



Session 4: Unconditionally Secure Multi-Party Computation

Benny Pinkas
Bar-Ilan University



Bar-Ilan University
Dept. of Computer Science

► What is N ?

Overview



Bar-Ilan University
Dept. of Computer Science

- ▶ “Completeness theorems for non-cryptographic fault-tolerant distributed computation”
 - M. Ben-Or, S. Goldwasser, A. Wigderson, 1988.
 - Published concurrently with “Multiparty unconditionally secure protocols” Chaum, Crepau, Damgard.
- ▶ Published after the results of Yao and GMW, with the motivation of obtaining results without any intractability assumptions.

Overview



Bar-Ilan University
Dept. of Computer Science

- ▶ “Completeness theorems for non-cryptographic fault-tolerant distributed computation”
 - M. Ben-Or, S. Goldwasser, A. Wigderson, 1988.
- ▶ The setting
 - A complete synchronous network of n parties
 - Each party P_i has an input x_i
 - Communication channels between parties are secure
 - The solution for the malicious case requires a **broadcast** channel

Overview (contd.)

- ▶ The function $f(x_1, \dots, x_n)$ is represented by an **arithmetic** circuit over a field F (say, modulo a large prime)
 - Contains addition and multiplication gates in F
 - Can be more compact than a Boolean circuit
 - We need only care about deterministic functionalities:
 - A randomized functionality $f(r; x_1, \dots, x_n)$ can be computed by each party providing (r_i, x_i) , and the circuit computing and using $r = r_1 \oplus \dots \oplus r_n$.

Overview (contd.)

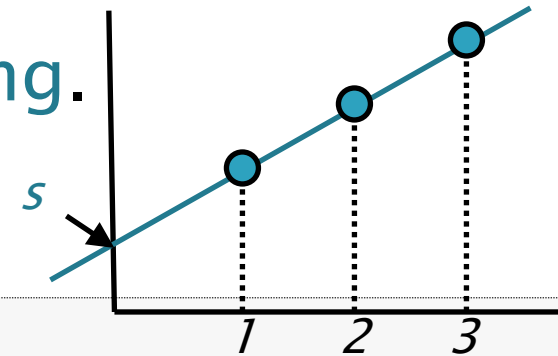


Bar-Ilan University
Dept. of Computer Science

- ▶ The construction provides unconditional security
 - Against semi-honest adversaries controlling $t < n/2$ parties
 - Against malicious adversaries controlling $t < n/3$ parties
- ▶ Unlike the GMW construction, which is based on cryptographic assumptions
 - oblivious transfer
 - ZK proofs

Main tool – secret sharing

- ▶ t -out-of- n secret sharing
- ▶ Given a secret s , provide shares to n parties, s.t.
 - Any t shares enable the reconstruction of the secret
 - Any $t-1$ shares reveal nothing about the secret
- ▶ Consider 2-out-of- n secret sharing.
 - Define a line which intersects the Y axis at S
 - The shares are points on the line
 - Any two shares define S
 - A single share reveals nothing



t-out-of-n secret sharing

- ▶ **Fact:** Let F be a field. Any $d+1$ pairs (a_i, b_i) define a unique polynomial P of degree $\leq d$, s.t. $P(a_i) = b_i$. (assuming $d < |F|$).
- ▶ **Shamir's secret sharing scheme:**
 - The secret S is an element in a field (say, in \mathbb{Z}_p).
 - Define a polynomial P of degree $t-1$ by choosing random coefficients a_1, \dots, a_{t-1} and defining
$$P(x) = a_{t-1}x^{t-1} + \dots + a_1x + S.$$
 - The share of party P_j is $(j, P(j))$.

t-out-of-n secret sharing

► Reconstructing the secret:

- Assume we have $P(x_1), \dots, P(x_t)$.
- Use **Lagrange interpolation** to compute the **unique** polynomial of degree $\leq t-1$ which agrees with these points.
- Output the **free coefficient** of this polynomial.

► Lagrange interpolation

- $P(x) = \sum_{i=1..t} P(x_i) \cdot L_i(x)$
- where $L_i(x) = \prod_{j \neq i} (x - x_j) / \prod_{j \neq i} (x_i - x_j)$
(Note that $L_i(x_i) = 1$, $L_i(x_j) = 0$ for $j \neq i$.)

Properties of Shamir's secret sharing



Bar-Ilan University
Dept. of Computer Science

- ▶ **Perfect secrecy:** Any $t-1$ shares give no information about the secret. $\Pr(\text{secret}=s \mid P(1), \dots, P(t-1)) = \Pr(\text{secret}=s)$.
- ▶ **Proof:**
 - Intuition from 2-out-of- n secret sharing:
 - The polynomial is generated by choosing a random coefficient a and defining $P(x) = a \cdot x + s$.
 - Suppose that the adversary knows the share $P(1) = a \cdot 1 + s$.
 - For any value of s , there is a one-to-one correspondence between a and $P(1)$ ($a = P(1) - s$).
 - Since a is uniformly distributed, so is $P(1)$.
 - Therefore $P(1)$ does not reveal any information about s .

Properties of Shamir's secret sharing



Bar-Ilan University
Dept. of Computer Science

- ▶ **Perfect secrecy:** Any $t-1$ shares give no information about the secret.
- ▶ Proved by showing that, even given S , any $t-1$ shares are uniformly distributed.
- ▶ **Proof:**
 - The polynomial is generated by choosing a random polynomial of degree $t-1$, subject to $P(0)=S$.
 - Suppose that the adversary knows the shares $P(1), \dots, P(t-1)$.
 - The values of $P(1), \dots, P(t-1)$ are defined by an invertible set of $t-1$ linear equations of a_1, \dots, a_{t-1}, S .
 - $P(i) = \sum_{j=1, \dots, t-1} (i)^j a_j + s$.

Properties of Shamir's secret sharing



Bar-Ilan University
Dept. of Computer Science

► Proof (cont.):

- The values of $P(1), \dots, P(t-1)$ are defined by an invertible set of $t-1$ linear equations of a_1, \dots, a_{t-1}, s .
 - $P(x_i) = \sum_{j=1, \dots, t-1} (i)^j a_j + s$.
- For any possible value of s , there is a exactly one set of values of a_1, \dots, a_{t-1} which gives the values $P(1), \dots, P(t-1)$.
 - This set of a_1, \dots, a_{t-1} can be found by solving a linear system of equations.
- Since a_1, \dots, a_{t-1} are uniformly distributed, so are the values of $P(x_1), \dots, P(x_{t-1})$.
 - $\Rightarrow P(x_1), \dots, P(x_{t-1})$ reveal nothing about s .

Additional properties of Shamir's secret sharing



Bar-Ilan University
Dept. of Computer Science

- ▶ Ideal size:
 - Each share is the same size as the secret.
- ▶ Homomorphic property:
 - Suppose $P(1), \dots, P(n)$ are shares of S ,
and $P'(1), \dots, P'(n)$ are shares of S' ,
then $P(1)+P'(1), \dots, P(n)+P'(n)$ are shares of $S+S'$.

The BGW protocol

- ▶ Input sharing phase
- ▶ Computation phase
- ▶ Output reconstruction phase

- ▶ Main idea:
 - for every wire, the parties will know a secret sharing of the value which passes through that wire.

BGW protocol – input phase

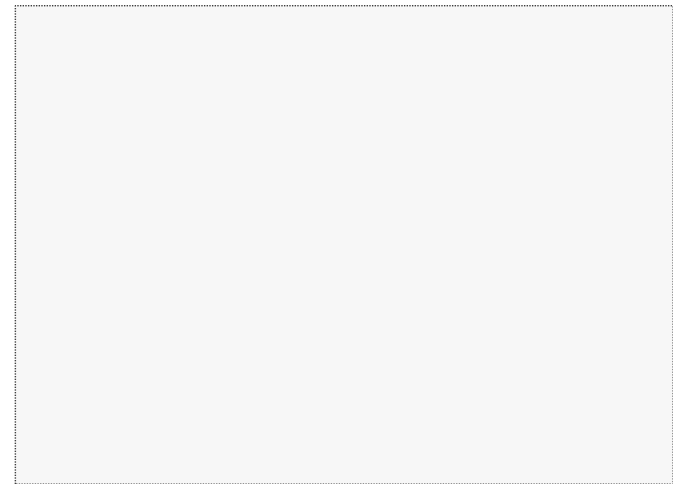
- ▶ Let $t < n/2$ be a bound on the number of corrupt parties.
- ▶ Each P_i generates a $(t+1)$ -out-of- n sharing of its input x_i .
 - Namely, chooses a polynomial $f_i()$ over F , s.t. $f_i(0) = x_i$
 - Any subset of t shares does not leak any information about x_i
 - $t+1$ shares reveal x_i
- ▶ P_i sends to each P_j the value $f_i(j)$.
- ▶ The protocol continues from the input wires to the output wires.

Computation phase

- ▶ All parties participate in the computation of every gate
 - already know a sharing of its input wires
 - Must generate a sharing of the output wire
- ▶ Addition gate: $c = a + b$
 - Must generate a polynomial $f_c()$ of degree t , which is **random** except for $f_c(0) = a + b$. Each P_i learns $f_c(i)$.
 - Define $f_c(\cdot) = f_a(\cdot) + f_b(\cdot)$
 - Each P_i sets $c_i = a_i + b_i = f_a(i) + f_b(i) = f_c(i)$
 - No interaction is needed!
- ▶ What about multiplication gates?

Output phase

- ▶ Easier to describe than the protocol for multiplication gates
- ▶ **Output wires**
 - If output wire y_i must be learned by P_i , then all parties send it their shares of y_i .
 - P_i reconstructs the secret and learns the output value.



Computation phase – multiplication gates



Bar-Ilan University
Dept. of Computer Science

- ▶ $c = a \cdot b$. First attempt:
 - Define $f_{ab}(\cdot) = f_a(\cdot) \cdot f_b(\cdot)$.
 - Each P_i computes $a_i \cdot b_i = f_a(i) \cdot f_b(i) = f_{ab}(i)$.
 - Indeed, $f_{ab}(0) = a \cdot b$.
 - But the degree of f_{ab} is $2t$,
and f_{ab} is not a random polynomial.
- ▶ Interpolation:
 - f_{ab} is of degree $2t < n$, and $f_{ab}(0) = a \cdot b$.
 - Therefore \exists coefficients r_1, \dots, r_n s.t.
$$f_{ab}(0) = a \cdot b = r_1 f_{ab}(1) + \dots + r_n f_{ab}(n) =$$
$$r_1 \cdot a_1 b_1 + \dots + r_n \cdot a_n b_n.$$
 - Each r_i is easily computable.

Computation phase – multiplication gates



Bar-Ilan University
Dept. of Computer Science

- ▶ Each P_i
 - Has $a_i \cdot b_i$
 - Creates a random polynomial $g_i(\cdot)$ of degree t s.t.
 $g_i(0) = a_i \cdot b_i$
- ▶ Consider $g(x) = \sum_{i=1 \dots n} r_i \cdot g_i(x)$
 - of degree t
 - $g(0) = \sum_{i=1 \dots n} r_i \cdot g_i(0) = \sum_{i=1 \dots n} r_i \cdot a_i b_i = a \cdot b$.
 - This is exactly the polynomial we need.
 - Must provide each P_i with a share of $g()$.

Computation phase – multiplication gates



Bar-Ilan University
Dept. of Computer Science

- ▶ Each P_i
 - Creates a random polynomial $g_i(\cdot)$ of degree t s.t.
 $g_i(0) = a_i \cdot b_i$
 - Define $g(x) = \sum_{i=1 \dots n} r_i \cdot g_i(x)$, of degree t .
 $g(0) = \sum_{i=1 \dots n} r_i \cdot g_i(0) = a \cdot b$.
- ▶ P_i sends to every P_j the value $g_i(j)$
- ▶ Every P_j receives $g_1(j), \dots, g_n(j)$, computes $g(j) = \sum_{i=1 \dots n} r_i \cdot g_i(j)$
- ▶ This is the desired sharing of $a \cdot b$.

Properties



Bar-Ilan University
Dept. of Computer Science

- ▶ **Correctness is straightforward**
- ▶ **Overhead:**
 - $O(n^2)$ messages for every multiplication gate
 - # of rounds linear in depth of circuit (where only multiplication gates count)

Security



Bar-Ilan University
Dept. of Computer Science

- ▶ **Main idea:** every set of t players, receives in each round values which are t -wise independent, and therefore uniformly distributed.
 - Therefore no information about the actual wire values are leaked.

Simulation based proof

- ▶ Recall what we showed
 - In $(t+1)$ -out-of- n secret sharing, any t shares are uniformly distributed, independently of the secret.

- ▶ Suppose first that multiplication is computed by an oracle (this is the f_{mult} hybrid model)
 - The simulator obtains the inputs and outputs of the t corrupt parties
 - The transcript of a party includes its input, randomness used, all messages received.

Simulation based proof

- ▶ Adversary controls a set J of $t < n/2$ parties.
- ▶ The simulator:
 - $\forall P_i \in J$, set input $z_i = x_i$. $\forall P_i \notin J$, set input $z_i = 0$.
 - Share inputs z_i according to protocol.
 - Addition gates: add shares as in protocol.
 - Mult gates: provide $P_i \in J$ with shares of a random sharing of the value 0.
- Simulation is correct since t shares of any value are uniformly distributed.

Simulation based proof

- ▶ Output stage:
 - ▶ \forall wire, the simulator already defined shares for all $P_i \in J$.
 - ▶ Let w be an output wire of $P_i \in J$. The simulator has the output value y_w , and the t shares of $P_i \in J$.
 - ▶ The simulator interpolates the t -degree polynomial f_w going through these values. It then simulates receiving the shares $f_w(i)$ from all $P_i \notin J$.
 - ▶ Let w be an output wire of $P_j \notin J$. For all $P_i \in J$, the simulator sends the corresponding share to P_j .

Simulating the multiplication protocol



Bar-Ilan University
Dept. of Computer Science

- ▶ Recall, the multiplication protocol
 - P_i creates a random poly $g_i(\cdot)$ of deg t s.t. $g_i(0) = a_i \cdot b_i$
 - P_i sends to $\forall P_j$ the value $g_i(j)$, and receive shares $g_j(i)$
 - P_i computes its share as $g(i) = \sum_{j=1 \dots n} r_j \cdot g_j(i)$.
 -
- ▶ Simulation $\forall P_i \in J$:
 - Create a random poly $g_i(\cdot)$ of deg t s.t. $g_i(0) = P_i$'s share
 - Send to every P_j the value $g_i(j)$
 - $\forall P_j \notin J$ simulate receipt of a random share $g_j(i)$
 - Compute share of wire value as $g(i) = \sum_{j=1 \dots n} r_j \cdot g_j(i)$

Security against malicious parties

- ▶ Aka security against Byzantine adversaries
- ▶ Possible problems in using the previous protocol:
 - When sharing its input, P_i might send values of a polynomial of degree **greater than t** .
 - As a result, different subsets of the clients might recover different values as the secret.
 - Parties might send **incorrect shares**
 - How can we interpolate in this case?
- ▶ Protocol secure against $t < n/3$

Major tool – Verifiable Secret Sharing (VSS)



Bar-Ilan University
Dept. of Computer Science

► Sharing stage

- Add elements to the shares so that parties are assured to receive values of a polynomial of degree t (even if the dealer is malicious)

► Recovery stage

- As long as $t < n/3$ shares are corrupt, use error correction techniques to recover the secret.
- Based on the fact that Shamir's secret sharing scheme is a **Reed–Solomon code**, which can correct up to $t < n/3$ errors.

The Reed–Solomon code

► Reed–Solomon code

- A linear $[n,k,d]$ -code, with $k=t+1$, and $d=n-t$.
- The message is (m_0, \dots, m_t) .
- Use it as the coefficients of a degree t polynomial, P_m .
- Codeword is $\langle P_m(1), \dots, P_m(n) \rangle$.
- Two codewords differ in at least $d=n-t$ locations.
- \exists efficient decoding correcting $(n-t-1)/2$ errors.
- If $t < n/3$, correcting up to t errors.

Using the Reed–Solomon code

► Usage:

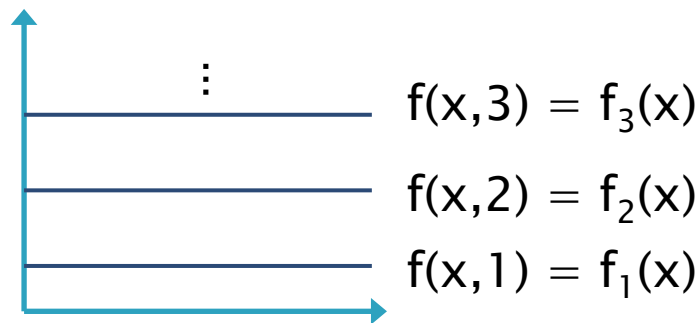
- Let $P()$ be a polynomial of degree t . (E.g., the polynomial used for $(t+1)$ -out-of- n secret sharing.)
- If instead of receiving $\langle P(1), P(2), \dots, P(n) \rangle$, we receive up to $t < n/3$ corrupt values, can still recover P . (And in particular, recover $P(0)$, the secret.)

► Conclusion:

- Can easily handle corrupt parties which send corrupt shares.
- Need to **focus** on forcing the dealer to distribute shares **consistent** with a t -degree polynomial.

Bivariate polynomials

- ▶ $f(x,y) = \sum_{i=0 \dots t} \sum_{j=0 \dots t} a_{i,j} \cdot x^i \cdot y^j$
- Defined by $(t+1)^2$ coefficients
- Claim: $f(x,y)$ can be defined by $t+1$ univariate polynomials:
 - Given $t+1$ polynomials of degree t : $f_1(x), \dots, f_{t+1}(x)$ there exists a single bivariate polynomial of degree t such that $f(x,1)=f_1(x), \dots, f(x,t+1)=f_{t+1}(x)$



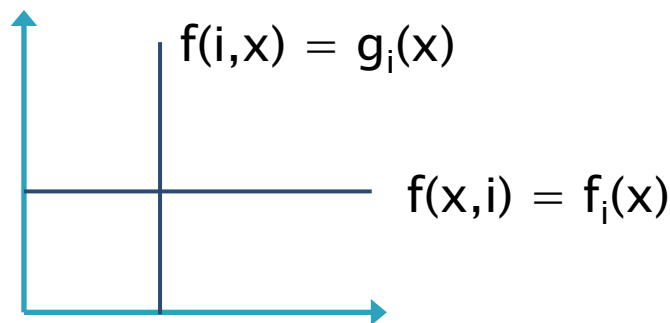
VSS using Bivariate polynomials – Step 1

$(t+1)$ -out-of- n secret sharing



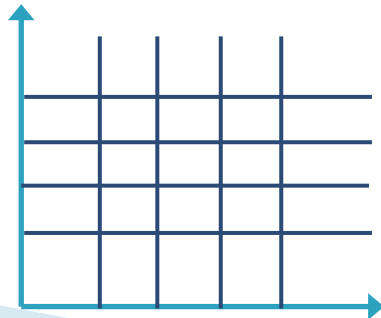
Bar-Ilan University
Dept. of Computer Science

- ▶ Dealer defines a **random** $f(x,y)$ of degree t , s.t. $f(0,0) = \text{secret}$.
- ▶ Sends to P_i the **share** $f_i(x) = f(x,i)$. (t -deg poly)
 - By the claim, any $t+1$ shares suffice to reveal secret.
- ▶ Sends to P_i the **dual share** $g_i(x) = f(i,x)$.
 - Will be used for checking shares received from other parties



VSS using Bivariate polynomials

- ▶ Claim: \forall subset J of size t , the shares and dual shares of $P_i \in J$ do not reveal the secret.
 - Assume wlog $J = 1, 2, \dots, t$.
 - $f_1(x), \dots, f_t(x)$, each of degree t , enforce $t \cdot (t+1)$ constraints of the bivariate polynomial f .
 - $g_1(x), \dots, g_t(x)$, each add another constraint.
 - Total # of constraints is $t(t+1) + t = t^2 + 2t = (t+1)^2 - 1$.
None of them defines $f(0,0)$ directly.



VSS using Bivariate polynomials – Step 2

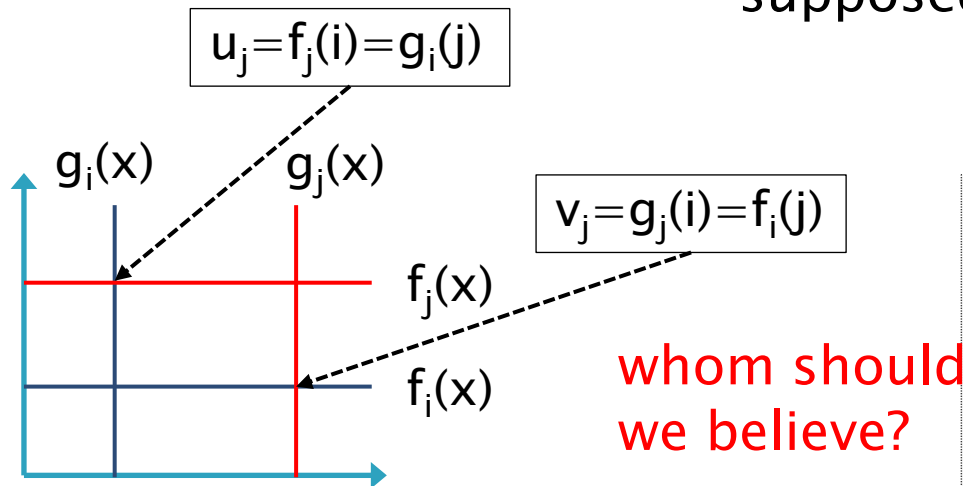


Bar-Ilan University
Dept. of Computer Science

► Each party P_i :

- $\forall j$, send $f_i(j)$ and $g_i(j)$ to P_j .
- $\forall j$, let (u_j, v_j) the values received from P_j .

If $u_j \neq g_i(j)$ or $v_j \neq f_i(j)$, then **broadcast**
“complaint($i, j, f_i(j), g_i(j)$)”. (the two values P_i was supposed to receive)



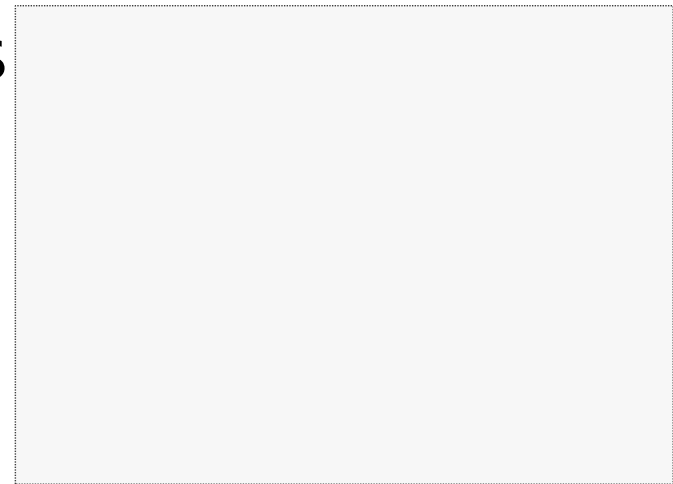
VSS using Bivariate polynomials – Step 3



Bar-Ilan University
Dept. of Computer Science

- ▶ The dealer:
 - Upon receiving the message “complaint($i, j, f_i(j), g_i(j)$)” sent by P_i ,
check that $f_i(j)=f(i,j)$ and that $g_i(j)=f(j,i)$.
 - If the checks fail, broadcast $\text{reveal}(i, f_i(x), g_i(x))$.
 - (Namely, if P_i sent an incorrect complaint,
broadcast the shares that it received from dealer.)

- ▶ Now, whom should the parties believe, P_i or the dealer?



VSS using Bivariate polynomials – Step 4



Bar-Ilan University
Dept. of Computer Science

- ▶ Each P_i
 1. If P_i views two messages $\text{complaint}(k, j, u_1, v_1)$ and $\text{complaint}(j, k, u_2, v_2)$, and the dealer did not broadcast a corresponding reveal message, go to 3.
 2. If P_i views a message $\text{reveal}(j, f_j(x), g_j(y))$, check if it agrees with P_i 's shares: $f_i(j) = g_j(i)$ and $g_i(j) = f_j(i)$. If the check succeeds, broadcast “good”.
 3. If at least $n-t$ parties broadcasted “good” then output $f_i(0)$ as your share of the secret. Otherwise abort.

VSS Security proof – Sketch

► Assume dealer is **honest**

- An **honest** P_j **complains** only if a corrupt P_i sends it incorrect values. But since the **complaint** of P_i contains good values, the dealer does not reveal P_j 's share.
- If a **corrupt** P_i complains with incorrect values, dealer sends a **reveal message** of P_i 's shares, which passes the test of the **$n-t$** honest parties, which then send **$n-t$ good** messages and therefore output the correct shares.

VSS Security proof – Sketch

► Assume dealer is **corrupt**

- Suppose P_i, P_k are honest and receive inconsistent shares: $f_j(k) \neq g_k(j)$, or $g_j(k) \neq f_k(j)$.
- Both parties complain, and therefore dealer must send **reveal** message or else no honest party broadcasts **good**.
- The shares are output only if $n-t$ parties output “good”. At least $(n-t)-t=t+1$ of them are honest.
- Their polynomials agree with those revealed by the dealer.
- These $t+1$ polynomials define a unique bivariate poly, which defines the secret.
- That’s all that we need.

The full protocol

- ▶ Inputs are shared using VSS.
 - Therefore dealer deals consistent shares.
- ▶ Addition gates are trivial.
- ▶ Multiplication gates:
 - Must ensure that each party multiplies its own shares.
 - Must use a VSS to perform the sharing defined by the protocol.
 - The full description and proof are quite intricate.

Overhead



Bar-Ilan University
Dept. of Computer Science

- ▶ No public key operations are needed!
- ▶ Input sharing step is more complicated than in the semi-honest case
 - Length of messages increases by $O(n)$
 - But this protocol is run only once, and has $O(1)$ rounds.
- ▶ Multiplication gates
 - Requires the use of a VSS
 - Message length increases by $O(n)$

