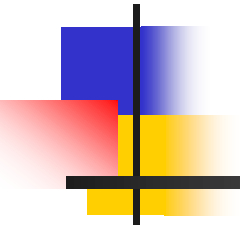
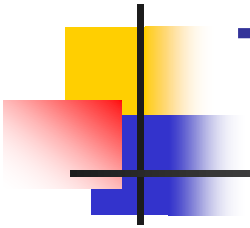


Secure function evaluation
: few examples

Somitra Sanadhya, IIIT Delhi

somitra@iiitd.ac.in

(some slides are taken from a talk by Yehuda Lindell)

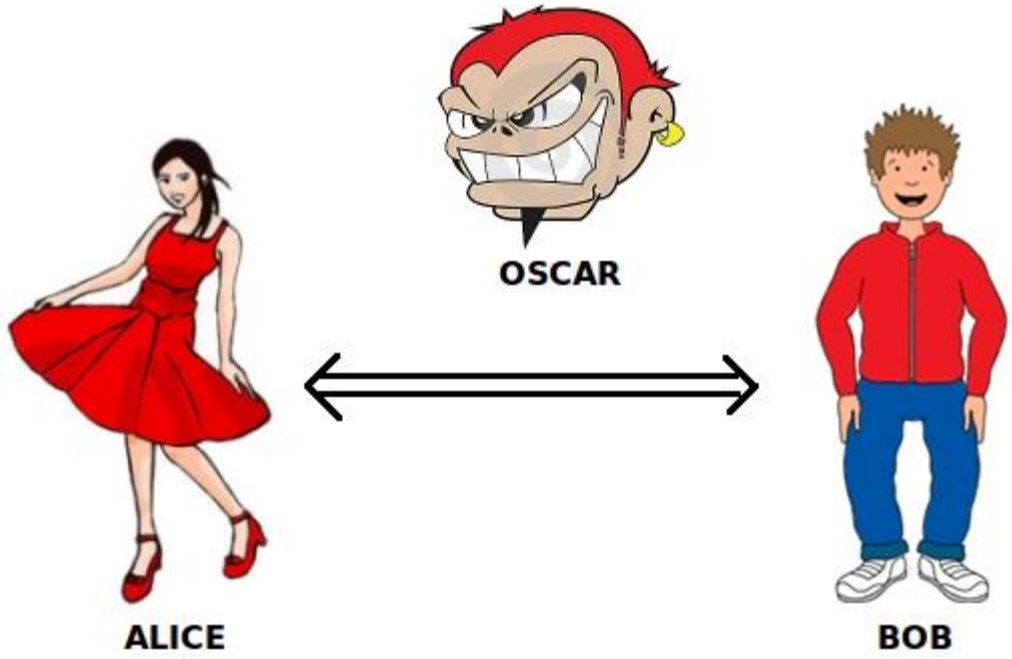


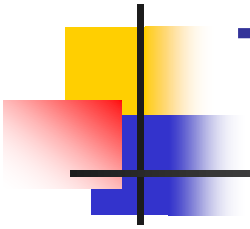
The setting …

- Hospitals generate large amount of patient data
- Combining data could be of help in developing better diagnosis and treatment tools
- HIPAA rules protect patient privacy
- Would be easy if everyone could work on encrypted data and compute some desired function of the data.

Basic setting of Cryptography

- Secure communication over insecure channel





The problem of security

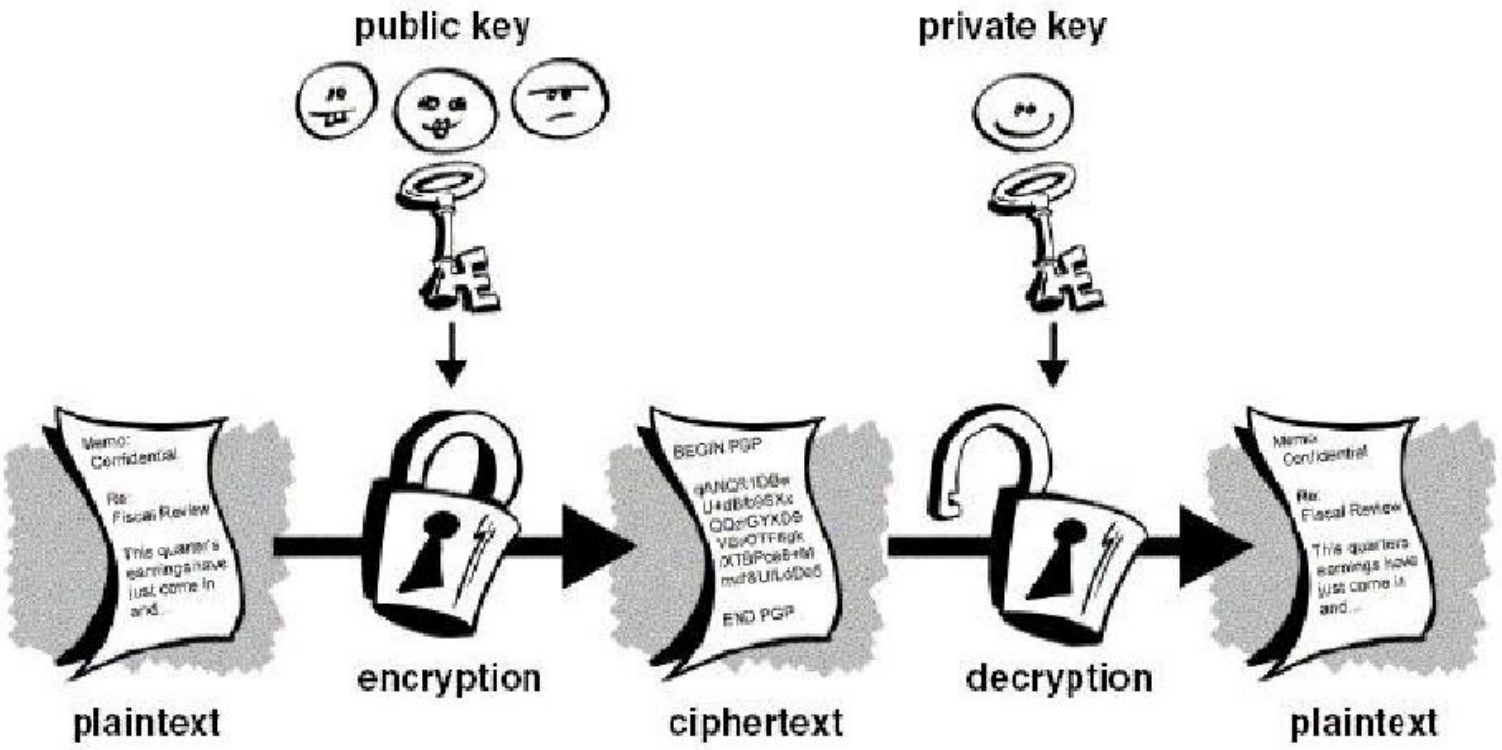
- How can Alice and Bob communicate securely? (over an open channel)
- If Oscar knows everything that A and B know, then this is impossible
- A and B must have some secret. Two options:
  - (a) share some secret data (key)
  - (b) use a secret algorithm (cipher)

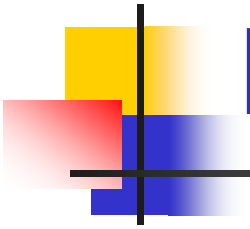
Kerckhoff's Law

- La Cryptographie Militaire [1883]
- A cryptosystem should remain secure even if everything about the system, except the key, is public.
- Shannon [1949]: "The enemy knows the system"

Figure: Only the key is secret

Assymetric cryptosystem



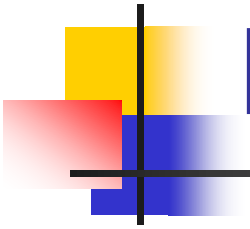


PKC

- A : public key e, private key d
- B : can access e, but not d

- Further, note that
  - $Dec_d(Enc_e(x)) = x$
  - $Dec_d(Enc_e(x)+y)$ = random looking thing
    (for someone who knows x,y,e but not d)

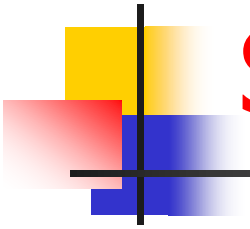
Multiparty Computation



- A set of parties with private inputs wish to compute some joint function of their inputs.
- For example, A: x_1, B: x_2, C: x_3.
- Supply to a function $f(x_1, x_2, x_3)$ which outputs a 3-tuple:

$(g_1(x_1, x_2, x_3), g_2(x_1, x_2, x_3), g_3(x_1, x_2, x_3))$

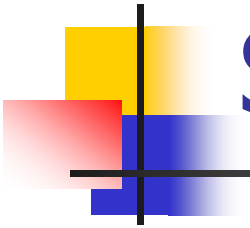
- Sometimes, $g_1 = g_2 = g_3$. But this is not necessary.



Secure Multiparty Computation

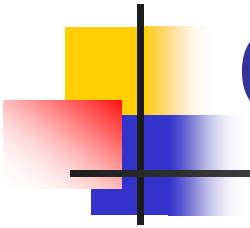
- A set of parties with private inputs wish to compute some joint function of their inputs.
- Parties wish to preserve some security properties. E.g., privacy and correctness.
- Security must be preserved in the face of adversarial behavior by some of the participants, or by an external party.

Some Use-cases



- Why study this ?

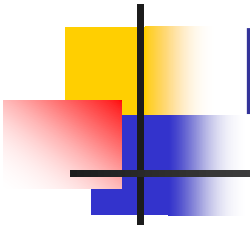
- Secure election
- Auction bidding
- Private Information Retrieval
- …



Case studies

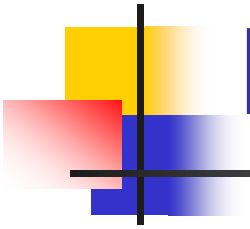
- Yao's millionaire protocol
- Oblivious transfer
- Coin tossing over Telephone

  - We see them one by one, showing the problem and the solution, followed by a brief idea of the proof.
  - Before this, we first discuss more on "security".



Protocols and Functions

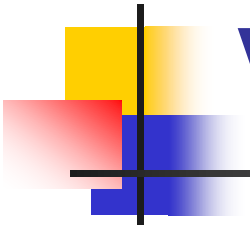
- Cryptography aims for the following (regarding privacy):
  - A secure protocol must reveal no more information than the output of the function itself
  - That is, the process of protocol computation reveals nothing.
- Cryptography does not deal with the question of whether or not the function reveals much information
  - E.g., mean of two parties' salaries
- Deciding which functions to compute is a different challenge that must also be addressed in the context of privacy preserving data mining.



Defining Security

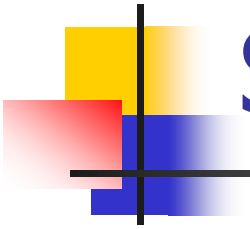
- Components of ANY security definition
  - Adversarial power
  - Network model
    - Type of network
    - Existence of trusted help
    - Stand-alone versus composition
  - Security guarantees

- It is crucial that all the above are **explicitly and clearly defined**.



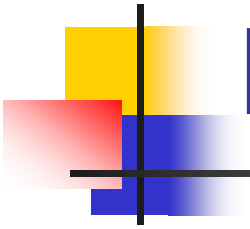
Vague Definitions

- If the network and adversarial model are not fully defined, then:
  - Secure protocols can be "broken" because they were proven in an unreasonable setting
  - If the adversary is unknown, then we can only reason about security very informally (this is a very common mistake)
  - It is not clear what is and what is not protected against. (It may be impossible to protect against everything – but we must be up-front about it.)



Security Requirements …example

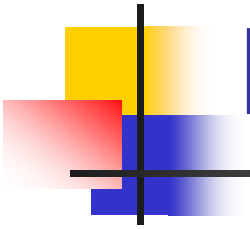
- Consider a secure auction (with secret bids):
  - An adversary may wish to learn the bids of all parties – to prevent this, require PRIVACY
  - An adversary may wish to win with a lower bid than the highest – to prevent this, require CORRECTNESS
  - But, the adversary may also wish to ensure that it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



Defining Security

- Option 1:
  - Analyze security concerns for each specific problem
    - Auctions: as in previous slide
    - Elections: privacy and correctness only (?)
- Problems:
  - How do we know that all concerns are covered?
  - Definitions are application dependent (need to redefine each time).

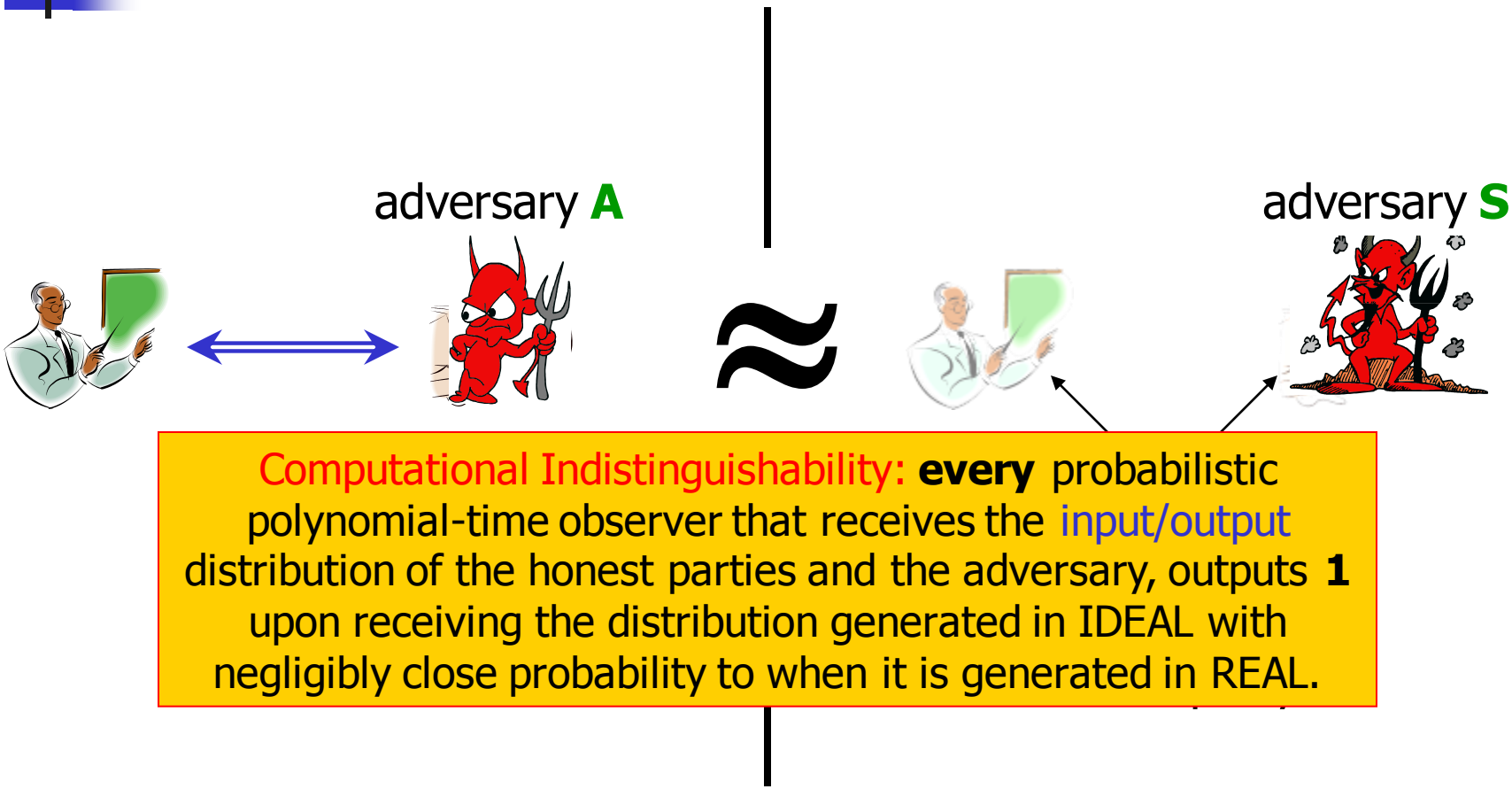
Defining Security – Option 2

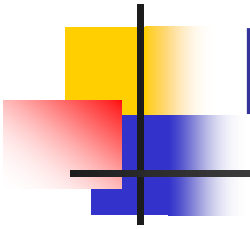


- The real/ideal model paradigm:
  - Ideal model: parties send inputs to a trusted party, who computes the function and sends the outputs.
  - Real model: parties run a real protocol with no trusted help.

- Informally: a protocol is secure if any attack on a real protocol can be carried out (or simulated) in the ideal model.
- Since essentially **no** attacks can be carried out in the ideal model, security is implied.

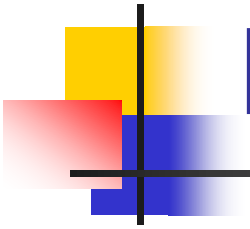
The Security Definition:





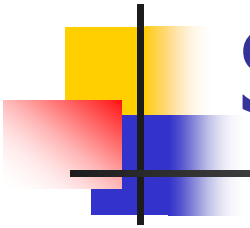
Meaning of the Definition

- Interpretation 1:
  - Security in the ideal model is absolute. Since no attacks are possible in the ideal model, we obtain that the same is also true of the real model.
- Interpretation 2:
  - Anything that an adversary could have learned/done in the real model, it could have also learned/done in the ideal model.
  - Note: real and ideal adversaries have same complexity.



More Definitions

- There are numerous ways to define the real model, regarding both the adversary and network.
- The main thing is to realistically (and conservatively) model the real world scenario and adversarial threats.



Some protocols

- We will discuss some protocols now.
- But we will study their security only informally.
- Note: Security proofs are important. Hand waving proofs (like in this lecture) are no substitute for rigorous mathematical proofs.
- There are often counter intuitive facts in proofs. We may miss them without rigorous proof process.

Yao's Millionaire problem

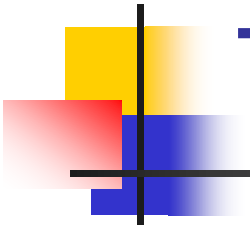
- Two millionaires wish to figure out who is wealthier.

- They do not want to reveal any other information.

Tools and conditions:

- Range of the inputs known: (0, N)
- Recap:
  - A : public key e, private key d
  - B : can access e, but not d
- And
  - $Dec_d(Enc_e(x)) = x$
  - $Dec_d(Enc_e(x)+y)$ = random looking thing

    (for someone who knows x,y,e but not d)



The protocol

Step 1

- A has i and B has j
- B generates a random x (of m bits)
- $C = Enc_e(x)$
- u =C- (j-1)
- Send u to A.

The protocol …contd.

- Step 2
- A computes: for (t= 1 to N)

$$y_m = Dec_d(u+t)$$

- Takes a prime p (of size about $\sqrt{m}$) and computes

$$z_i = y_i \bmod p \quad \ldots \text{ for } i = 1 \text{ to } N$$

- p chosen such that $|z_m - z_n| \geq 2$ for any m,n in [1 to N]

The protocol … contd.

- Step 3
- A send to B the following list:

P, z_1, z_2, …. z_i, $(z_{i+1}+1)$, $(z_{i+2}+1)$, …, (z_N+1).

- Bob compares the j^{th} entry of this list (excluding the prime p) with x mod p.
- If x mod p is = j^{th} entry of the list

  implies $i \geq j$.

The protocol … contd.

- If exact answer is required, then run the protocol in reverse direction.
- Rough security argument (remember, it is not a substitute for rigor. Only meant to appeal to our intuition)
  - The view of A:

    (C-j+1), and the final answer.
  - The view of B:

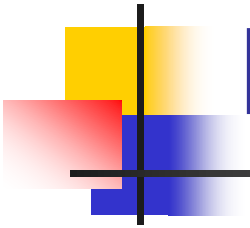
    z_m sequence. Everything other than z_j looks random to B.

Comments on the security…

- B could try to cheat …
- He was earlier computing $Enc_e(x) = C$ and then sending $(C-j+1)$
- He could try to compute

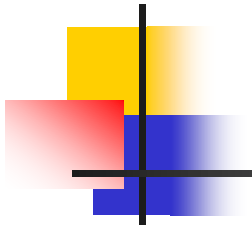
  $$Enc_e(\delta) = (C-j+9).$$

- If he succeeds then he knows $y_9 = \delta$.
- Comparing with z_9, he knows if $i \geq 9$?
- However, he can't do this in poly time.

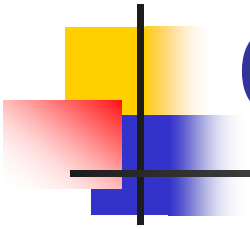


Efficiency

- Let us assume we wish to cover the range 1 to 1,00,000,000 at a resolution of 1 unit.
- A sends to B a list containing 1,00,000,000 entries, each of which is (at least) an 8-byte integer.
- Overheads + this data = many Gigabytes.
- Processing at the end of A = encrypting 1,00,000,000 entries.

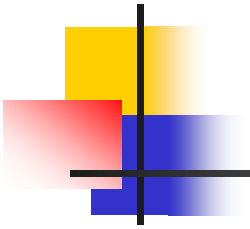


Oblivious Transfer



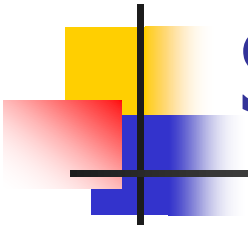
Oblivious Transfer

- A has message m, B wants to access it
- Should happen 50% of the time, without A knowing when B accesses it
- Original idea by Rabin (1981)
- Later made into more useful form by Even, Goldreich, Lempel
  - 1 out of 2 OT.
  - (Crepau) both notions are equivalent.



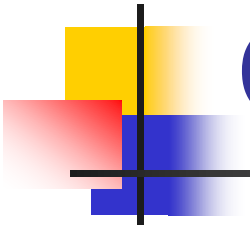
1-out-of-2 Oblivious Transfer

- Inputs
  - Sender has two messages m_0 and m_1
  - Receiver has a single bit $\sigma \in \{0,1\}$
- Outputs
  - Sender receives nothing
  - Receiver obtain m_σ and learns nothing of $m_{1-\sigma}$



Some mathematics…

- Square root modulo prime p
  - Z_p^* = {1,2, …., p-1} under * is a group
  - x is called Quadratic residue mod p if $y^2 = x \bmod p$ for some y
  - x is called Quadratic Non-residue mod p if $y^2 \neq x \bmod p$ for any y
  - Notation: QR mod p and QNR mod p

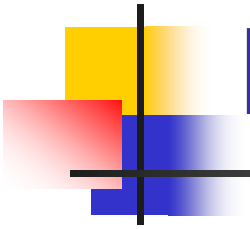


Contd…

- $x^{p-1} = 1 \bmod p$
- p is odd hence (p-1)/2 is an integer

- $y_1 = x^{(p-1)/2}$ is a square root of 1
- $y_2 = p - x^{(p-1)/2}$ is another square root of 1

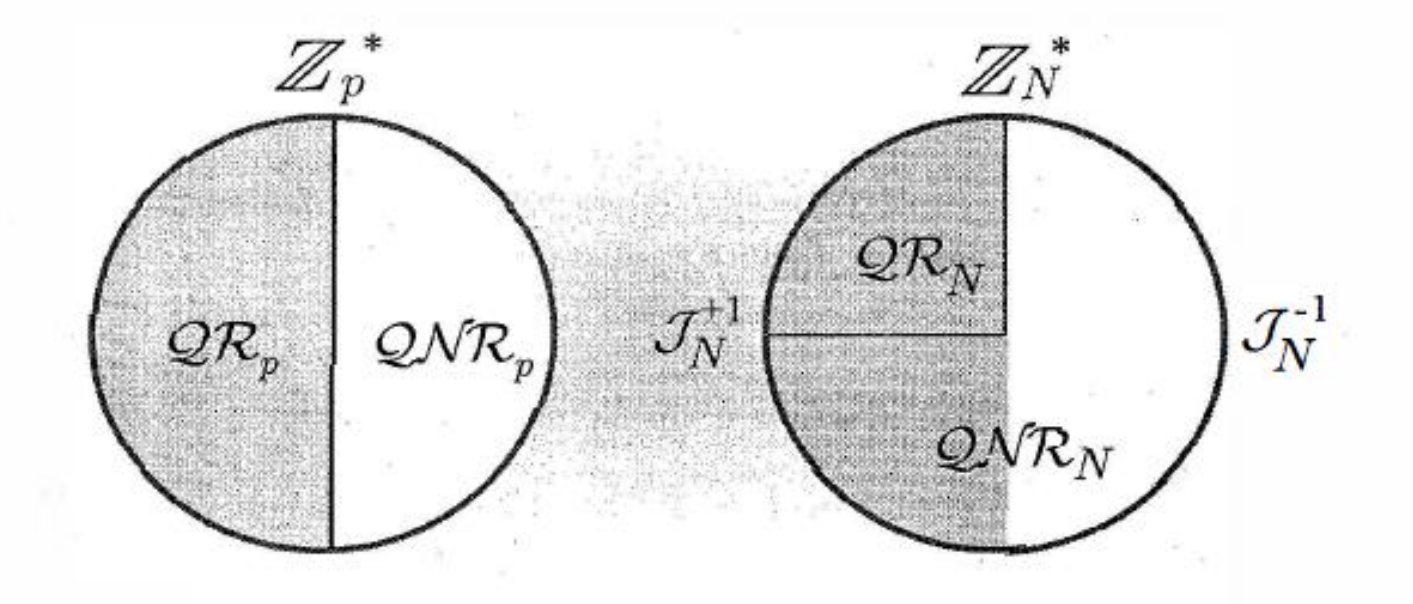
- Any number x has either 2 square roots or 0 square roots modulo p



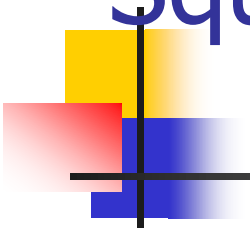
More on square roots

- Square roots modulo N= p*q where p, q are primes
- Find sq root mod p, and mod q (there are 2 or 0 for each case)
- Combine using Chinese remainder theorem
- These are the 4 or 0 solutions mod N

Structure of Z^*_p and Z^*_N

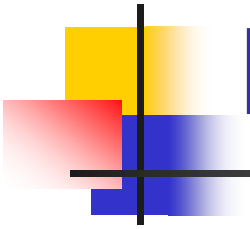


Square roots modulo a composite no



- Computing square roots mod N is easy if factors of N are known
- What if the factors of N are not known ?
- Hard problem !
- Knowing all square roots, one can factor N

Rabin's OT protocol

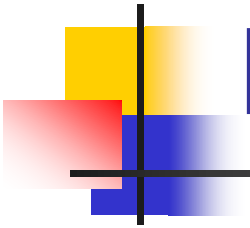


- N=pq, p,q large primes, both 3 mod 4
- Sender uses RSA encryption
  - (any one can decrypt is factors of N are known)
- Sender sends $c = Enc_e(m)$
- Receiver chooses random x, and sends

$$y = x^2 \bmod N$$

- With high probability gcd(x,N)=1.
- => There are 4 square roots of y mod N.

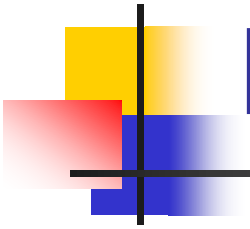
Rabin's OT protocol …



- Sender knows factors of N, and hence can find all 4 square roots of y mod N.
- Sender returns z = one square roots of y to the receiver
- If the root is ±x
  - The receiver learns nothing
- If the root is not ±x
  - then receiver can use this to decrypt m
  - And he gets m

Rabin's OT protocol… contd…

- Any number mod p has either 2 or 0 square roots. (QR or QNR mod p)
- By our choice, we ensure that y is QR mod p as well as mod q. (Recall CRT)
- Hence, y has 4 roots mod N
- These roots are x, -x, z, -z
- (mod p, mod q) these are: (a,b) .{(1,1), (-1,-1), (1,-1), (-1,1)}.
- How ? Hint: What is x^2-z^2 mod N ?



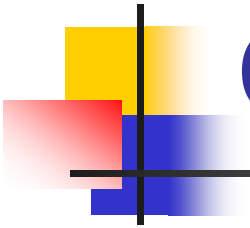
Rabin's OT protocol

- With probability 0.5, the receiver gets to know m
- The sender has no idea if the receiver knows m.

EGL Protocol

Protocol for 1-2 Oblivious Transfer

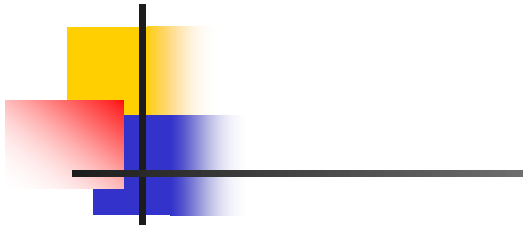
- Sender has m_0, m_1. chooses x_0, x_1 randomly and sends to Receiver
- Receiver chooses b ϵ {0,1} and a random k
- Generates $v = (x_b + Enc_e(k)) \bmod N$ and sends
- Sender computes $k_i = Dec_d(v - x_i) \bmod N$ (one of them is k, but sender does not know which one)
- Sends $m_i' = m_i + k_i$,. Receiver: $m = m_b' - k$



Generalization

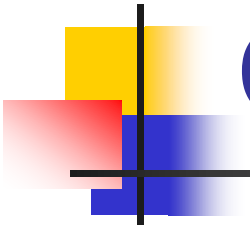
- Can define 1-out-of-n oblivious transfer
- and k-out-of-n oblivious transfer

- (i) by Naor, Pinkas, ….
- (ii) by Brassard et al



Coin tossing over telephone

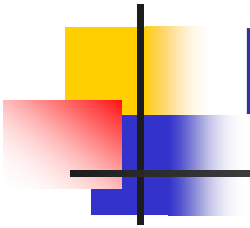
Coin tossing over telephone



- Alice commits to a bit b, sends some commitment value so that she can't change her mind later
- Bob can verify, when Alice decommits, whether she is speaking truth
- Properties:
  - (i) Binding
  - (ii) Hiding

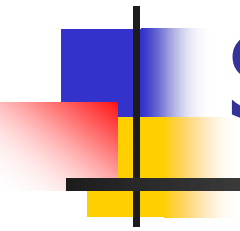
Coin tossing over telephone

- Alice generates primes p,q, N=pq
- Bob generates random x and sends $c=x^2 \bmod N$
- Alice finds 4 square roots of c and sends one of them
- If Alice chooses x, she wins
- Otherwise Bob wins (by revealing his x)
- Security: Alice can take roots, Bob can't

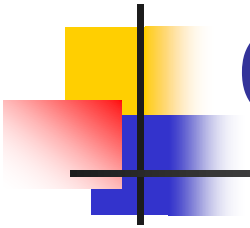


PIR

- PIR = Private Information Retrieval
- Aim: allow a client to query one element of a database (of size n) without the server learning what the query is.
- The query complexity should be sub-linear in n



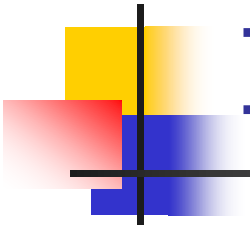
Some Feasibility Results



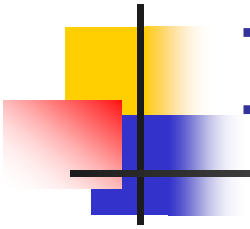
Computational Setting

- Any two-party function can be securely computed in the semi-honest adversarial model [Yao]
- Any multiparty function can be securely computed in the malicious model, for any number of corrupted parties [GMW]
- Remarks:
  - Above results assume the existence of trapdoor permutations
  - With an honest majority, fairness and guaranteed output delivery are achieved. Otherwise, not.
  - Model: static corruptions, authenticated channels, polynomial-time adversary, stand-alone…

Information Theoretic Setting

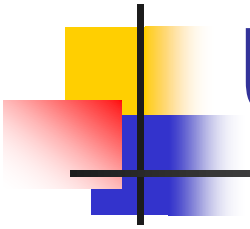


- Assuming a 2/3 honest majority, any multiparty function can be securely computed in the malicious model [BGW,CCD]
- Assuming a (regular) honest majority and a broadcast channel, any multiparty function can be securely computed in the malicious model [RB]
- Remarks:
  - Above results do not assume any complexity assumptions
  - Model: adaptive corruptions, perfectly private and authenticated channels, unbounded adversary, stand-alone…



Interpretation

- In the models described above, **any distributed task can be securely computed.**
- These are fundamental, and truly amazing, results.
- Theorem: any functionality f can be securely computed in the semi-honest model.



Useful References

- Oded Goldreich. *Foundations of Cryptography Volume 1 – Basic Tools.* Cambridge University Press.
  - Computational hardness, pseudorandomness, zero knowledge
- Oded Goldreich. *Foundations of Cryptography Volume 2 – Basic Applications.* Cambridge University Press.
  - Chapter on secure computation