Multiparty Computations – 30 Years in the Making

Tal Rabin – IBM Research

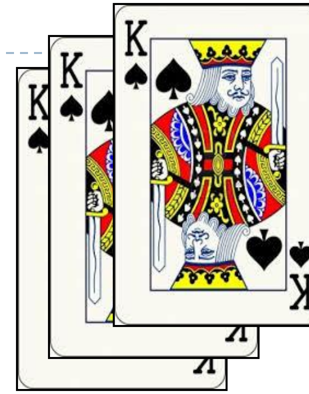
Including slides from: Gilad Asharov, Shai Halevi, Yuval Ishai , Hugo Krawczyk and Yehuda Lindell

Alice and Bob's First Date or Will There be a Second date?

Alice & Bob plan their first date

- After the date
  - Alice knows whether or not she likes Bob
  - Bob knows whether or not he likes Alice
  - But neither know how the other feels
- Then they plan to play a game
  - Game ***only*** reveals if they *both* like each other
    - The logical-AND function
  - But if Alice doesn't like Bob, then she does not learn whether or not Bob likes her (and vice versa)

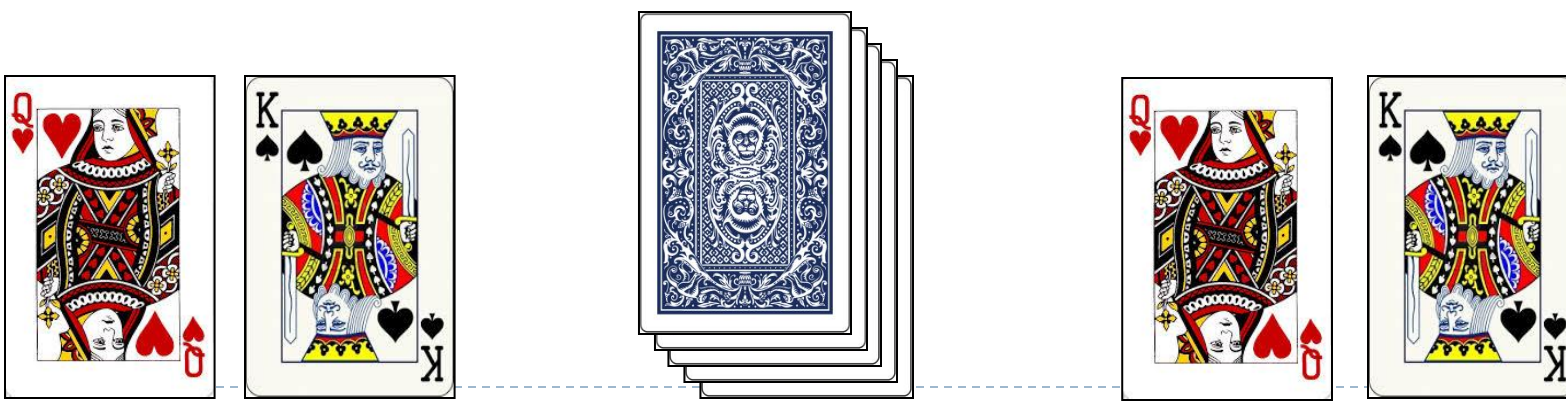
The “Game of Like” [dB’89]



- Alice and Bob use five cards:
  - Two identical queen of hearts
  - Three identical king of spades
- Each of them gets one queen and one king
- Third king is left on the table, face down

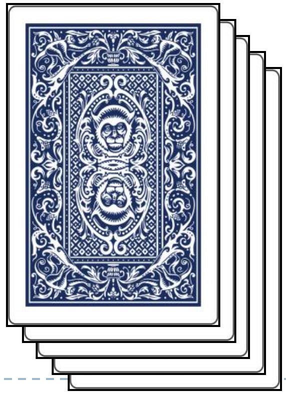
The “Game of Like”

- Bob puts his cards face down on top
  - Queen on top means he likes Alice,
    king on top means he does not
- Alice puts her cards face down on top
  - King on top means she likes Bob,
    queen on top means she does not



The “Game of Like”

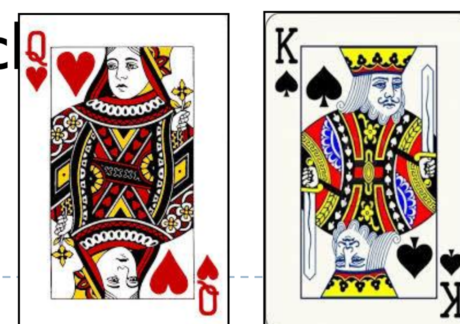
- Alice and Bob take turns cutting the deck
  - Result is a cyclic shift of the deck



The Game of Like

- Alice likes Bob and Bob likes Alice
- View from the bottom
- Bob has to put Queen on top as he likes Alice
- Alice needs to put King on top as she likes Bob

- End result the Queens are next to eac



The Game of Like

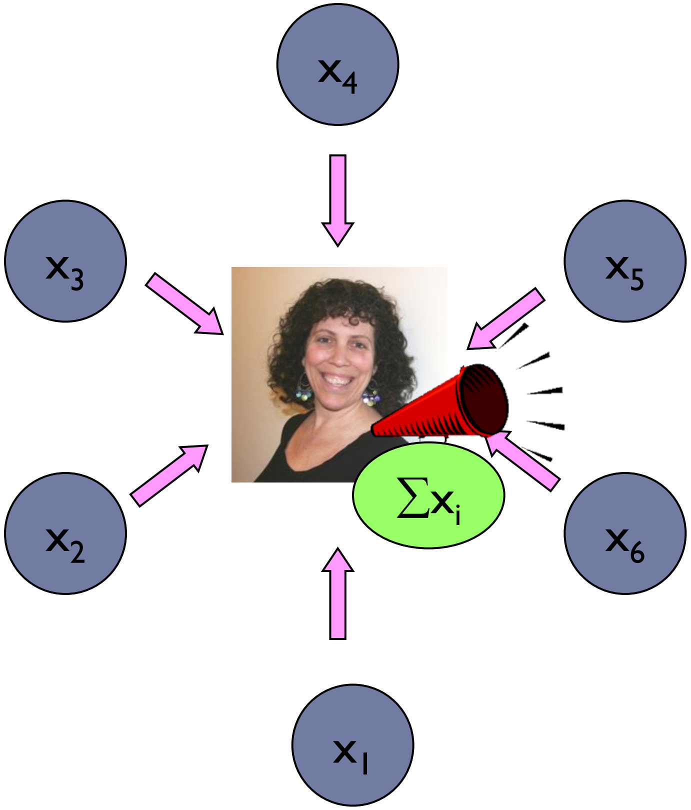
- Bob like Alice but unfortunately Alice doesn’t like Bob
- View from the bottom
- Bob has to put Queen on top as he likes Alice
- Alice needs to put Queen on top as she does not like Bob

- End result the Queens are not next to

The “Game of Like”

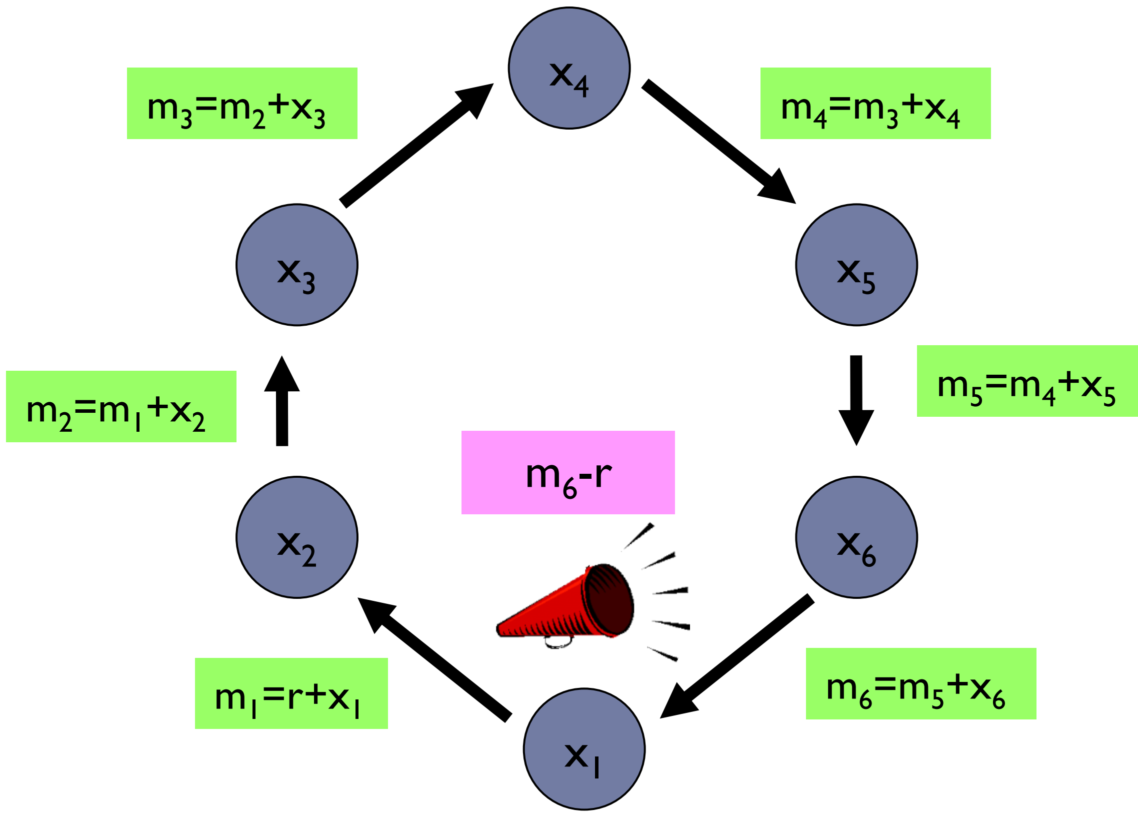
- Alice and Bob take turn cutting the deck
  - Result is a cyclic shift of the deck
- Then they open the cards in order (on a circle)
  - If queens are adjacent they like each other
- Theorem: nothing is revealed when the queens are not adjacent, unknown whether they l like each other or wheth es and one doesn’t

What is the Sum of Our Earnings?



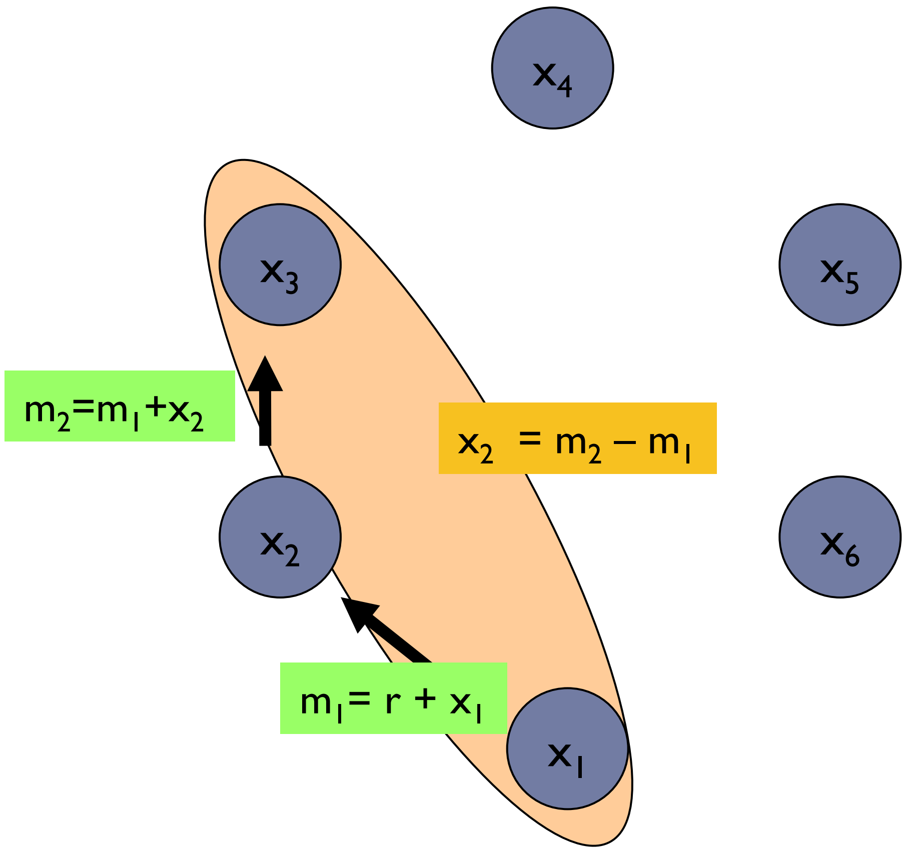
Goal: compute $\sum x_i$ without revealing anything else

What is the Sum of Our Earnings? Or Possibly a Better Way?



Assumption: $\sum x_i < M$ (say, $M=10^{10}$)
(+ and – operations carried modulo M)

What is the Sum of Our Earnings? Problems Seem to Arise



Interesting Problem Emerging [Yao'82,GMW'86]

- Parties $P_1,\ldots,P_n$
- Hold inputs $x_1,\ldots x_n$
- Want to compute $f(x_1,\ldots x_n)$
- Want to preserve their privacy, that their inputs are not exposed
- Want the protocols to work even when some, say t, parties are colluding or corrupt

- Is all this possible?

- The short answer is "yes".
- The long answer is that hundreds and hundreds of papers have been written on the topic.

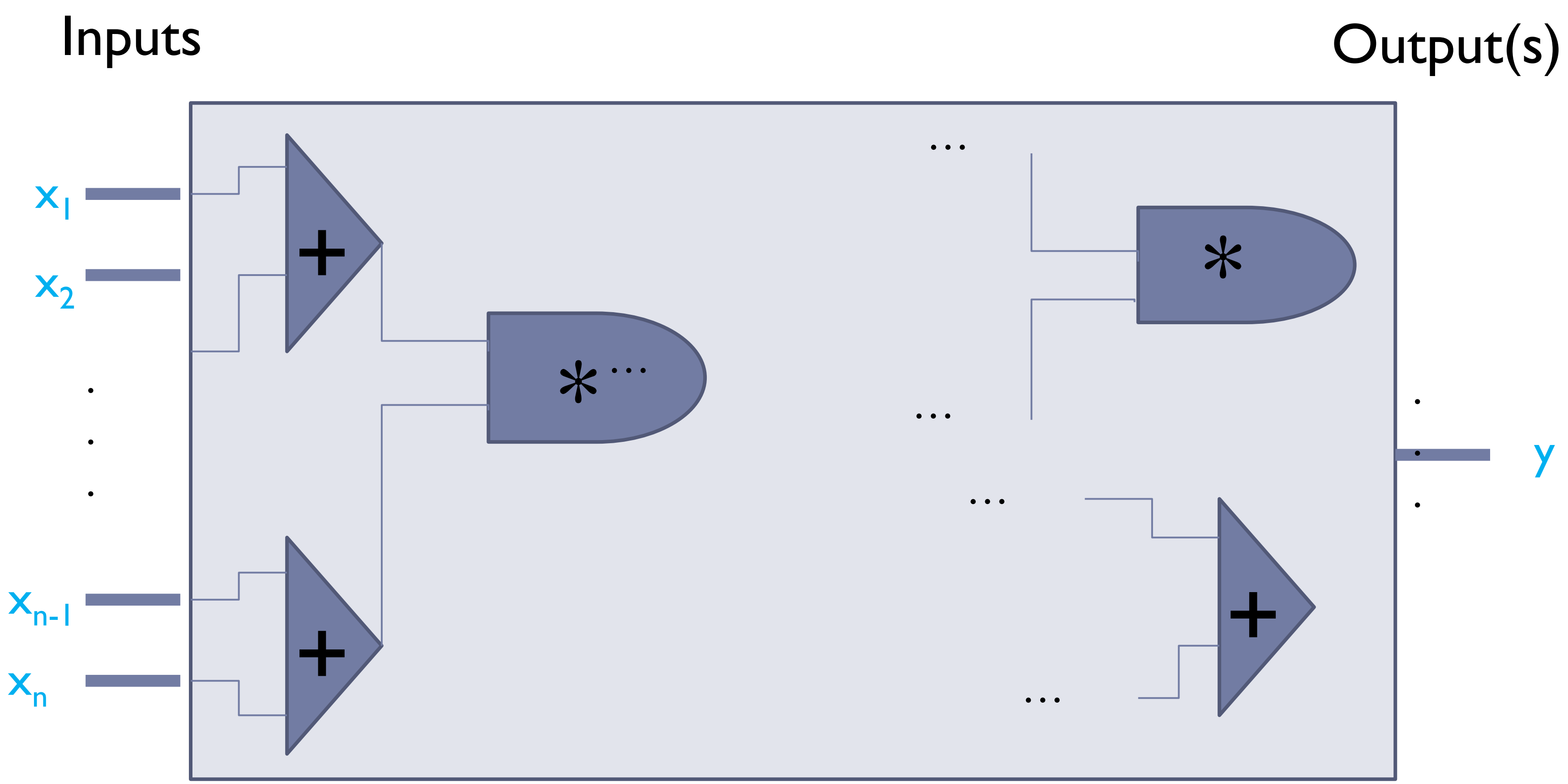
Yes, We Can

- Theorem: For any multiparty function f, there exists a protocol to securely compute f

- Information-theoretic security possible when t<n/2 [BGW88,CCD88,RB89]

- Computational security possible for any t (under standard cryptographic assumptions) [Yao86,GMW87,CLOS02]

The Circuit for Computing a Function



Secret Sharing [Shamir79]

Want to simulate the following concept

Secret Sharing [Shamir79]

- Unique player D holding secret value s (the treasure)

- Two phase protocol:
  - Sharing -- the locking of the treasure in the chest and providing the keys to the parties
  - Reconstruction -- opening of the chest by the parties

  - $f(x) = a_t x^t + \ldots + a_1 x + a_0 (=s) \bmod p, \; f(0) = s$
  - Party P_i receives $s_i = f(i)$

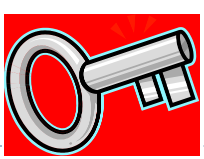
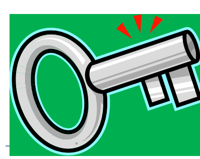
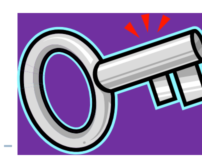
 $= f(1)$  $= f(2)$  $= f(3)$

  - Any t+1 “keys” can be used to reconstruct s

Secret Sharing [Shamir79]

- Remember our problem with the colluders who exposed the value of one of the parties
- Can happen here as well, so we make an assumption on the number of colluding parties, say t
- And then we set the number of "shares" needed to t+1 by fixing the degree of f(x) to t
- And the polynomial needs to be random

- We may think that two might try to steal the treasure so we put three locks on the chest

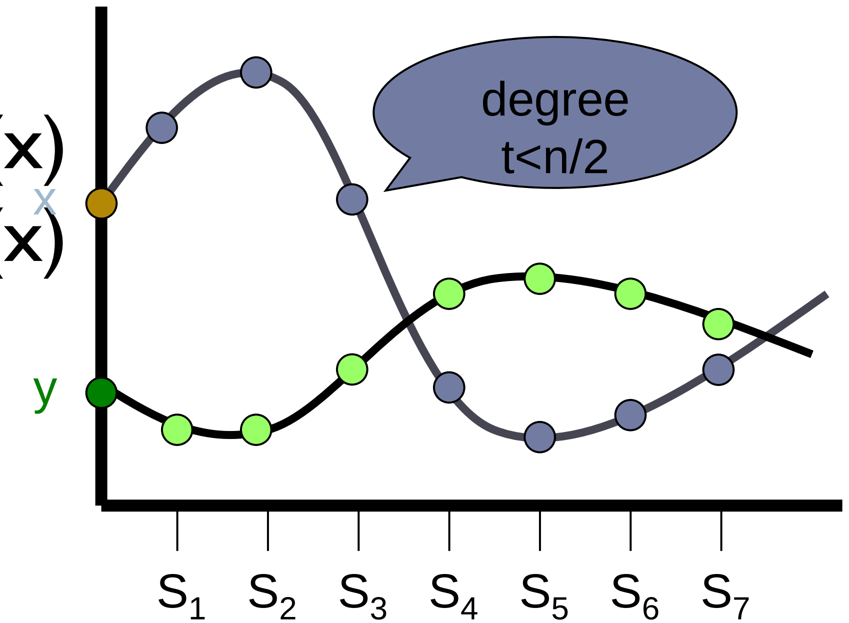
 = f(1)  = f(2)  = f(3)

Computing x_1+x_2 [BGW88]

Player P_i creates $f_i(x)$ and gives P_j the share $f_i(j)$

$f_i(0) = x_i$

P_1	P_2	P_3	…	P_7	
s_{11}	s_{12}	s_{13}		s_{17}	$\rightarrow f_1(x)$
s_{21}	s_{22}	s_{23}		s_{27}	$\rightarrow f_2(x)$



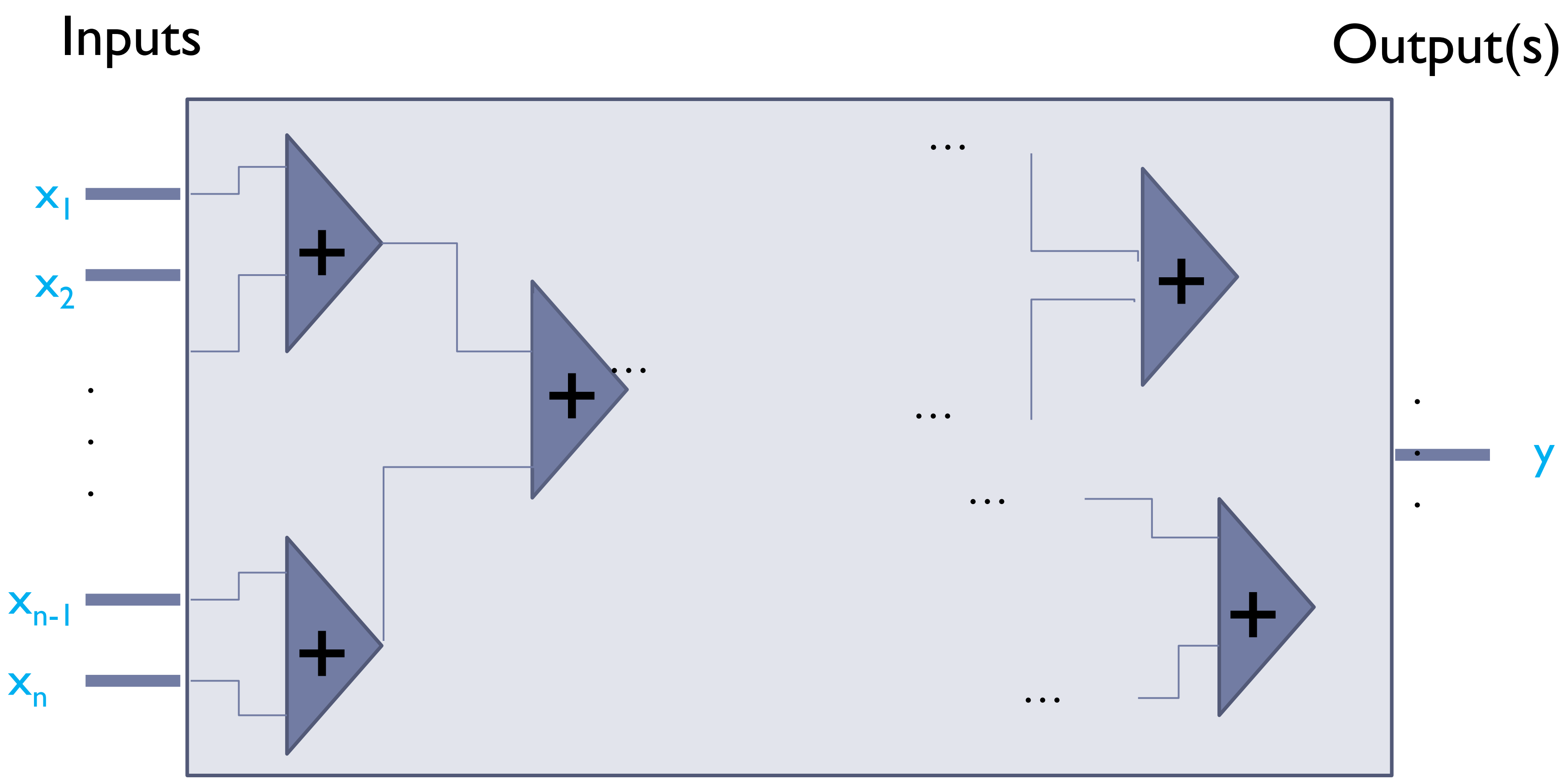
$\sum s_{j1} \quad \sum s_{j2} \quad \sum s_{j3} \ldots \sum s_{j7} \qquad \sum f_i(x) = f(x)$

Computing x_1+x_2 [BGW88]

$\sum s_{j1}$ $\sum s_{j2}$ $\sum s_{j3}$ … $\sum s_{j7}$ $\sum f_i(x) = f(x)$

- $\sum f_i(x) = f(x)$
- $f(0) = x_1 + x_2$
- $f(i) = \sum s_{ji}$
- So now each party has a share of the sum of all the values
- We are still guaranteed that t+1 shares are needed to reconstruct s

We Can Do Unlimited Number of Additions



There is no communication required for computing the additions

BGW Protocol – Multiplication by a Constant

Player P_i creates $f_i(x)$ and gives P_j the share $f_i(j)$, $f_i(0) = x_i$

To compute $c*f_i(x)$ parties compute locally $c*f_i(j)$

Multiplication by a constant does not require communication

BGW Protocol -- Multiplication

Each party P_i shares $f_i(x)$, s.t. $f_i(0)=x_i$

- We saw how to do addition; all parties locally add their shares of both the values, creating a polynomial f(x), s.t. $f(0) = \sum x_i$
- We need multiplication; we have

  $f_i(x)$, s.t. $f_i(0)=x_i$ and $f_j(x)$ s.t. $f_j(0)=x_j$

  $f_i(x) * f_j(x) = f(x)$ observe that $f(0)= x_i * x_j$

However, more work needs to be done!

BGW Protocol -- Multiplication

$f_i(x)$, s.t. $f_i(0)=x_i$ and $f_j(x)$ s.t. $f_j(0)=x_j$

$f_i(x) * f_j(x) = f(x)$ observe that $f(0)= x_i * x_j$

The problems that need to be addressed are

1. f(x) is of degree 2t
2. f(x) is not random

To deal with these problems the multiplication protocol requires communication

One Step Degree Reduction and Randomization [GRR]

- $f(x)$ of degree $2t$
- Each party holds $f(i)$
- Let $f(x) = a_0 + a_1 x + \ldots + a_{2t} x^{2t}$ $a_0 = x_1 * x_2$

$$[a_0, a_1, \ldots, a_{2t}] \begin{bmatrix} 1, 1, \ldots, & 1 \\ 1, 2, \ldots & 2t+1 \\ 1, 2^2, \ldots, & (2t+1)^2 \\ \ldots & \\ 1, 2^{2t}, \ldots, & (2t+1)^{2t} \end{bmatrix} = [f(1), f(2), \ldots, f(2t+1)]$$

One Step Degree Reduction and Randomization (cont)

$$[f(1), f(2), \ldots, f(2t+1)]\ A^{-1} = [a_0 = x_1 x_2, a_1, \ldots, a_{2t}]$$

Thus,

$$x_1 x_2 = \lambda_1 f(1) + \lambda_2 f(2) + \ldots + \lambda_{2t+1} f(2t+1)$$

And now we are ready.

One Step Degree Reduction and Randomization (cont)

- Each party P_i shares its share f(i) using a random t degree polynomial $h_i(x)$ such that $h_i(0)=f(i)=f_1(i)f_2(i)$
- Examine the polynomial h(x)

  $h(x)=\lambda_1 h_1(x) +\lambda_2 h_2(x)+\ldots+ \lambda_{2t+1} h_{2t+1}(x)$

  this is a random, t-degree polynomial such that

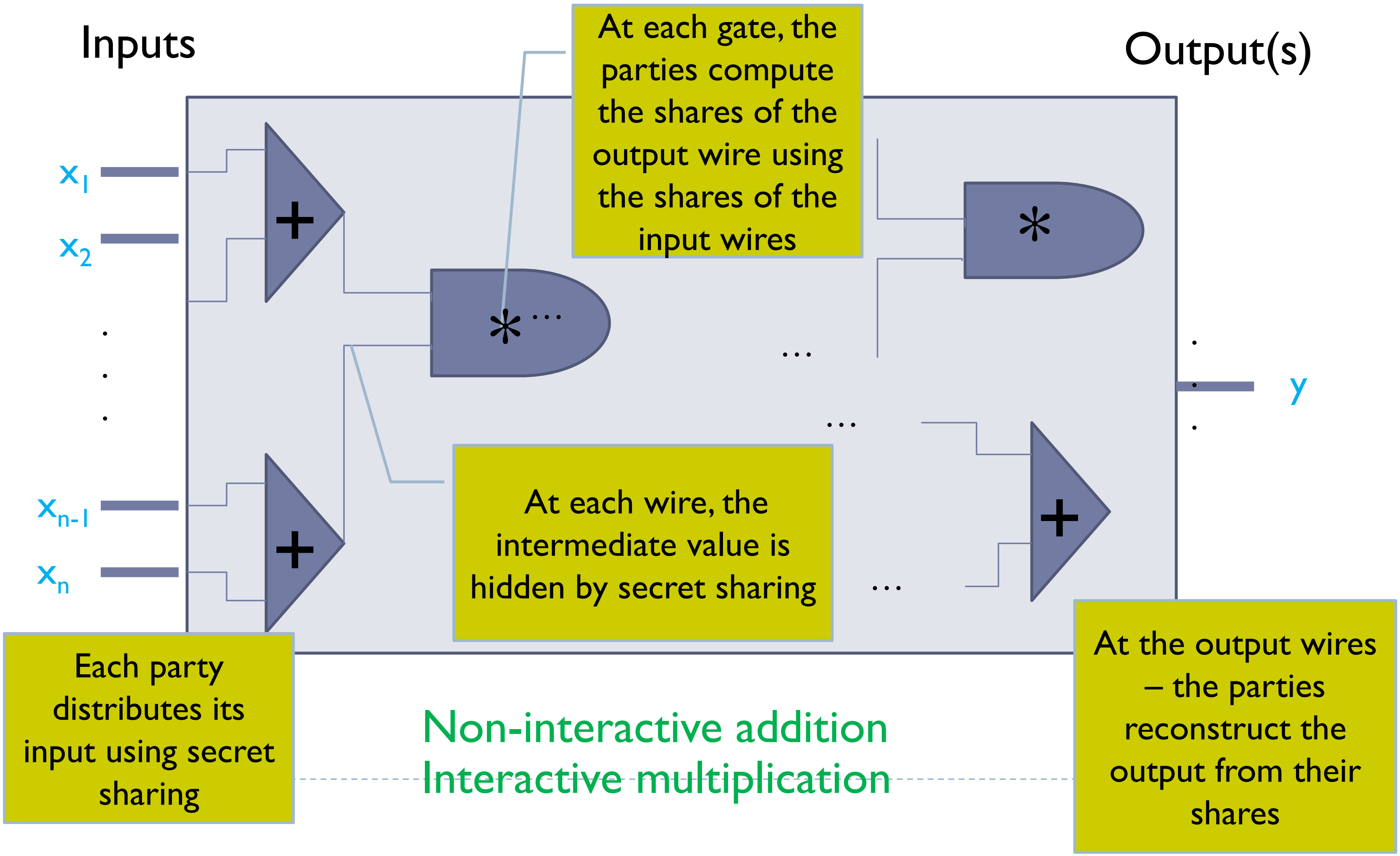
  $h(0) = x_1 x_2$

  and each party holds the value h(i)

Adding a Malicious Advesary

- The secret sharing, addition and multiplication are for honest-but-curious behavior of the adversary
- Methods to ensure correctness:
  - Verification in the secret sharing
  - Use of error correction in the reconstruction
  - And proofs of correct actions via computation in the multiplication step

The BGW Protocol



Non-interactive addition
Interactive multiplication

Open Question (for 25 years)

- Information-theoretic security possible when $t<n/2$ [BGW88,CCD88,RB89]. The round complexity is depended on the depth of the circuit.

- Computational security possible for any t (under standard cryptographic assumptions) in a constant number of rounds [Yao86,BMR]

- Open Question
  Can we compute any function in a constant number of rounds with Information theoretic security?

General Multiparty Computations

- Functionality f mapping n inputs to n outputs
  - possibly randomized
- Goal: t-secure protocol realizing f
  - Emulate an ideal evaluation of f using a trusted party
    … even if up to t of the n parties can be corrupted
- Variants:
  - Semi-honest vs. malicious corruptions
  - Honest majority ($t<n/2$) vs. no honest majority ($t\geq n/2$)
  - Information-theoretic vs. computational security
  - Standalone vs. composable security
  - Different network models, setup assumptions
  - FAIRNESS

Applications of Multiparty Computations

- **Electronic voting:**

  Correctness, acountability, privacy, coersion-freeness...
- **"E-commerce":** *Fairness, accountability*
  - On-line Auctions, trading and financial markets, shopping
- **On-line gambling:** *Unpredictability, accountability...*
- **Computations on databases:** *Privacy*
  - Private information retrieval
  - Database pooling
  - No-fly list FBI has list of suspect, airline has list of passengers, output is the intersection of the two lists
- **Secure distributed storage:** *Availability, integrity, secrecy*
  - Centrally controlled
  - Open, peer-to-peer systems

Example: Auctions

- Consider a secure auction (with secret bids):
  - An adversary may wish to learn the bids of all parties – to prevent this, require PRIVACY
  - An adversary may wish to win with a lower bid than the highest – to prevent this, require CORRECTNESS
  - But, the adversary may also wish to ensure that it always gives the highest bid – to prevent this, require INDEPENDENCE OF INPUTS

Real-World Secure Computation

- Prices of Sugar Beets in Denmark are determined by secure computation

  - Jan 2008: "MPC gone live" in Denmark

- Some universities and other organizations are using cryptographic voting protocols

Improving Efficiency

- Extensive research over last decade into improving efficiency and usability
- Yao semi-honest
  - In 2004 Fairplay ran Yao for billionaire's problem. Median on ten 16-bit numbers (circuit of size 4383 gates) took between 7.09 and 16.63 seconds
  - In 2011 using state-of-the-art algorithmic improvements, and systems optimizations AES computation (with 9,280 non-XOR gates) took just 0.2 seconds overall (after an additional 0.6 seconds of preprocessing that can be used for many executions)

Improving Efficiency

- Extensive research over last decade into improving efficiency and usability
- MPC against malicious adversaries
  - In 2004 – Nothing
  - In 2012 an implementation of secure AES computation took < 30 seconds on 4-cores, and about 8 seconds on 16-cores
- SCAPI, Towards Billion-Gate Secure Computation with Malicious Adversaries, Some start-ups, code libraries, etc.

- Much room for efficiency improvements

What if the Function is Exponentiation?

- In what setting may we want to compute an exponentiation?
- Digital signatures: RSA or DSA

RSA: Public verification key (N, e)

Secret signing key: d

Simplification of signing algorithm on message m:

$$m^d \bmod N = s$$

When secret signing key stored in a single location we can compute signature easily

May not Want to Store the Whole Signing Key in Single Location

- A few reasons:
  - Important key, can sign million dollar transactions
    - Would be target for attacks
  - Vulnerability of signing key
    - Malicious attacks (hackers) -- Issue of secrecy
    - Hardware or virus problems -- Issue of availability
  - Want at least two, say of three, position holders to make the decision to sign

Could Use Multiple Keys

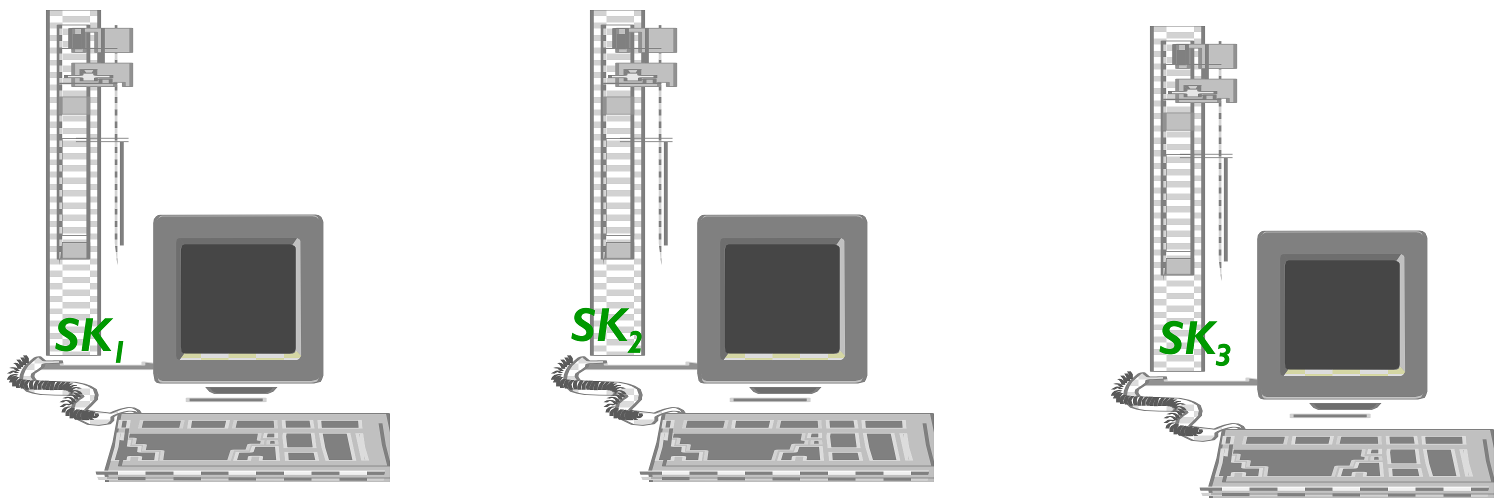
- Could say that now a legal signature constitutes having two valid signature under two different keys

Two different signatures

- However, this is an efficiency issue for the verifier of the signature

Seems that Using Shamir Would be Good

- Split the key! Provide secrecy
- The key is *sk* and it is split into (eg) three shares: sk_1 sk_2 *and* sk_3 two of which are needed in order to sign



Secret Sharing Helps

- It protects our secret data, the signing key
- Recall our goal: to generate signatures
  - Problem: the key, *sk*, is split
- (Faulty) solution: combine the key from its shares into a single location and sign -- single point of failure
- A solution: we have our theorems stating that any function can be computes securely
- Can we do better than general purpose multiparty computations?

Threshold Cryptography – Signature Generation [DF,GJKR]

- Secret key sk
- Message m
- Signature m^{sk}
- Reminder: $sk = sk_1 + sk_2 + sk_3$
- Processor P_i publishes m^{ski}
- $m^{sk1}\, m^{sk2}\, m^{sk3} = m^{sk1+sk2+sk3} = m^{sk}$
- Scheme can be modified to fit the threshold representation!

Proactive Security

- Is our security assumption realistic?
  - That over the life time of the system only t processors will be compromised?
- No! For some applications
- Want to provide security for a long period of time
- Split the lifetime into periods $C_1,\ldots,C_T$
  - Time periods can be minutes, days, weeks
- In each time period at most t processors can fail
- During the lifetime of the system every processor can be broken into

Proactive Security

- Start with the initial key *SK*

Shares held by parties | Share captured by attacker

SK =	SK_1	SK_2	SK_3	time period T_1	SK_2
SK =	SK_1	SK_2	SK_3	time period T_2	SK_1
SK =	SK_1	SK_2	SK_3	time period T_3	SK_3
SK =	SK_1	SK_2	SK_3	time period T_4	SK_3
SK =	SK_1	SK_2	SK_3	time period T_5	SK_1

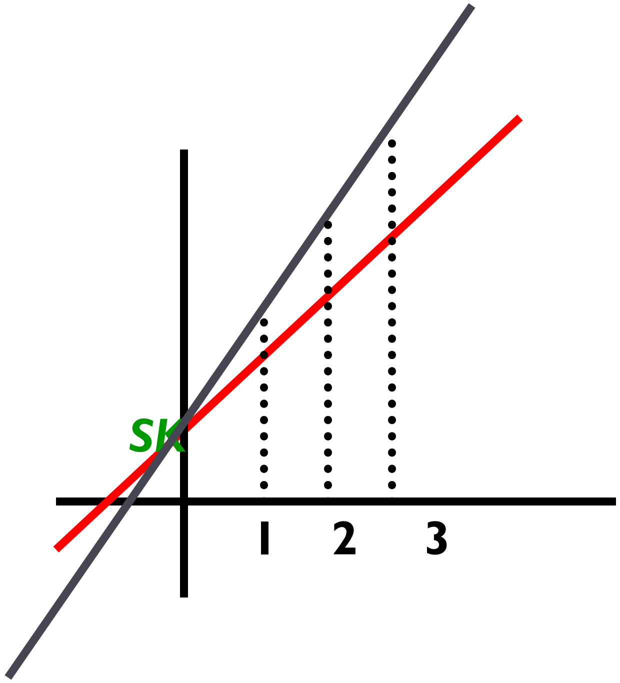
USELESS INFORMATION!

SK

Recall that the actual key is *SK*

Changing the Representation

- First representation: $f_1(x) = a_1x + sk$
  - P_i holds $f_1(i)$
- Second representation: $f_2(x) = a_2x + sk$
  - P_i holds $f_2(i)$



Now, two shares, each from a different representation do not expose *sk*

Changing Representation (cont.)

Time period 1:	$f_1(1)$	$f_1(2)$	$f_1(3)$
Time period 2:	$f_2(1)$	$f_2(2)$	$f_2(3)$

Attacker breaks into P_3 in time period 1 and into P_2 in time period 2

Thus, the attacker knows $f_1(3)$ and $f_2(2)$

USELESS INFORMATION!

How to Change the Representation

- Assume that the processors have shares on $f_1(x) = a_1 x + sk$, i.e. the value $f_1(i)$ for $i=1,2,3$
- Add a random polynomial $r(x)$ such that $r(0)=0$, $r(i)$ for $i=1,2,3$
- This gives the polynomial $f_2(x) = f_1(x) + r(x)$ which is the new representation
- Processor P_i computes $f_1(i) + r(i)$

The Conflict

Data in the clear is not secure or private

Solution: Encrypt the data

Encrypted data prevents search/query

Solution: Decrypt data or do not encrypt data

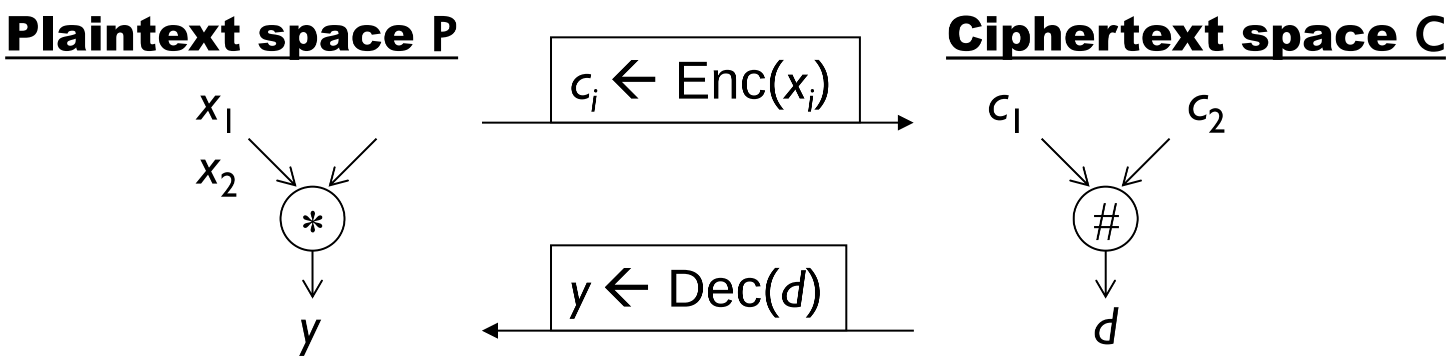
OR provide advanced technologies

Encrypted Database or Storage

- Databases without search capabilities are mostly useless
- Security and privacy call for encrypting databases
- When wanted to search:
  - Need to decrypt to be able to search even if all we are looking for is a single row (i.e. single record)
- It's even more serious when database is outsourced (e.g. cloud)
  - If encrypted by owner of data then cloud server can't decrypt hence can't search
  - Who has the keys to search/decrypt?
  - Today: The cloud server has it, thus (all) plaintext data visible to server
  - Ideally: Owner of data encrypts and keeps the keys but then how it searches the remote encrypted data?
    - Giving keys to server takes us back to above case

“Privacy Homomorphisms”

Rivest-Adelman-Dertouzos 1978



Fully Homomorphic Encryption

Big question open for 30 years:

- Can we have a *single* system which is *both* additively and multiplicatively homomorphic, i.e. FULLY HOMOMRPHIC ENCRYPTION

Fully Homomorphic Encryption

Can Alice securely delegate the processing of her data without giving away access to it?

Problem introduced 30 years ago by Rivest, Adleman and Dertouzos

Alice

Cloud server provider

Please send me the tax report for 2007

Fully Homomorphic Encryption

Encryption:

5

Alice 

Bob

Fully Homomorphic Encryption

Encryption:

Decryption:

Alice

Bob

Fully Homomorphic Encryption

Additively Homomorphic Encryption:

Alice  Encryption

Decryption  **Bob**

5  **9** 

Charlie

Fully Homomorphic Encryption

Additively Homomorphic Encryption:

Alice 

Encryption

Decryption

 **Bob**

Eval **Charlie**

Homomorphic Encryption

Multiplicatively Homomorphic Encryption:

Alice  Encryption

Decryption  **Bob**

Eval **Charlie**

Example of Additive Homomorphism

- Goldwasser-Micali Encryption [GM'82]
  - Encrypt 0 by a square mod *N*
  - Encrypt 1 by a non-square mod *N*
- If $ctxt_1$ encrypts b_1 and $ctxt_2$ encrypts b_2 then $ctxt_1 \cdot ctxt_2 (mod\ N)$ encrypts the bit $b_1 + b_2\ (mod\ 2)$
  - You can add encrypted bits

What Did We Have?

- Encryptions that were additively homomorphic
- Encryptions that were multiplicatively homomorphic
- A system with many additions and one multiplication [BGN]

Fully Homomorphic Encryption

Break-through result of Craig Gentry [2009]:

Can achieve fully homomorphic encryption!

- Compute on encrypted data

  Encryptions of $m_1, \ldots m_t$ under PK, $E(m_1),..E(m_t)$
  $\rightarrow$ Encryption of $f(m_1,\ldots,m_t)$ for any function f,
  $E(f(m_1,\ldots,m_t))$

Can in theory compute any function under the encryption.

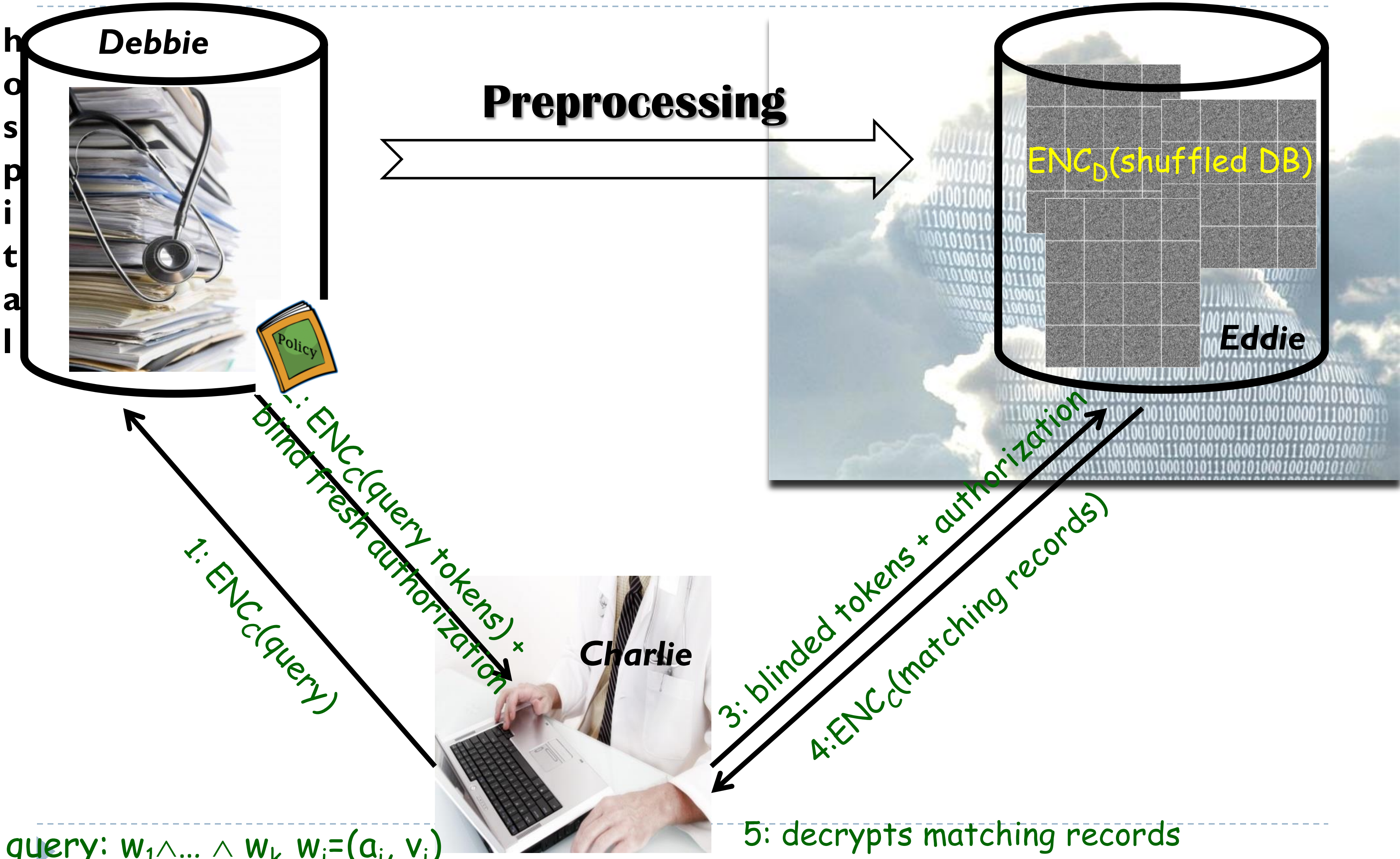
Combining FHE and Threshold Crypto for Multiparty Computations

- Create a distributed sharing of a key for an FHE scheme
- Each party holds a share of the decryption key
- The parties publish the public key PK of the scheme
- Each party encrypts its input $ENC_{PK}(x_i) = e_i$
- Each party computes the function on $e_1,\ldots,e_n$ using the addition and multiplication operations
- The parties together decrypt the final output

Status of Real-World HE

- Still Experimental
- Open-source `HElib` implementation on github
- Performance improved by ~6 orders of magnitude since 2009, but still very costly
- May be suitable for niche applications

Example: Medical Application



What has been achieved: Highlights

- Algorithmic support for full-text and *general* Boolean queries
  - "lastname=Mills" and "name=Steve or Stephen" and "not(company=IBM)"
  - Upcoming: range queries, substring/wildcards query
- Validated on synthetic census data: 10Terabytes, 100 million records, > 100,000,000,000 indexed record-keyword pairs !
  - Equivalent to a DB with one record for each American and 400 keywords in each record (including textual fields)
- Pre-processing and query time scales linearly with DB size
- Support for updates: add/delete/modify documents