexity but higher round complexity

Summary

- We study perfect multiplication
- We filled a missing gap in the BGW protocol
- A full proof of security
- A simpler construction
  - more efficient
  - and simpler

Thank You!