Advanced Topics in Cryptography

Lecture 3:

• A two-party protocol for a function which does not have a short circuit.

• Multi-party protocols.

Benny Pinkas

Related papers

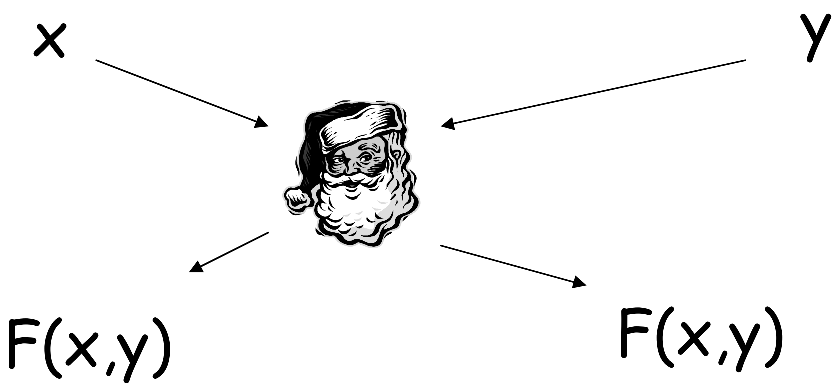
- Secure computation of medians
  - Aggarwal, N. Mishra and B. Pinkas, *Secure Computation of the K'th-ranked Element,* Eurocrypt '2004.
- Secure Computation
  - Ronald Cramer and Ivan Damgard, *Multiparty Computation, an Introduction,* Lecture notes. http://www.daimi.au.dk/~ivan/mpc_2004.pdf
  - Slides on MPC computation, Ivan Damgard, http://www.daimi.au.dk/~ivan/MPC2005.pdf.
  - M. Ben-Or, S. Goldwasser, A. Wigderson. *Completeness theorems for non-cryptographic fault-tolerant distributed computation*. 20th ACM symposium on Theory of Computing (STOC), 1988.

Secure two-party computation - definition

Input: x y

Output: F(x,y) *and nothing else*

As if…



Secure Function Evaluation

- Major Result [Yao]: "Any function that can be evaluated using polynomial resources can be securely evaluated using polynomial resources" (under some cryptographic assumption)

- This is shown through a transformation which takes a combinatorial circuit computing a function F, and constructs a secure protocol computing F() and leaking no other information.

- This protocol is efficient for medium size circuits, but what about functions which cannot be represented as small circuits?

k^{th}-ranked element (e.g. median)

- Inputs:
  - Alice: S_A Bob: S_B
  - *Large* sets of unique items ($\in D$).
- Output:
  - $x \in S_A \cup S_B$ s.t. x has k-1 elements smaller than it.
- The rank k
  - Could depend on the size of input datasets.
  - Median: $k = (|S_A| + |S_B|) / 2$
- Motivation:
  - Basic statistical analysis of distributed data.
  - E.g. histogram of salaries in CS departments

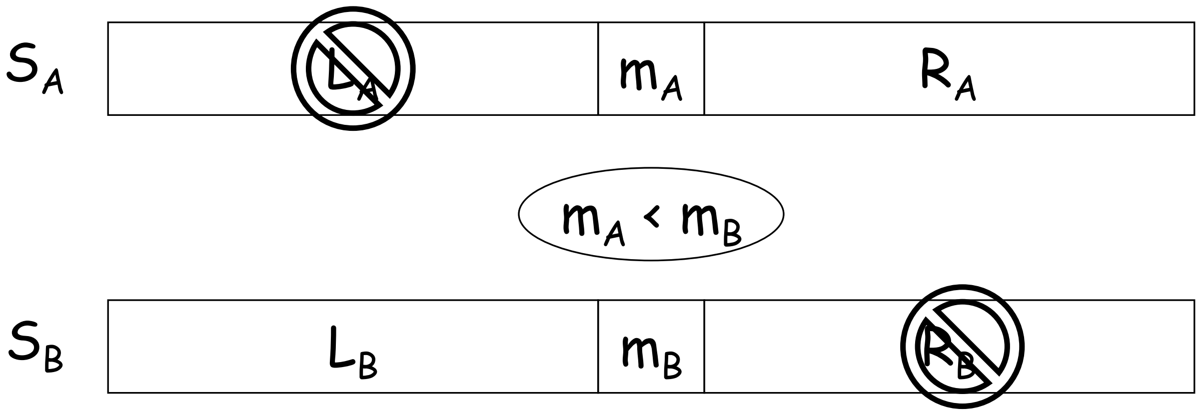
Secure computation in the case of large circuit representation

- The Problem:
  - The size of a circuit for computing the k^{th} ranked element are at least linear in k.
  - Generic constructions using circuits [Yao …] have communication complexity which is linear in the circuit size, and therefore in k.

- However, it is sometimes possible to design specific protocols for specific problems, and obtain a much better overhead.

- We will show such a protocol for computing the k^{th} ranked element, for the case of semi-honest parties.

An (insecure) two-party median protocol



L_A lies below the median, R_B lies above the median.

$|L_A| = |R_B|$

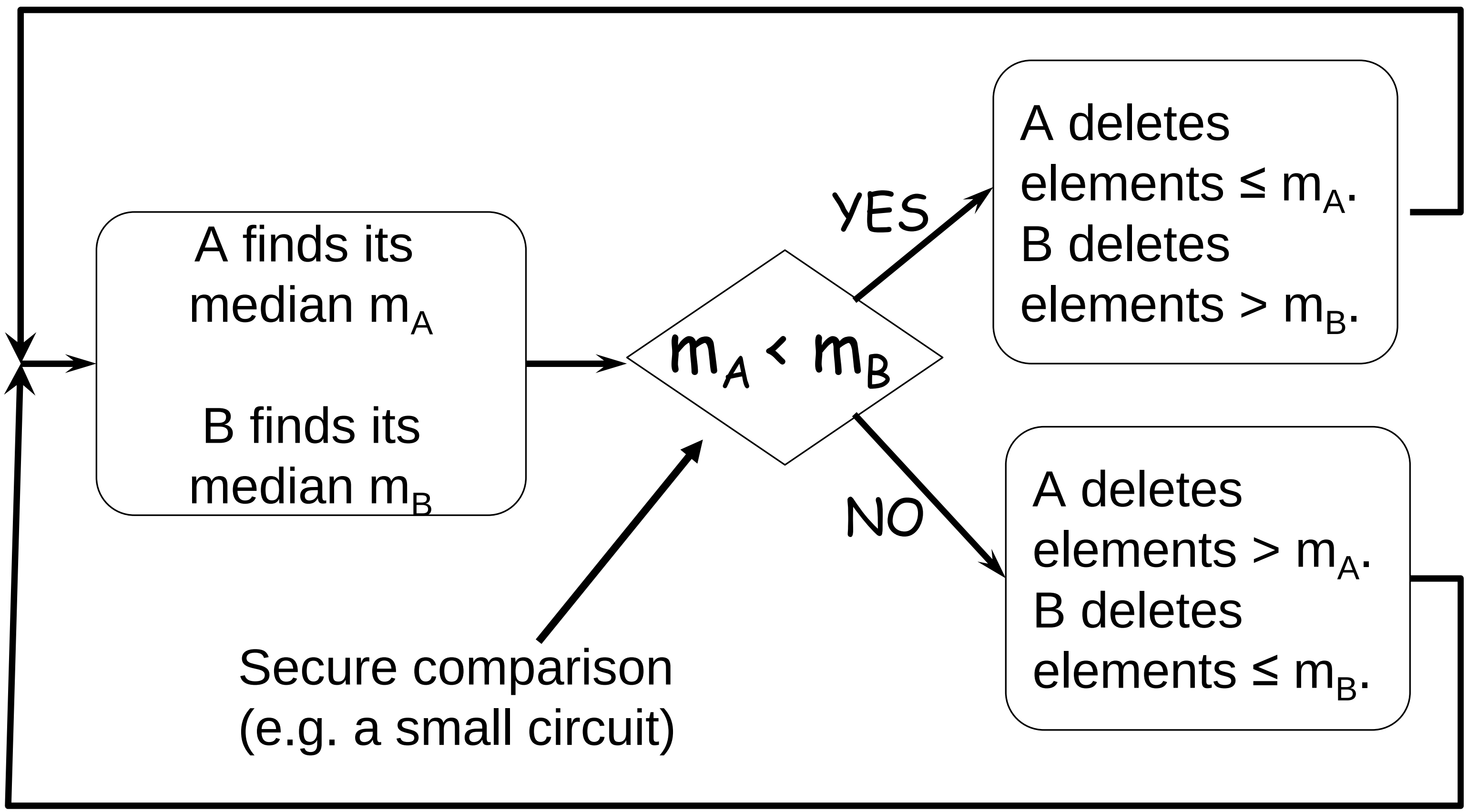
⇩

New median is same as original median.

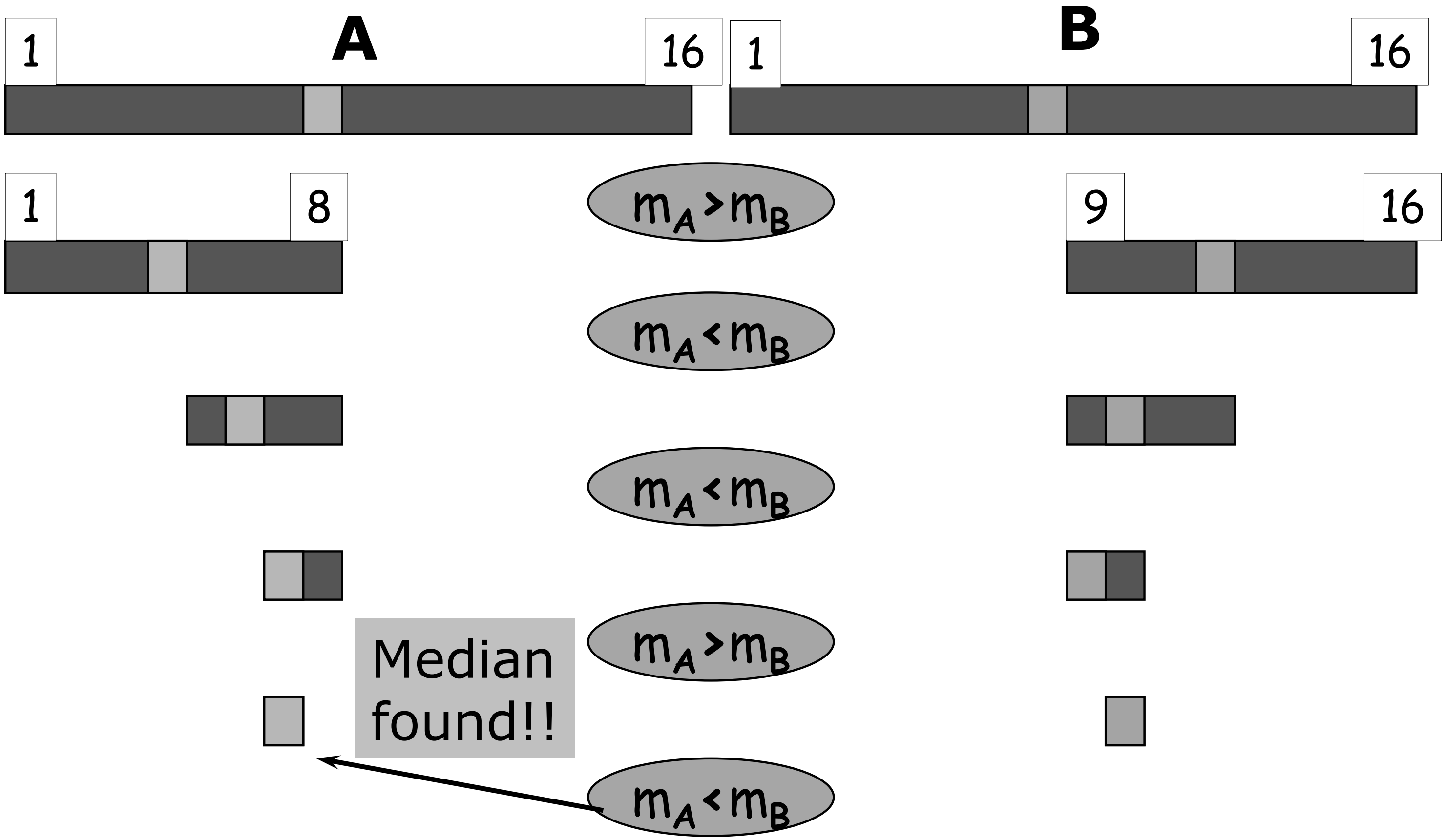
Recursion → Need log n rounds

(assume each set contains $n=2^i$ items)

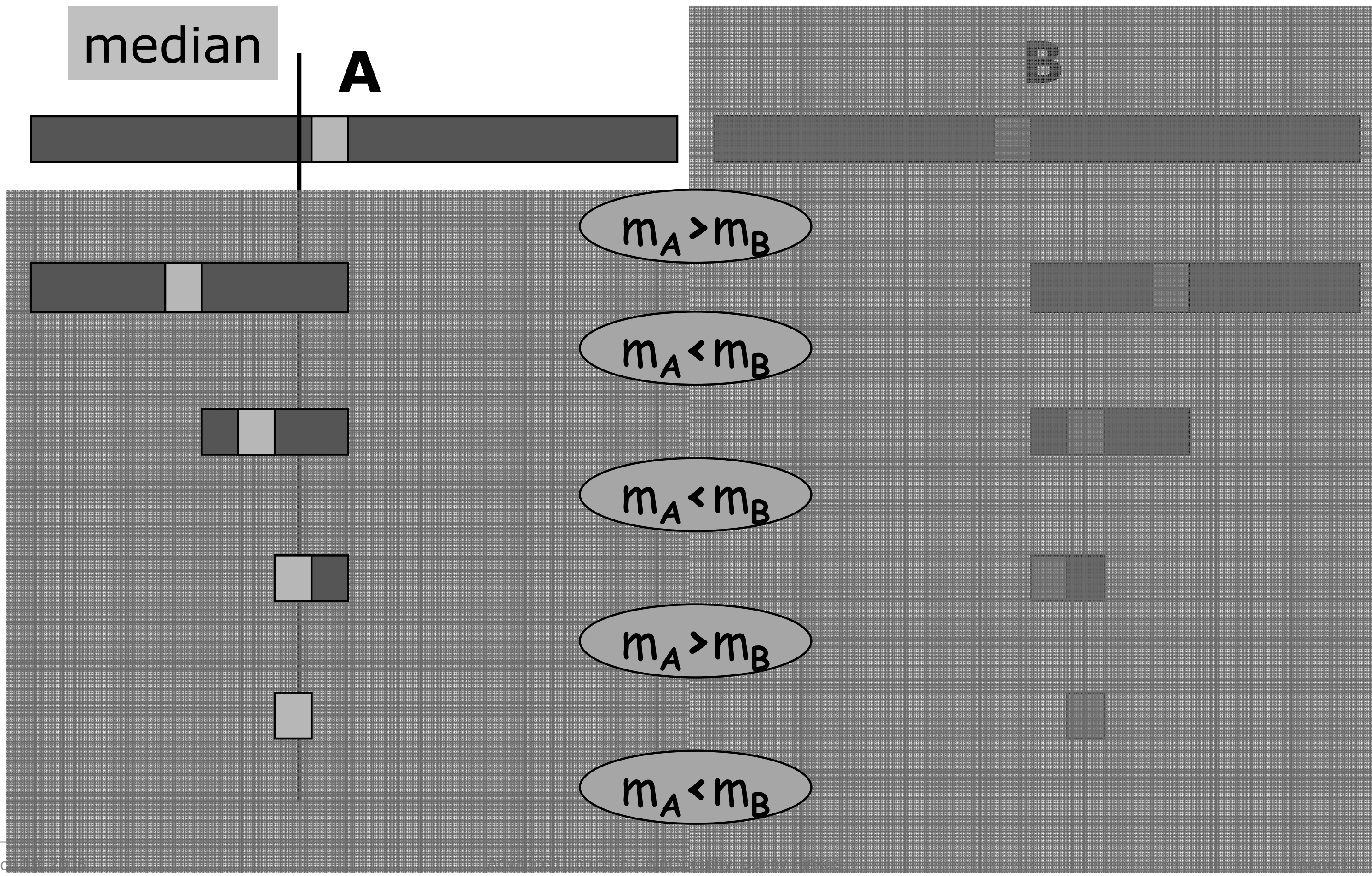
A Secure two-party median protocol



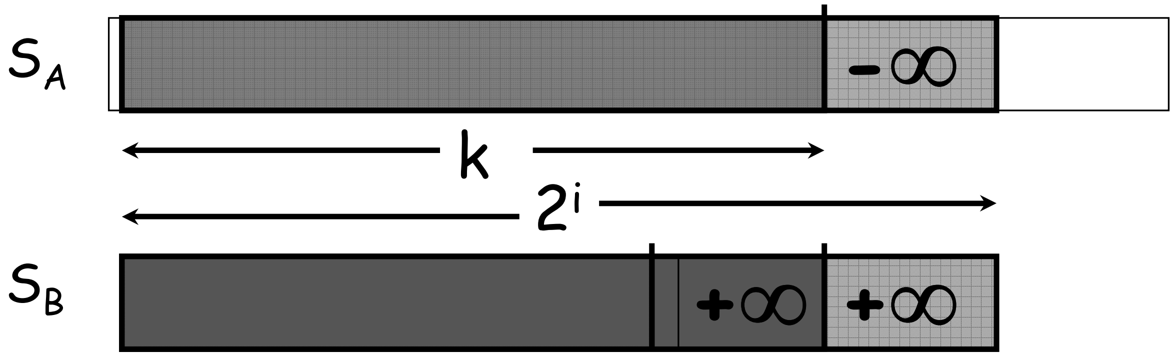
An example



Proof of security



Arbitrary input size, arbitrary k



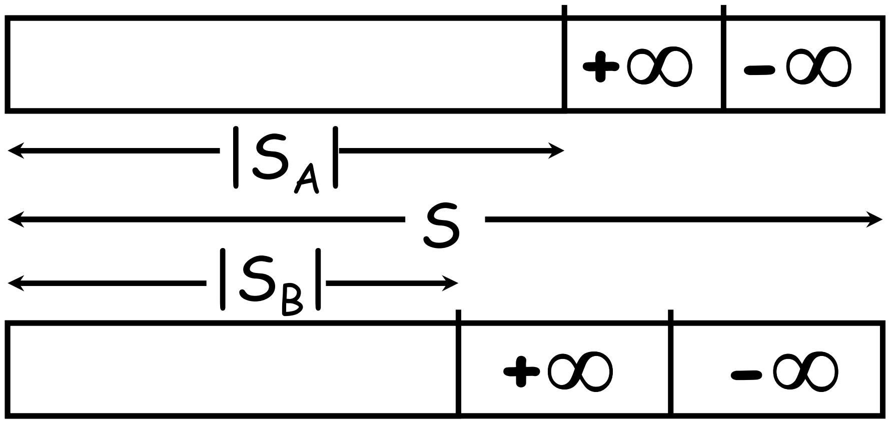
Now, compute the median of two sets of size k.

Size should be a power of 2.

median of new inputs = k^{th} element of original inputs

Hiding size of inputs

- Can search for k^{th} element without revealing size of input sets.
- However, k=n/2 (median) reveals input size.
- Solution: Let $S=2^i$ be a bound on input size.



Median of new datasets is same as median of original datasets.

Secure multi-party computation

- Problem statement:
  - n players $P_1, P_2, \ldots, P_n$
  - Player P_i has input x_i
  - There is a known function $f(x_1,\ldots,x_n) = (y_1,\ldots y_n)$
- Goals:
  - P_i should learn y_i, and nothing else (except for what can be computed from x_i and y_i)
  - This property should also hold for coalitions of corrupt parties (e.g., $P_1,\ldots,P_{n/3}$ should learn nothing but $x_1,\ldots,x_{n/3},y_1,\ldots,y_{n/3}$)
  - Security should hold even against malicious parties
- Examples…

More on MPC

- Generality: MPC is extremely general, covers all protocol problems.
- Adversaries:
  - Semi-honest vs. malicious
  - Static (decide in advance which parties to corrupt) vs. adaptive (decide on the fly which parties to corrupt)
  - Unbounded vs. probabilistic polynomial-time

Defining security

- It is not enough to list the desired properties that the protocol should satisfy
  - How can we be sure that we covered all properties?
- Basic security definition: comparison to an ideal scenario
  - In the ideal scenario there is a trusted party which receives $x_1,\ldots,x_n$, computes the function and sends y_i to P_i.
  - The real protocol is secure if its execution reveals no more than in the ideal scenario.
- The actual definition is much more complicated, in particular if we consider multiple invocations of the same protocol.

More on MPC

- Bounded corruption: We will consider scenarios where there is a bound on the number of parties which the adversary can corrupt.
  - Namely, there is a bound t and it is assumed that the adversary corrupts no more than t of the n parties.
- Synchronous network: communication proceeds in rounds. All messages sent in during a round are received during the same round.
- Adversarial power:
  - Information theoretic scenario: adversary cannot listen to communication channels, except those to/from parties it controls. (This does not make sense in the two-party case)
  - Cryptographic scenario: adversary sees all messages.

What is known

- Information theoretic scenario:
  - Semi-honest, adaptive adversary: any function can be computed iff adversary controls up to $t<n/2$ parties.
  - Malicious, adaptive adversary: any function can be computed iff adversary controls up to $t<n/3$ parties.
    - If broadcast is available, can withstand up to $t<n/2$.
- Cryptographic scenario:
  - Semi-honest, adaptive, polynomial-time adversary: assuming one-way trapdoor permutations exist, any function can be computed if $t<n$.
  - Malicious, adaptive, polynomial-time adversary: assuming one-way trapdoor permutations exist, any function can be computed if $t<n/2$.

An MPC protocol for semi-honest parties

- We will show a construction in the unconditional security scenario, against semi-honest, adaptive adversaries which control up to $t<n/2$ parties.

- The basic idea:
  - Any input value can be shared between the n participants, such that no t of them can reconstruct it.
  - It is possible to make computations on shared values.

- Initial step:
  - Write the function as an arithmetic circuit modulo a prime number p.
    - Note that arithmetic circuits can be much more compact than combinatorial (Boolean) circuits. For example, for computing $a+b$ or $a \cdot b$.

An MPC protocol for semi-honest parties

- The first step of the protocol:
  - Each P_i generates a (t+1)-out-of-n sharing of its input x_i
    - Namely, chooses a random polynomial $f_i()$ over Z_p^* such that $f_i(0)=x_i$.
    - Any subset of t shares does not leak any information about x_i
    - t shares enable to reconstruct x_i using polynomial interpolation
  - Every P_i sends to each P_j ($j \neq i$) the value $f_i(j)$

- The protocol continues by induction from the input wires to the output wires.
  - We will show that for every gate, if the parties know shares of the input values, they can compute shares of the output values.

Computation stage

- All parties participate in the computation of every gate
- Addition gate: c= a+b
  - The parties must generate a sharing of c.
  - Namely, there should be a polynomial $f_c()$ of degree t, such that $f_c()$ is random except for $f_c(0)=c$
  - and each P_i has the share $c_i=f_c(i)$
- The protocol:
  - Each player P_i already has shares of a and b.
  - Namely, shares $a_i=f_a(i)$ and $b_i=f_b(i)$ of polynomials $f_a()$ and $f_b()$ of degree t, for which $f_a(0)=a$ and $f_b(0)=b$.
  - P_i sets $c_i=a_i+b_i = f_a(i)+f_b(i) = f_c(i)$
  - No communication is needed for this computation.

Computation stage: multiplication gate

- Each player P_i already has shares $a_i=f_a(i)$ and $b_i=f_b(i)$.
- Needs to have a share d_i of $d=a \cdot b$.
- First attempt:
  - P_i sets $d_i=a_i \cdot b_i = f_d(i)$.
  - Obtains a share of $f_a() \cdot f_b()$
  - Indeed, $f_d(0) = d = a \cdot b$.
  - But $f_d()$ is of degree 2t and not t.
    - If we do this twice, the degree becomes 4t>n…

Computing multiplication gates

- P_i sets $d_i=a_i \cdot b_i = f_d(i)$.
- $f_d(i)$ is of degree $2t < n$.
- We know that there are (Lagrange) coefficients $r_1,..,r_n$ such that $d=f_d(0)=a \cdot b= r_1 d_1+\ldots+r_n d_n = r_1 f_d(1)+\ldots+r_n f_d(n)$.

- Each P_i creates a random polynomial g_i of degree t such that $g_i(0)=d_i$.
- Consider $g(x)=\sum_{i=1}^{n} r_i \cdot g_i(x)$
  - This a polynomial of degree t.
  - $g(0) = \sum_{i=1}^{n} r_i \cdot g_i(0) = \sum_{i=1}^{n} r_i \cdot d_i = d$.
- Now, if only we could provide each P_i with g(i)…

Computing multiplication gates

- Each P_i creates a random polynomial g_i of degree t such that $g_i(0)=d_i$
- We need the parties to share $g(x)=\sum_{i=1}^{n} r_i \cdot g_i(x)$

- P_i sends to every P_j the value $g_i(j)$
- Every P_j receives $g_1(j),\ldots,g_n(j)$, and computes $g_j = \sum_{i=1}^{n} r_i \cdot g_i(j) = g(j)$

- This is the desired share of $a \cdot b$

Opening the outputs

- At the end of the circuit, for each value y_i it holds that the parties hold shares of a polynomial f(x) of degree t such that $f(0)=y_i$.
- Each party P_j sends f(j) to P_i.
- P_i interpolates $f(0)=y_i$.

Properties

- Correctness: straightforward
- Privacy: For every set of t players, it holds that all values they see in the protocol are shares of (t+1)-out-of-n secret sharing schemes, and therefore all their t shares are uniformly distributed.
  – The proof needs to make sure that this property holds even if adversary gets shares of a,b, and $a \cdot b$

- Overhead:
  – $O(n^2)$ messages for every multiplication gate.
  – Communication rounds linear in the depth of the circuit (where only multiplication gates are counted)