Secure Multiparty Computation (MPC)

Yaman Yağız Taşbağ

Outline

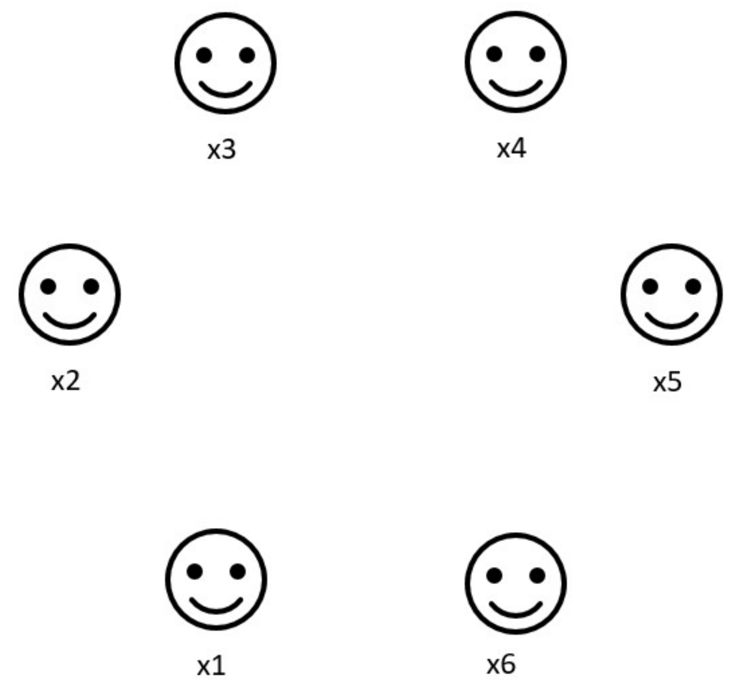
- Introduction and Basic Security Definition
- Shamir Secret Sharing Scheme and BGW protocol
- Linear Secret Sharing Schemes
- Oblivious Transfer
- GMW Protocol

What MPC is trying to solve

Assume n Parties are trying to compute a function $f(x_1, x_2 \cdots x_n)$ together with their private inputs $x_1, x_2 \cdots x_n$ while not revealing anything but the output of the function.

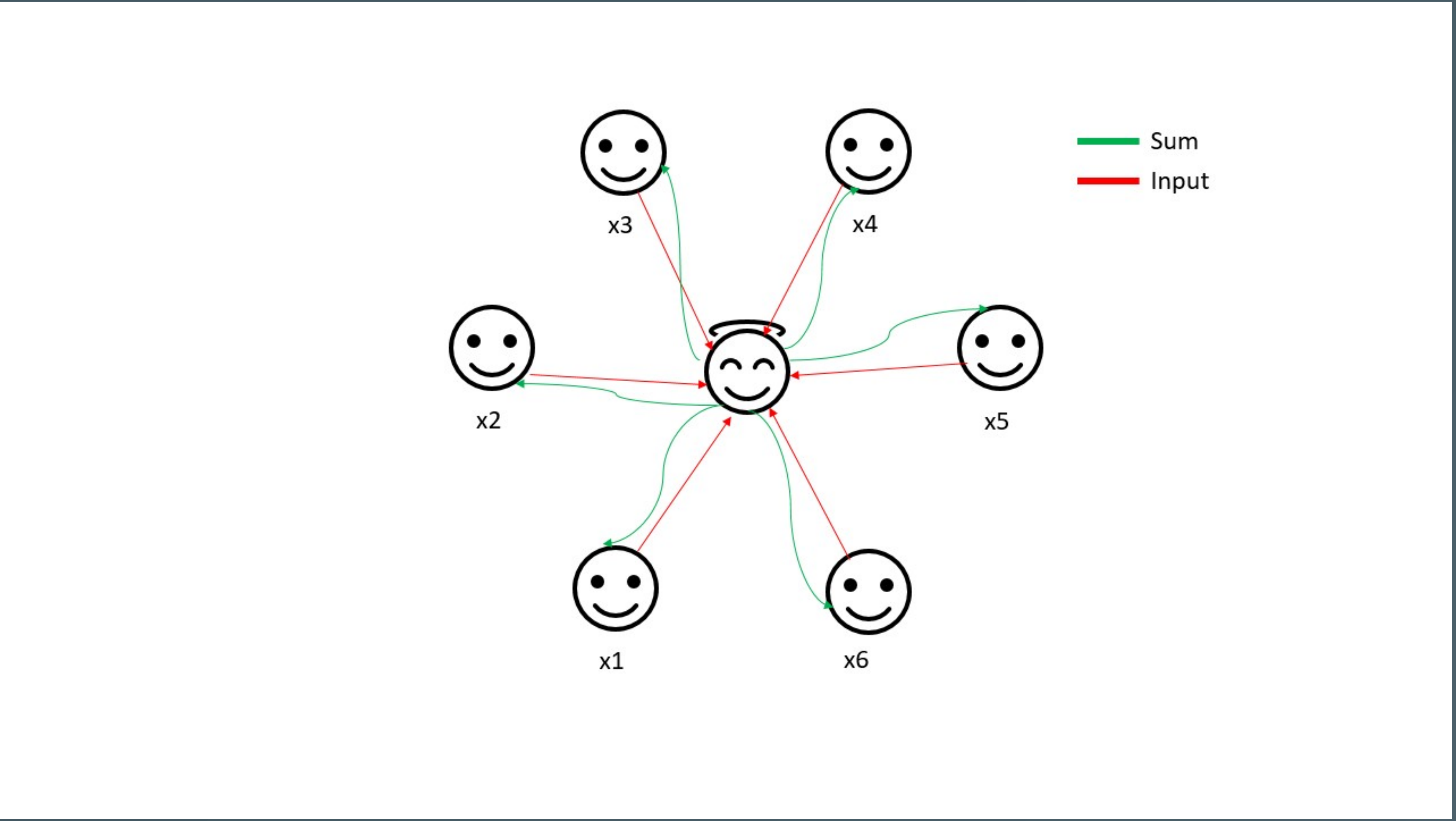
Example #1

Sum of incomes



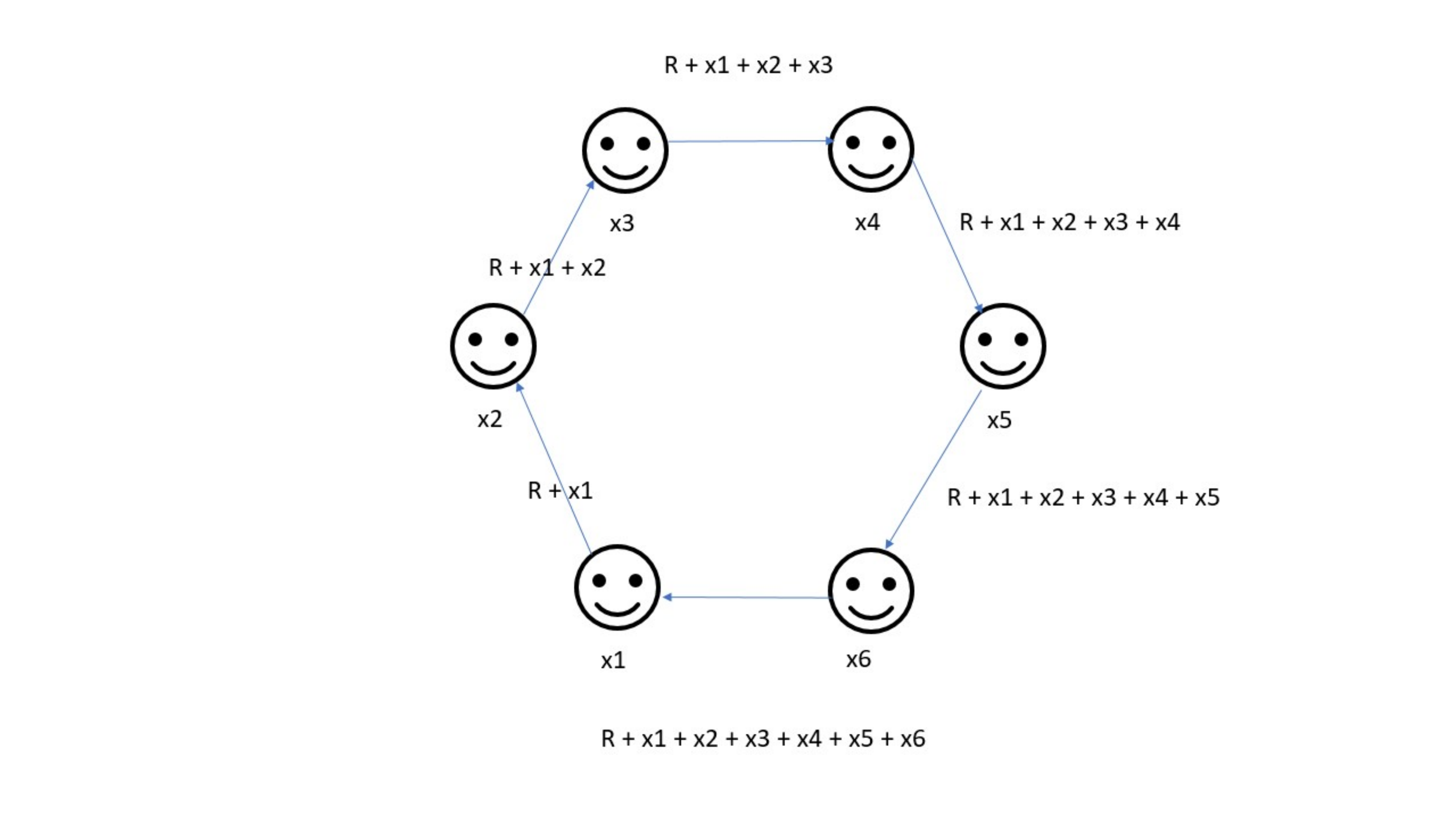
Example #1

In an ideal world



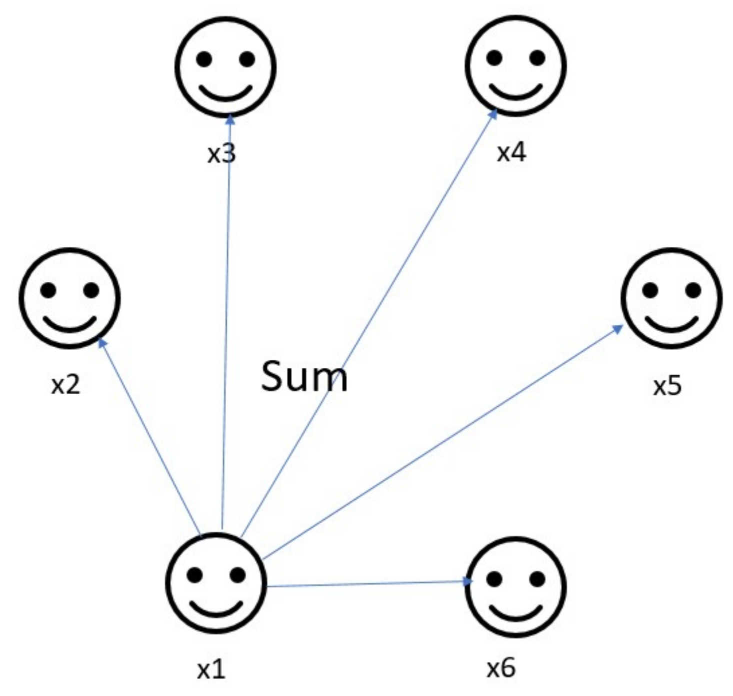
Example #1

One Solution:



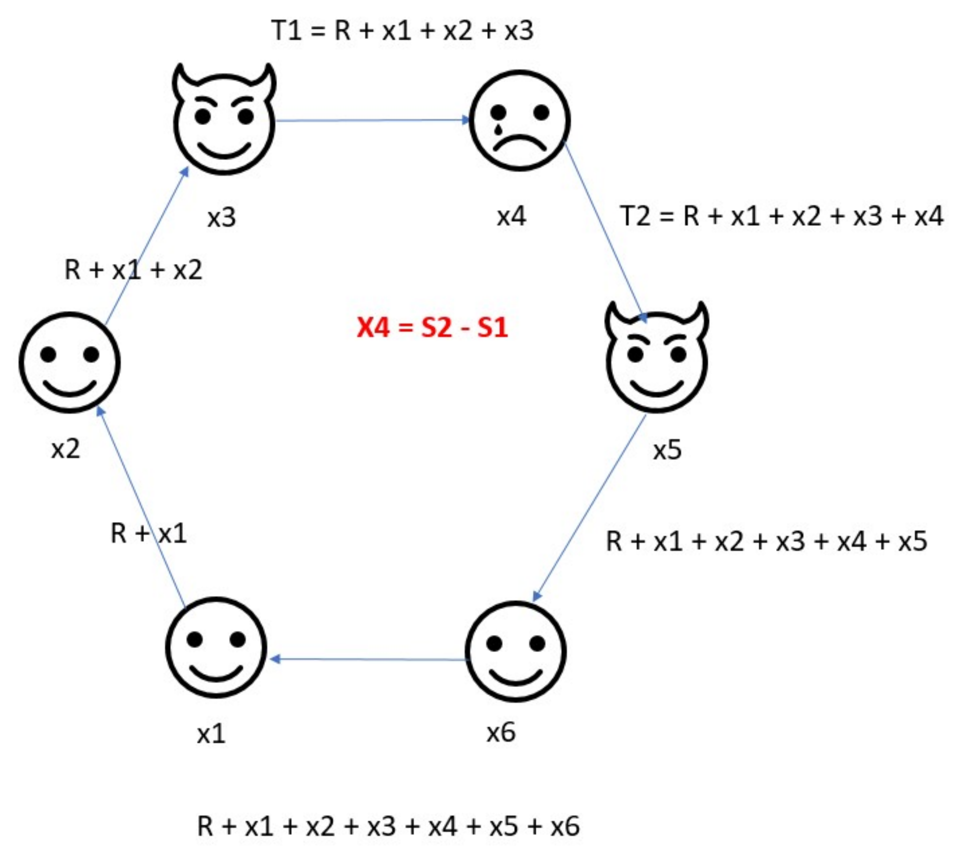
From now on all operations are in mod p

Example #1



Is it **secure?**

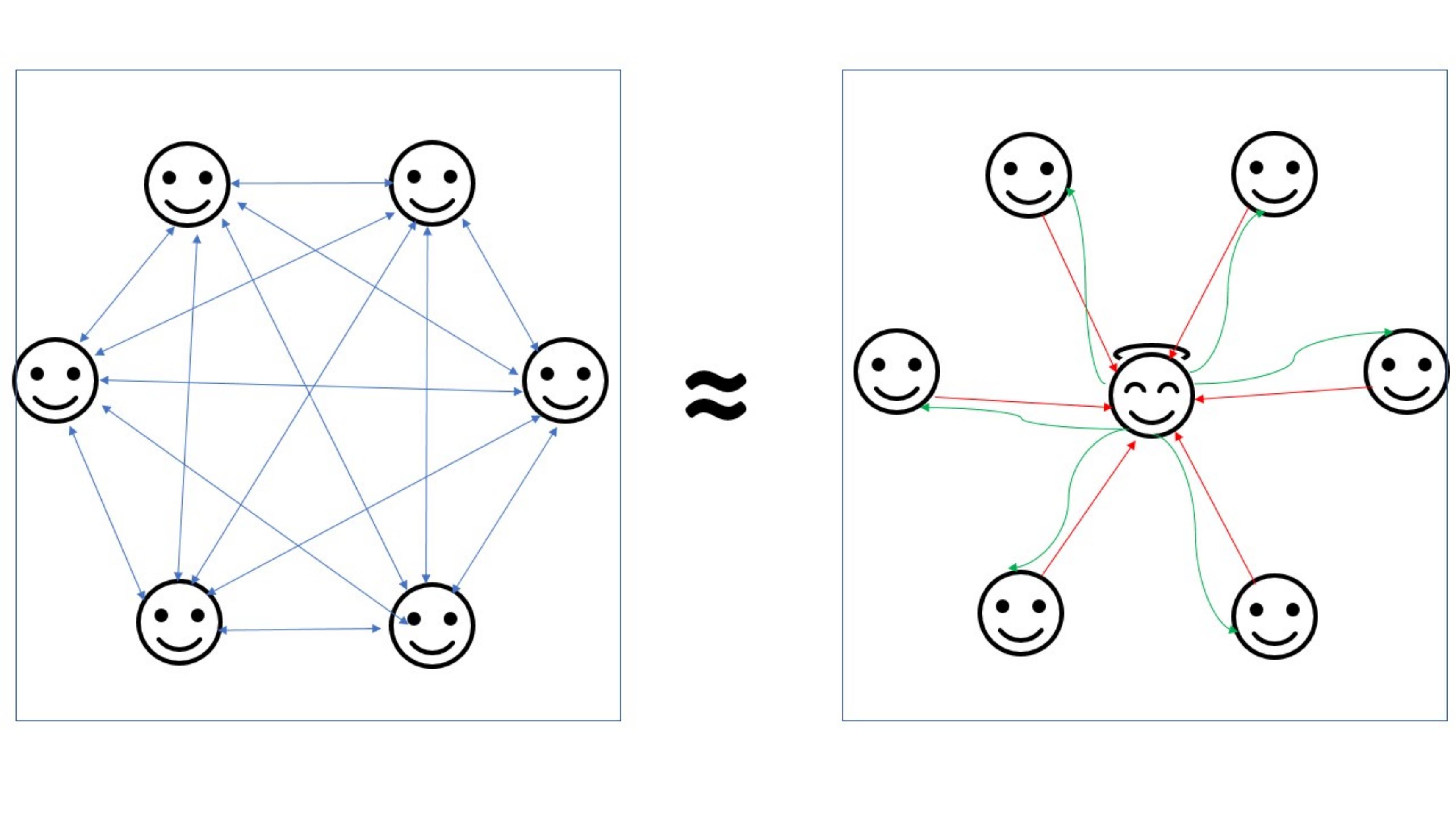
Example #1



T1 = R + x1 + x2 + x3
x3
x4
T2 = R + x1 + x2 + x3 + x4
R + x1 + x2
X4 = S2 - S1
x2
x5
R + x1
R + x1 + x2 + x3 + x4 + x5
x1
x6
R + x1 + x2 + x3 + x4 + x5 + x6

Security Definition

In MPC the goal is to simulate the trusted third party with provable security (computational or information theoretic) with passive or active adversaries



Shamir Secret Sharing Scheme

Assume Party P_0 wants to share secret value s with n parties $P_1, P_2 \cdots P_n$ with a treshold of t

P_0 creates the secret polynomial f of degree t with random coefficients a.

$$f(x) := s + a_1 x + a_2 x^2 \cdots a_{t-1} x^{t-1}$$

Note that $f(0) = s$

Shamir Secret Sharing Scheme

P_0 calculates n different points on f and distributes points to the respective parties.

$$s_i := f(i), 1 \le i \le n$$

ie. P_i gets s_i. They can share their shares and together calculate f then $f(0)$ using Lagrange's Interpolation

$$s + a_1 1 + a_2 1^2 \cdots a_{t-1} 1^{t-1} = s_1$$

$$s + a_1 2 + a_2 2^2 \cdots a_{t-1} 2^{t-1} = s_2$$

$$.$$

$$.$$

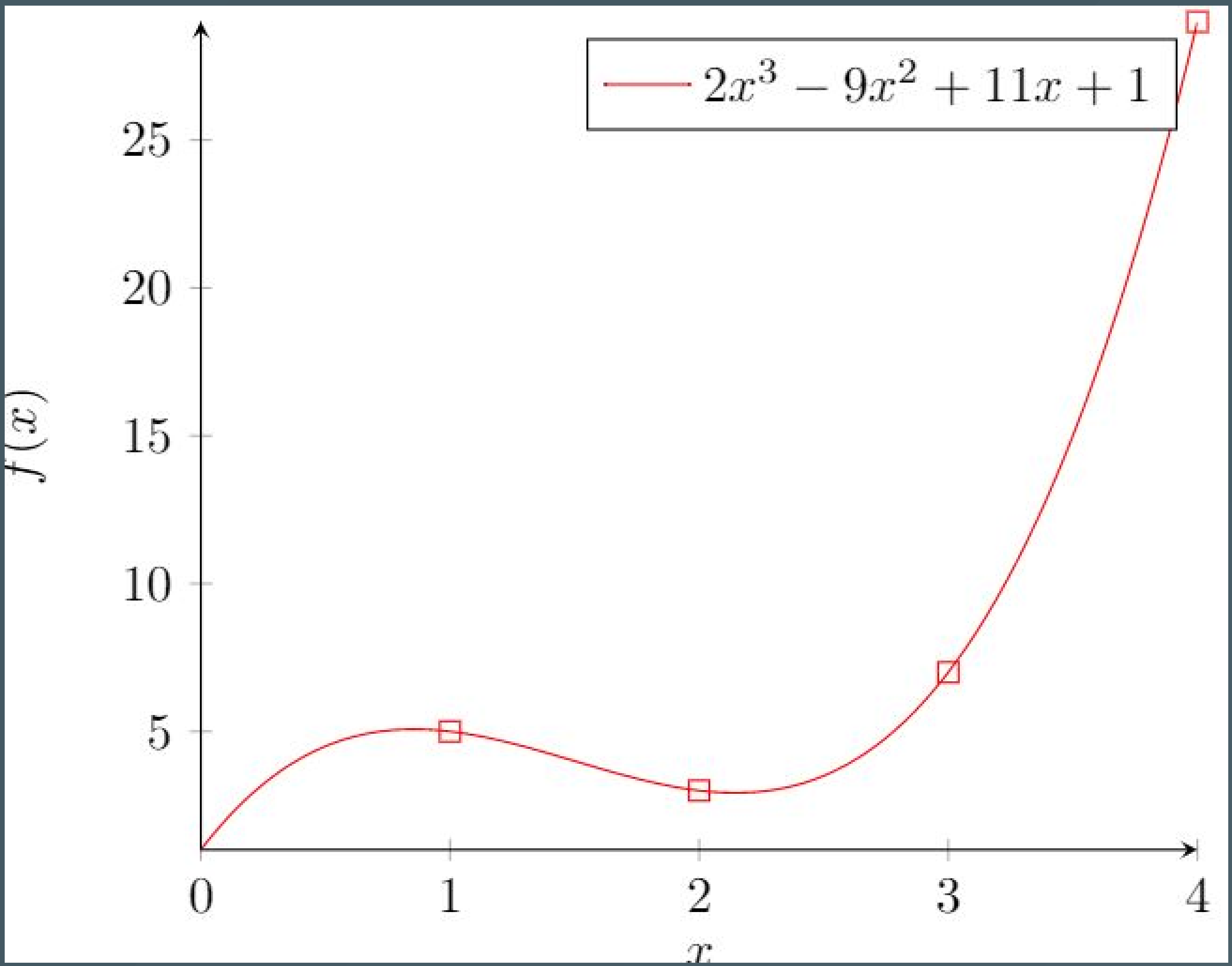
$$s + a_1 n + a_2 n^2 \cdots a_{t-1} n^{t-1} = s_n$$

n equations t unknowns.

Shamir Secret Sharing Scheme

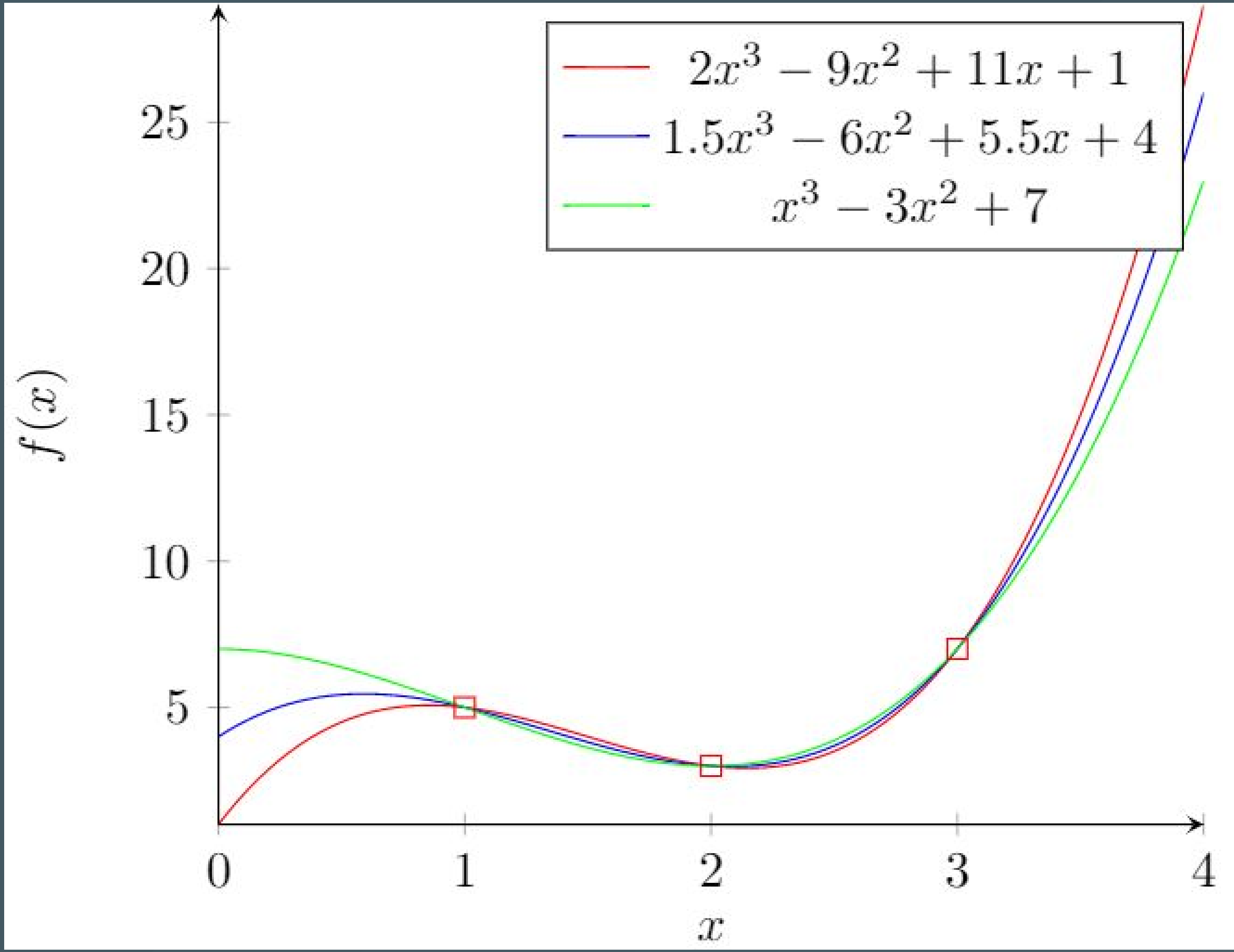
For $t = 4$, $n = 4$ and $s = 1$

$$f(x) := 2x^3 - 9x^2 + 11x + 1; f(1) = 5; f(2) = 3; f(3) = 7; f(4) = 29$$



Security

Assume P_1, P_2 and P_3 came together and tried to recover s without P_4. They can't since there are many polynomials that satisfy s_1, s_2, s_3 but have a different s



Operations on secrets

Let $[x] := [x_1, x_2 \cdots]$ denote the shares of secret value x.
For a publicly known c notice the following:

$$[x + y] = [x] + [y]$$

$$[x + c] = [x] + c$$

$$[c * x] = c * [x]$$

Linear combinations are free in the terms of no communication is required but the initial dealing.

Operations on secrets

However, multiplication of two secret variables is not that easy. $[x * y] = [x] * [y]$ but the new polynomial is of degree $2t - 2$ and not random. Let $h(x) := f(x)g(x)$ with coefficients c notice that:

$$\begin{bmatrix} c_0, c_1 \cdots c_{2t-2} \end{bmatrix} \begin{bmatrix} 1 & 1 \cdots & 1 \\ 1 & 2 \cdots & 2t-2 \\ \cdot & & \\ \cdot & & \\ 1 & 2^{2t-2} \cdots & (2t-2)^{2t-2} \end{bmatrix} =$$

$$\begin{bmatrix} h(1), h(2) \cdots h(2t-2) \end{bmatrix}$$

Operations on secrets

$$\left[c_0, c_1 \cdots c_{2t-2}\right] =$$

$$\left[h(1), h(2) \cdots h(n)\right] * \begin{bmatrix} 1 & 1 \cdots & 1 \\ 1 & 2 \cdots & 2t-2 \\ \cdot & & \\ \cdot & & \\ 1 & 2^{2t-2} \cdots & (2t-2)^{2t-2} \end{bmatrix}^{-1}$$

$c_0 = a_0 b_0$ can be calculated by a linear combination of h values. Now we can compute any Arithmetic Circuit since we have addition and multiplication when $2t < n$. But communication cost is $O(n^2)$

Linear Secret Sharing Schemes

Let s be the secret the secret value. We share s as follows:

$$s = s_0 + s_1 + s_2 \cdots s_n$$

Notice now treshold $t = n$.

Linear Secret Sharing Schemes Operations

For a public c

$$[x + y] = [x] + [y]$$

$$[c * x] = c * [x]$$

For addition with c

$$x_0 \longleftarrow x_0 + c$$

Linear combination of secret variables is also free.

Linear Secret Sharing Schemes Operations (Multiplication)

Assume before the protocol a trusted dealer $\mathcal{T}$ distributed shares of random triplets a, b and c called "Beaver('92) triplets" st.

$$[a] * [b] = [c]$$

One can create such triplets without a trusted dealer $\mathcal{T}$ (SPDZ'12).

Linear Secret Sharing Schemes Operations (Multiplication)

Assume one wants to compute $[z] = [x] * [y]$

$$e := Reveal([x + a])$$

$$d := Reveal([y + b])$$

$$[z] = [c] + e * [y] + d * [x] - ed$$

Note that revealing e and d does not reveal anything about x and y since a and b act as a one time pad. Communication cost is $O(n)$

Oblivious Transfer(OT)

OT is a protocol where

- Sender has messages $x_0, x_1 \cdots x_n$
- Receiver has $s \leq n$
- Receiver learns only x_s
- Sender learns nothing

1/2 OT

This is the case where $n = 2$. Chou, Orlandi('15):

$$\begin{array}{cc}
Sender: & Receiver: \\
a \leftarrow \mathbb{Z}_p & b \leftarrow \mathbb{Z}_p \\
A := g^a \longrightarrow & \\
 & s \leftarrow \{0,1\} \\
 & \longleftarrow B := A^s g^b \\
k_0 := H(B^a) & K := A^b \\
k_1 := H((A^{-1}B)^a) & \\
e_0 := E_{k_0}(x_0) \longrightarrow & \\
e_1 := E_{k_1}(x_1) \longrightarrow & \\
 & x_s = D_K(e_s)
\end{array}$$

Where E is a symmetrix encription function and H is a Hash function.

1/4 OT

We can generate a 1/4 OT with using 2 1/2 OTs.

- Sender has messages x_0, x_1, x_2, x_3 and random r_0, r_1 of same size as x's
- Receiver has 2 bit value s
- Using s's lsb do an OT on r_0, r_1.
- Using s's msb do an OT on $(x_0 \oplus r_0, x_1 \oplus r_1), (x_2 \oplus r_0, x_3 \oplus r_1)$

GMW Protocol

- GMW protocol allows users to compute a boolean circuit as apposed to an aritmetic circuit as before.
- GMW allows two operations XOR and AND which is equalevent to addition and multiplication under $mod\ 2$

  All the operations are in $mod\ 2$ from now on.

  Sharing of the values are done in similar to LSSS

$$x = x_0 + x_1 \cdots x_n$$

GMW Protocol Operations

$$[x + y] = [x] + [y]$$

Since the calculations are under $GF(2)$ operations with public values are trivial. Once again addition is free.

$$[x * y] = (x_0 + x_1 \cdots x_n) * (y_0 + y_1 \cdots y_n)$$

$$= \sum_{1 \leq i \leq n} x_i y_i + \sum_{1 \leq i < j \leq n} x_i y_j + x_j y_i$$

GMW Protocol Operations (Multiplication)

P_i will obtain a share of $x_i y_j + \sum_{i \neq j} x_i y_j + x_j y_i$
To obtain each $x_i y_j + x_j y_i$, P_i interacts with P_j

- P_i has x_i, y_i; P_j has x_j, y_j
- P_j generates a random s
- P_j calculates output of $x_i y_j + x_j y_i + s$ for each possible pair of inputs from P_i
- P_i, P_j perform a 1/4 OT on each possible cases.
- P_i has $x_i y_j + x_j y_i + s$; P_j has s

THANK YOU FOR LISTENING!

QUESTIONS?