# Public-Key Cryptography

Gianluigi Me
me@disp.uniroma2.it
Anno Accademico 2008/09

Main sources: Network Security Essential / Stallings
Applied Cryptography / Schneier
Handbook of Applied Cryptography / Menezes, van Oorschot, Vanstone

## Motivation

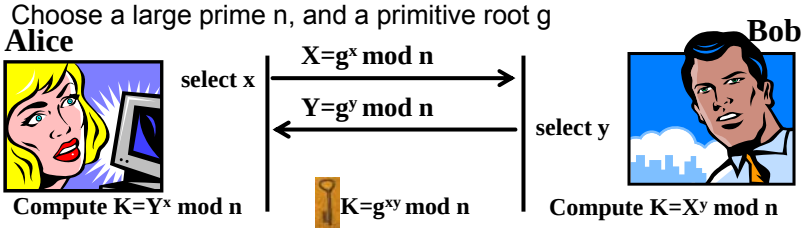
- Until early 70s, cryptography was mostly owned by government and military
- Symmetric cryptography not ideal for commercialization
  - Enormous key distribution problem; most parties may have never physically met
  - Must ensure authentication, to avoid impersonation, fabrication
- Few researchers (Diffie, Hellman, Merkle), in addition to the IBM group, started exploring Cryptography because they realized it is critical to the forthcoming digital world
  - Privacy
  - Effective commercial relations
  - Payment
  - Voting

# Public-Key Cryptography

- First proposed by Diffie and Hellman, and independently by Merkle (1976)
  - Idea: use separate keys to encrypt and decrypt
  - Merkle proposed puzzles, and then knapsack problems
- Pair of keys is generated by each user
  - Public key is advertised
  - Private key is kept secret, and is computationally infeasible to discover from the public key and ciphertexts
  - Each key can decrypt messages encrypted using the other key
- Applications:
  - Encryption
  - Authentication (Digital Signature)
  - Key Exchange (to establish Session Key)

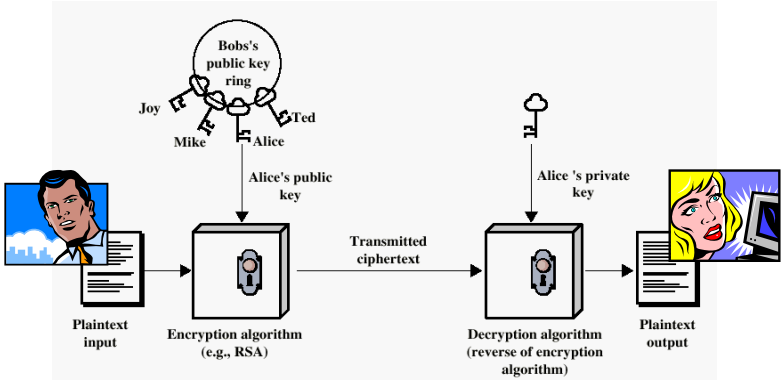
# Diffie-Hellman Key Exchange

- First public-key algorithm, based on the difficulty of computing discrete logarithms modulo n
- Protocol:
  - Use key exchange protocol to establish session key
  - Use session key to encrypt actual communication
- Algorithm:
  - Choose a large prime n, and a primitive root g



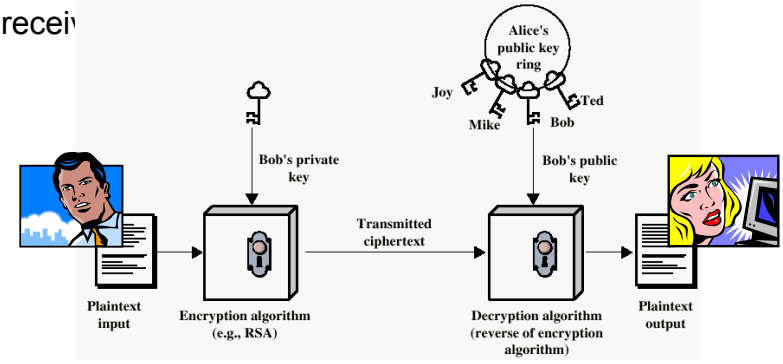
# Public-Key Encryption

- Sender uses the public key of the receiver to encrypt
- Receiver uses her private key to decrypt




# Authentication Using Public-Key

- The sender encrypts the message with his own private key
- The recei



# Public-Key Algorithms: Requirements

- It is computationally easy to generate a pair of keys
- It is computationally easy to encrypt using the public key
- It is computationally easy to decrypt using the private key
- It is computationally infeasible to compute the private key from the public key
- It is computationally infeasible to recover the plaintext from the public key and ciphertext
- Either of the related keys can decrypt a message encrypted using the other key

- Note: it should be computationally infeasible to decrypt using same key used for encryption

# RSA

- Developed by Rivest, Shamir, and Adleman (1977), and is most widely used
  - Classified version of RSA developed by GCHQ (Ellis and Cocks) in 1973

- Gets its security from the difficulty of factoring large numbers
- Works as a block cipher, where each plaintext/ciphertext block is integer between 0 and *n*
- Algorithm:
  - Receiver chooses *e*, *d*
  - The values of *e*, and *n* are made public; *d* is kept secret
  - Encryption: $C=M^e \bmod n$
  - Decryption: $M=C^d \bmod n = M^{ed} \bmod n$
- Requisite:
  - Find *e*, *d* such that $M=M^{ed} \bmod n$, for all $M<n$
  - Make sure that *d* cannot be computed from *n* and *e*, not even if a ciphertext is available

# RSA Key Generation

- Select primes $p$ and $q$, $n = pq$
- Calculate $\Phi(n) = (p-1)(q-1)$
  - Euler totient of $n$ – number of integers between 1 and $n$ that are relatively prime to $n$, i.e., $\{m \mid gcd(m,n)=1\}$
- Select integer $e < \Phi(n)$ such that $gcd(\Phi(n),e) = 1$
- Calculate $d$ such that $d = e^{-1} \bmod \Phi(n)$,
  - i.e. $ed = 1 \bmod \Phi(n)$

- Note:
  - The message could have been encrypted with $d$ and decrypted by $e$

# RSA Key Generation: Why it Works

- Fermat's Little Theorem
  - For a prime $p$, $\forall a$ such that $0<a<p$, $a^{(p-1)} = 1 \bmod p$
- Euler's extension
  - For primes $p,q$, $\forall$ $a$ such that $gcd(a,pq) = 1$, $a^{(p-1)(q-1)} = 1 \bmod pq$
  - Hence, $M^{ed} \bmod n = M^{k(p-1)(q-1)+1} \bmod n = 1xM = M$
- To generate primes, use primality test
  - For a non-prime, Fermat's theorem will usually fail on a random $a$
    - Carmichael numbers are very rare exception, and if chosen decryption wont work. Can reduce the probability by checking more $a$'s
  - Primes are dense enough (almost one of every k k-bit numbers)
- GCD to select $e$ takes $O(log\ n)$ time
- Calculate $d = e^{-1} \bmod n$ using Euler extended GCD algorithm
- Exponentiation (Encrypt/Decrypt) takes $O(log\ n)$ time

- RSA gets its security from the difficulty of factoring $n = pq$

# RSA Example

- Key Generation
  - Select $p = 7$, $q = 17$, $n = pq = 119$, $\Phi(119) = 96$
  - Select $e = 5$; Calculate $d = 77$

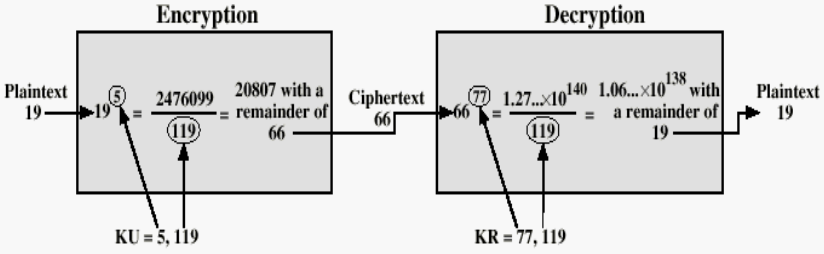



Figure 3.9 Example of RSA Algorithm

# Attacks on RSA Algorithm

- If one could factor *n,* which is available, into *p* and *q*, then *d* could be deduced, and then the message deciphered
- If one could guess the value of *(p-1)(q-1),* even without factoring n, then again *d* could be deduced

# Attacks on RSA Protocol

- Chosen ciphertext attack
  - Attack: get sender to sign (decrypt) a chosen message
  - Inputs: original ciphertext $C = M^e$
  - Construct
    - $X = R^e \bmod n$, for a random R
    - $Y = XC \bmod n$
    - $T = R^{-1} \bmod n$
  - Ask sender to sign Y, obtaining $U = Y^d \bmod n$
  - Compute
    - $TU \bmod n = R^{-1}Y^d \bmod n = R^{-1} X^d C^d \bmod n = C^d \bmod n = M$
  - Exploits preservation of multiplication under mod
- Conclusion:
  - never sign a random message
  - sign only hashes
  - use different keys for encryption and signature

# Other precautions when implementing RSA protocol

- Do not use same *n* for multiple users
  - Can decipher using two encryption (public) keys, without any decryption key
- Always pad messages with random numbers, making sure that M is about same size as *n*
  - If e is small, there is an attack that uses *e(e+1)/2* linearly dependent messages
- Do not choose low values for *e* and *d*
  - For e, see above, and there is also attack on small *d*'s

# Other Public-Key Algorithms

- Merkle-Hellman Knapsack Algorithms
  - First public-key cryptography algorithm (1976)
  - Encode a message as as series of solutions to knapsack problems (NP-Hard). Easy (superincreasing) knapsack serves as private key, and a hard knapsack as a public key.
  - Broken by Shamir and Zippel in 1980, showing a reconstruction of superincreasing knapsacks from the normal knapsacks
- Rabin
  - Based on difficulty of finding square roots modulo n
  - Encryption is faster: $C=M^2 \bmod n$
  - Decryption is a bit complicated and the plaintext has to be selected from 4 possibilities
- El Gamal
  - Based on difficulty of calculating discrete logarithms in a finite field
- Elliptic Curves can be used to implement El Gamal and Diffie-Hellman faster

# Digital Signatures

Main sources: Network Security Essential / Stallings
Applied Cryptography / Schneier
Handbook of Applied Cryptography / Menezes, van Oorschot, Vanstone

# Digital Signature

Key requirements which digital signature must fulfill, and is shown below.

- Authentication: digital signature must ensure strong authentication of the signing entity.
- Signature Non-Repudiation: once a signature is made, the owner entity cannot repudiate the signed document.
- Document Integrity: once a document is signed, a document change must produce a different signature.
- Untransferable Signature: a given signature, must be strictly connected with its document.

# Digital Signature

The main steps of a signature algorithm are shown below:

- 1. *Generation of document hash code:* in this step, the imprint of the document to be signed, is computed. The hashing algorithm, must ensure that different documents produce different hashes and must be impossible the document content recovering from its hash.
- 2. *Document imprint encoding:* in this step, the private key of the sender is used to encode the document imprint computed above. The resulting code represents the document signature.
- 3. *Signature affixing:* once the signature is computed, it must be docked to its document.

# Digital Signature

The main steps of a verification algorithm are shown below:

- *Generation of document hash code:* in this step, the imprint of the incoming document to be verified, is computed. The hashing algorithm must be the same used in the signature algorithm.
- *Document imprint decoding:* in this step, the public key of the sender is used to decode the incoming document imprint.
- 3. *Hashing comparison:* in this step, a comparison between decoded and computed hash codes is performed. If these codes are equal, the verification is passed.

# Digital Signature

$$T =< P, A, K, S, V >$$

P is the finite set of all documents;

- A is the finite set of all signatures;
- K is the finite set of all keys;
- S is the finite set of all signature algorithms;
- V is the finite set of all verification algorithms;
- $\forall k \in k$ does exist a signature algorithm $sig_k \in$ S and its verification algorithm $ver_k \in$ V .

$$sig_k : P \rightarrow A$$

$$ver_k : P \times A \rightarrow true|false$$

# Digital Signature

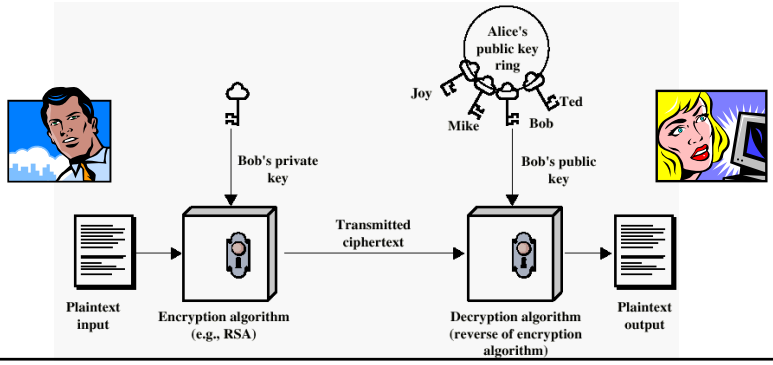
- Verification:

$$\forall(p,a) \mid p \in P, a \in A \rightarrow ver_k(p,a) = \begin{cases} true & \text{if } a = sig_k(p) \\ false & \text{otherwise} \end{cases}$$

- Standards:
  - FIPS 186-1/2
  - ANSI X9.31/9.62/9.63

# Public-Key Digital Signature

- The sender encrypts the message with his own private key
- The receiver, by decrypting, verifies key possession



## Digital Signatures

- The entire message, encrypted with the private key, serves as the digital signature
  - Computationally expensive
  - Anyone can decrypt the original message
- Alternatively, a *digest* can be used
  - Should be short
  - Prevent decryption of the original message
  - Prevent modification of original message
  - Difficult to fake signature for
- A hash code of the message (e.g., SHA-1)
- If only source authentication is needed, a different message can be used

## Digital Signature Algorithm (DSA)

- Proposed in 1991 by NIST as a standard (DSS)
- Based on difficulty of computing discrete logarithms (like Diffie-Hellman and El Gamal)
- Encountered resistance because RSA was already de-facto standard
  - Cannot be used for encryption or key distribution
  - Faster than RSA in signature, but slower in verification
  - Significant investment in RSA by large corporations
  - Concerns about NSA backdoor
- Key size was increased from 512 to up-to 1024 bits

# Description of DSA

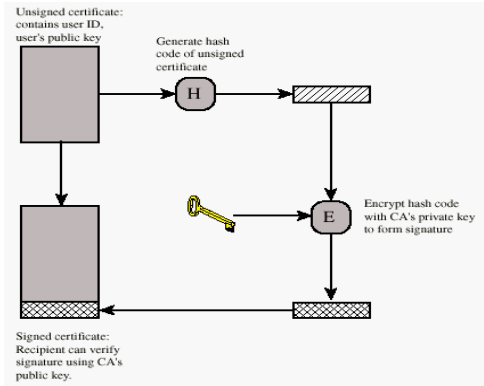
- Public parameters
  - p is a prime number with up to 1024 bits
  - q is a 160-bit factor of (p-1), and itself prime
  - $g=h^{(p-1)/q} \bmod p$
  - x is the private key and is smaller than q
  - $y=g^x \bmod p$ is the public key
  - H(M) is the secure hash code of the message
- Signature
  - Generate a random k<q
  - Compute and send $r=(g^k \bmod p) \bmod q$
  - Compute and send $s=k^{-1}(H(M)+xr) \bmod q$
- Verification
  - Compute $w=s^{-1} \bmod q$
  - Compute $u_1=H(M)w \bmod q$; $u_2=rw \bmod q$
  - Compute $v=(g^{u1}*y^{u2} \bmod p) \bmod q$
  - If v=r then the signature is verified

# Key Management for Public-Key Cryptography

Main sources: Network Security Essential / Stallings
Applied Cryptography / Schneier
Handbook of Applied Cryptography / Menezes, van Oorschot, Vanstone

# Certificate Authority: Verifying the Public Key

- How to ensure that Charles doesn't pretend to be Bob by publishing a public-key for Bob. Then, using a Man-in-the-Middle attack, Charles can read the message and reencrypt-resend to Bob
  - Bob prepares certificate with his identifying information and his public key (X.509)
  - The Certificate Authority (CA) verifies the details and sign Bob's certificate
  - Bob can publish the signed certificate




# More on Key Management

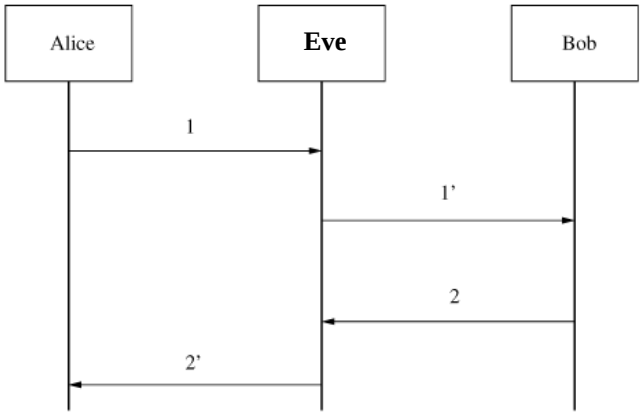
- Alice may have more than one key
  - e.g., personal key and work key
- Where shall Alice store her keys
  - Alice may not want to trust her work administrator with her personal banking key
- Distributed certification V1.0
  - CA certifies Agents who certify companies who certify employees
- Distributed Certification V2.0 (a la PGP)
  - Alice will present her certificate with "introducers" who will vow for her
- Key Escrow
  - US American Escrowed Encryption Standard suggests that private keys be broken in half and kept by two Government agencies
  - Clipper – for cellular phone encryption
  - Capstone – for computer communication

# Diffie Hellmann protocol

## The protocol

1 Alice picks a $\varepsilon_U$ [1, p – 1], computes $g_a \leftarrow g^a \pmod p$; she sends $g_a$ to Bob;
2 Bob picks b $\varepsilon_U$ [1, p – 1], computes $g_b \leftarrow g^b \pmod p$; he sends $g_b$ to Alice;
3 Alice computes $k_2 \leftarrow g_m^b \pmod p$;
4 Bob computes $k_1 \leftarrow g_a^m \pmod p$.

# The attack




# The attack

Alice picks a $\varepsilon_U$ [1, p – 1], computes $g_a \leftarrow g^a \pmod p$; she sends $g_a$ to Eve ("Bob");
1' Eve("Alice") computes $g_m \leftarrow g^m \pmod p$ for some m $\varepsilon$ [1, p – 1]; he sends $g_m$ to Bob;
2 Bob picks b $\varepsilon_U$ [1, p – 1], computes $g_b \leftarrow g^b \pmod p$; he sends $g_b$ to Eve("Alice");
2' Eve("Bob") sends to Alice: $g_m$;
3 Alice computes $k_1 \leftarrow g_a^m \pmod p$;
(* this key is shared between Alice and Eve since Eve can compute $k_1 \leftarrow g_a^m \pmod p$.*)
4 Bob computes $k_2 \leftarrow g_m^b \pmod p$.
(* this key is shared between Bob and Eve since Eve can compute $k_2 \leftarrow g_m^b \pmod p$.*)


Denaro digitale e
firme cieche


Privacy & Anonymity

# Privacy

*"We're an information economy. They teach you that in school. What they don't tell you is that it's impossible to move, to live, to operate at any level without leaving traces, bits, seemingly meaningless fragments of personal information. Fragments that can be retrieved, amplified..."*

**– William Gibson, "Johnny Mnemonic" (1981)**

# Definition

- The notion of privacy has been discussed at least as far back as Aristotle (384–327 BCE), who defined the separate spheres of public life: *polis* (city) and private life *oikos* (home).
- GARZANTI: *nella vita di una persona, la dimensione più privata, che essa ha diritto a salvaguardare*. Sinonimi: *intimità, privatezza, privato*. Di notizia, fatto: *riservatezza, segretezza*

## Privacy

- Privacy is the ability of a person to control the availability of information about and exposure of him- or herself. It is related to being able to function in society anonymously

Types of privacy giving raise to special concerns:

- Political privacy
- Consumer privacy
- Medical privacy
- Information technology end-user privacy; also called data privacy
- Private property

## Privacy

- Data Privacy problems exist wherever uniquely identifiable data relating to a person or persons are collected and stored, in digital form or otherwise. Improper or non-existent disclosure control can be the root cause for privacy issues.
- The most common sources of data that are affected by data privacy issues are:
- Health information
- Criminal justice
- Financial information
- Genetic information

## Privacy

- The challenge in data privacy is to share data while protecting the personally identifiable information.
- Consider the example of health data which are collected from hospitals in a district; it is standard practice to share this only in aggregate form
- The idea of sharing the data in aggregate form is to ensure that only non-identifiable data are shared.
- The legal protection of the right to privacy in general and of data privacy in particular varies greatly around the world.

## Privacy Concerns

- Biometrics (fingerprints, iris) and face recognition
- Video surveillance, ubiquitous networks and sensors
- Cellular phones
- Personal Robots
- DNA sequences, Genomic Data

## Privacy is not just Security

- Privacy is not just confidentiality and integrity of user data
- Privacy includes other requirements:
- Support for user preferences
- Support for obligation execution
- Usability
- Proof of compliance

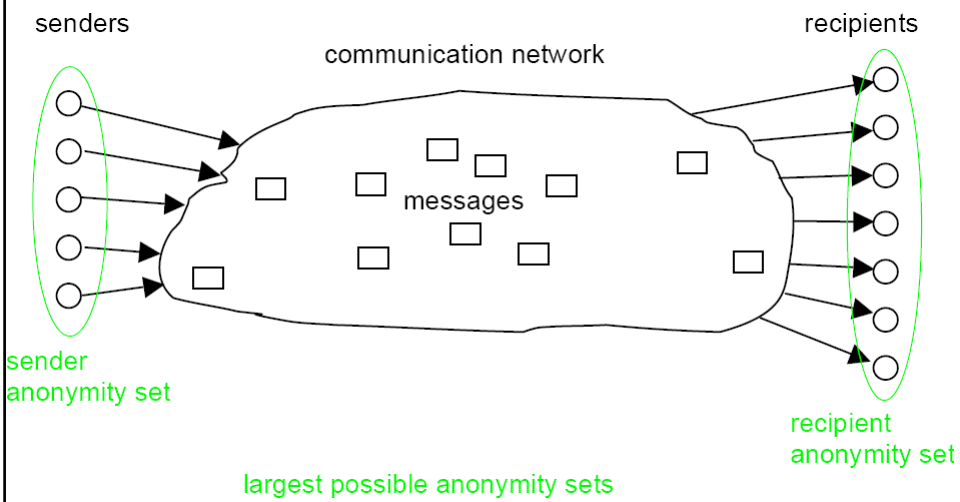
## Definition

- A *subject* is a possibly acting entity such as, e.g., a human being (i.e. a natural person), a legal person, or a computer.
- This generality is very fortunate to stay close to the everyday meaning of "anonymity" which is not only used subjects active in a particular context, e.g. senders and recipients of messages, but to subjects passive in a particular context as well, e.g. subjects the records within a database relate to.

# What is anonymity?

- To enable anonymity of a subject, there always has to be an appropriate set of subjects with potentially the same attributes.

# What is anonymity?



# What is anonymity?

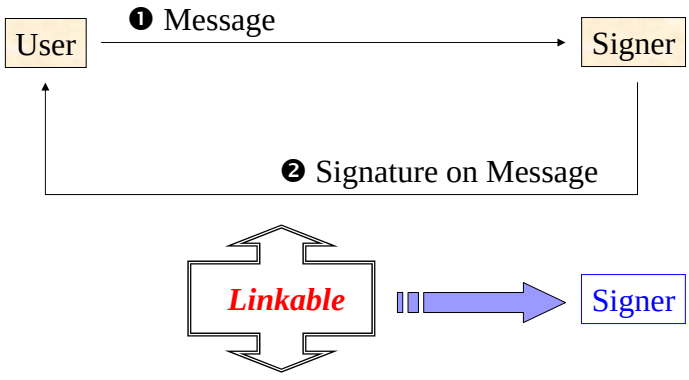
- ***Anonymity* is the state of being not identifiable within a set of subjects, the *anonymity set.***
- ISO IS 15408, 1999: “[Anonymity] ensures that a user may use a resource or service without disclosing the user’s identity. The requirements for anonymity provide protection of the user identity. Anonymity is not intended to protect the subject identity. [...] Anonymity requires that other users or subjects are unable to determine the identity of a user bound to a subject or operation.”

# What is anonymity?

- All other things being equal, anonymity is the stronger, the larger the respective anonymity set is and the more evenly distributed the sending or receiving, respectively, of the subjects within that set is.
- The entropy of a message source as defined by Claude E. Shannon [Shan48] might be an appropriate measure to quantify anonymity – just take who is the sender/recipient as the “message” in Shannon’s definition.
- For readers interested in formalizing what we informally say: “No change of probabilities” means “no change of knowledge” and vice versa. “No change of probabilities” (or what is equivalent: “no change of knowledge”) implies “no change of entropy”, whereas “no change of entropy” neither implies “no change of probabilities” nor “no change of knowledge”. In an easy to remember notation: No change of probabilities = no change ofknowledge -> no change of entropy

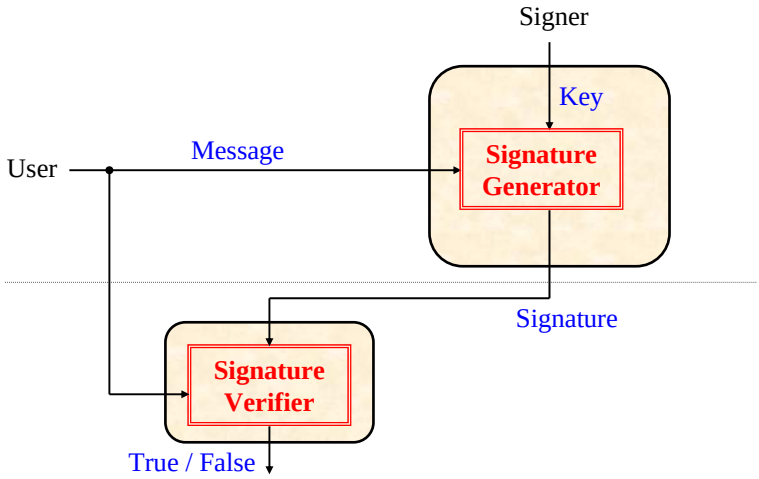
# Blind Signatures

# A Digital Signature Scheme

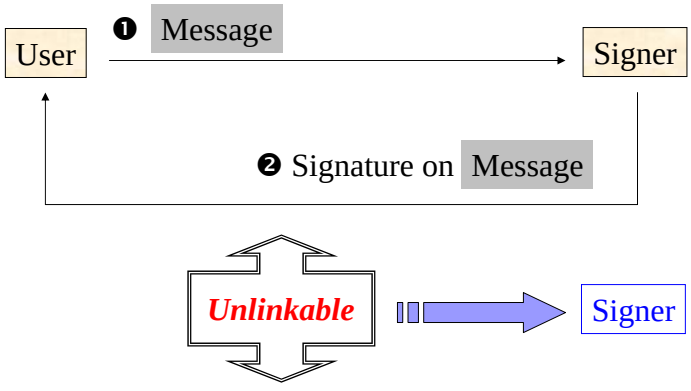



The signer's signature on "Message"

## Signature Generation and Verification




## Blind Signatures




❸ The signer's signature on "Message"

# The Scheme

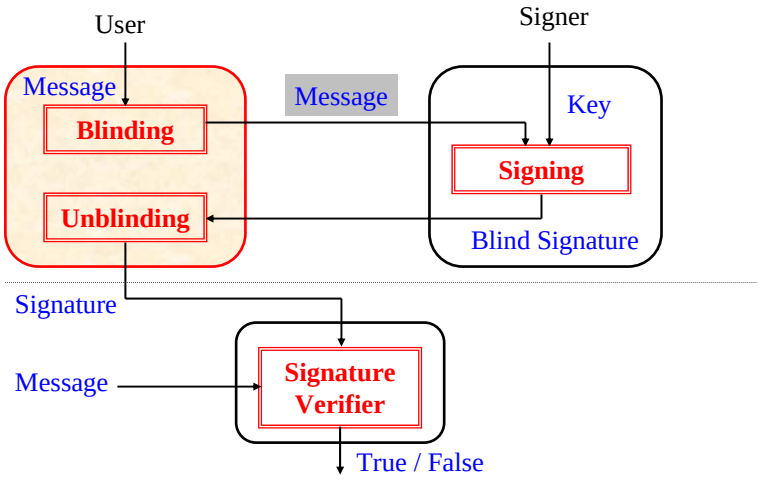
❶ "Message" : the **blinded** message

❷ Signature on "Message" : the **blind** signature

❸ Signature on "Message": to be obtained after **unblinding**

★ *Unlinkability*: it is intractable for the signer to link ❸ to ❷

# Signature Generation and Verification



## A Generic Blind Signature Scheme

- $M$ : the underlying set of messages
- $R$ : a finite set of random integers
- $S : M \rightarrow M^T$ : signing
- $V : M^T \times M \rightarrow \{true, false\}$ : verifying
- $B : M \times R \rightarrow M$ : blinding
- $U : M^T \times R \rightarrow M^T$ : unblinding

## RSA Blind Signatures

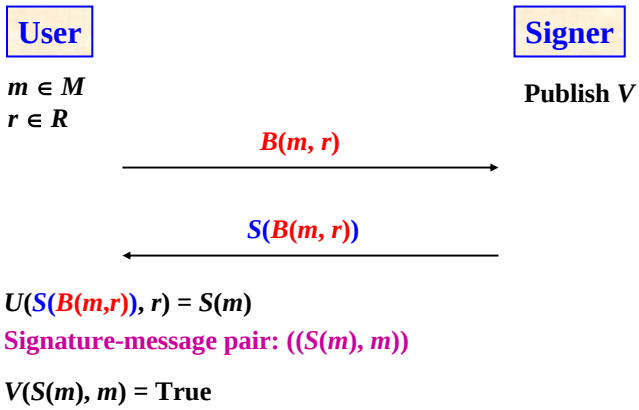
- Alice wants Bob to sign message M.
- She gives him $M*r^{e(bob)} \bmod n$
- Bob signs this giving Alice $s'=(M*r^{e(bob)})^{d(bob)} \bmod n$
- Alice can then remove the blind by calculating $s= s'*r^{-1} \bmod n$
  - $s = s'*r^{-1} = (M*r^{e(bob)})^{d(bob)} *r^{-1} = M^{d(bob)}*r^{e(bob)*d(bob)}*r^{-1}=M^{d(bob)} \bmod n$

# Example

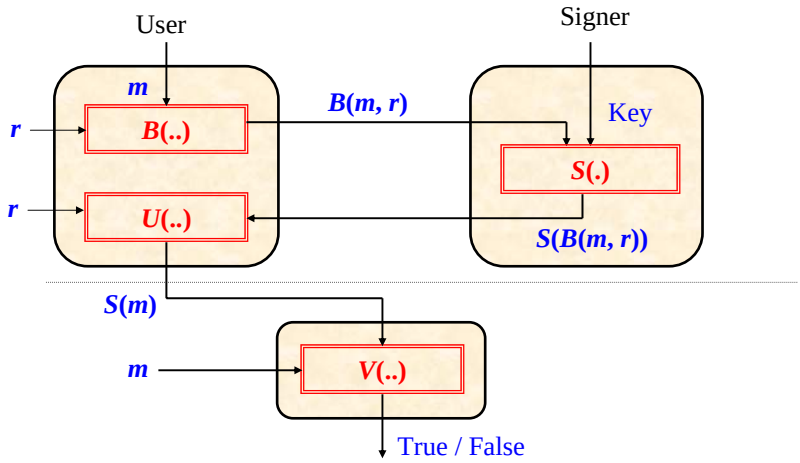
- Alice's Message: 28
- Bob's public key: 17
- Bob's private key: 53 (n = 77)

- Alice asks Bob to sign 70(=28*$6^{17}$ mod 77)
- Bob signs 70 and sends Alice 42
- Alice multiplies 42 by 13 (mod 77) to get 7
  - $28^{53}$ mod 77 = 7

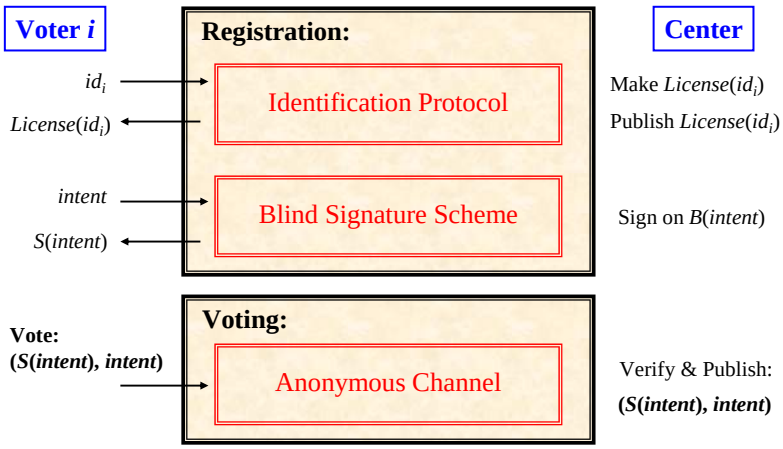
# The Protocol



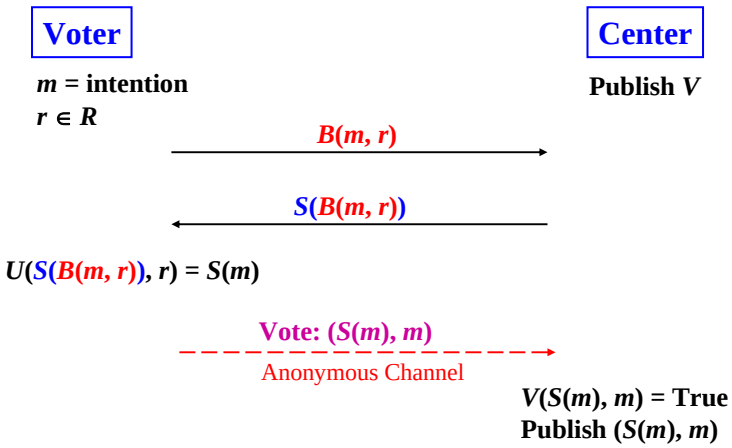
# Flow Diagram



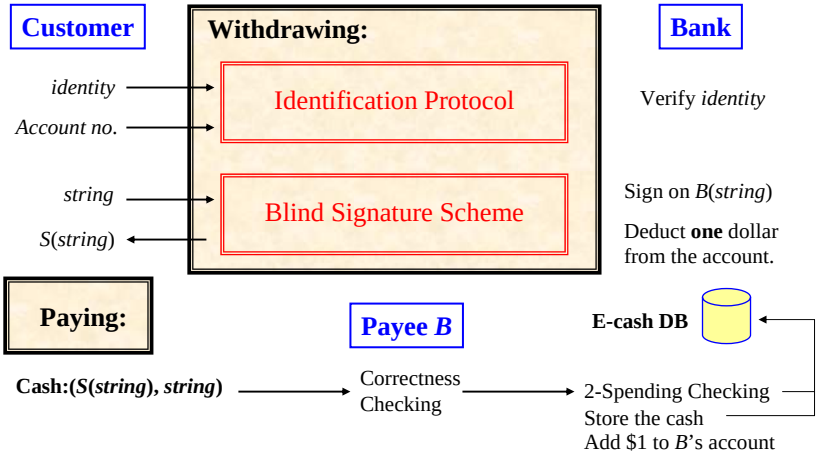

# Application: Anonymous Voting



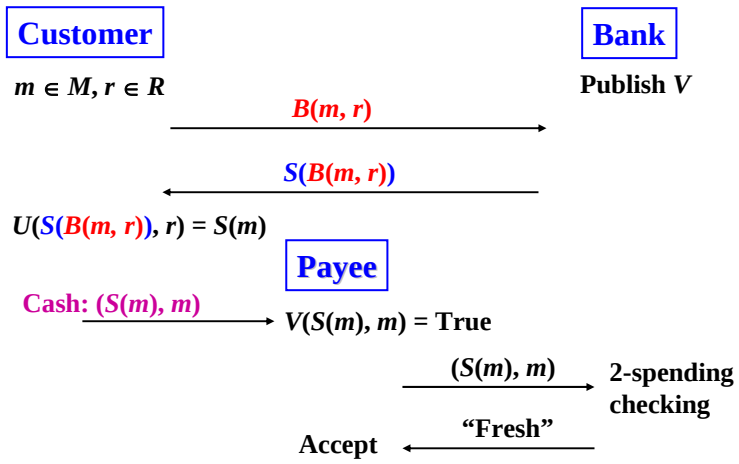
# An Anonymous Voting Protocol




# Application: Untraceable E-Cash

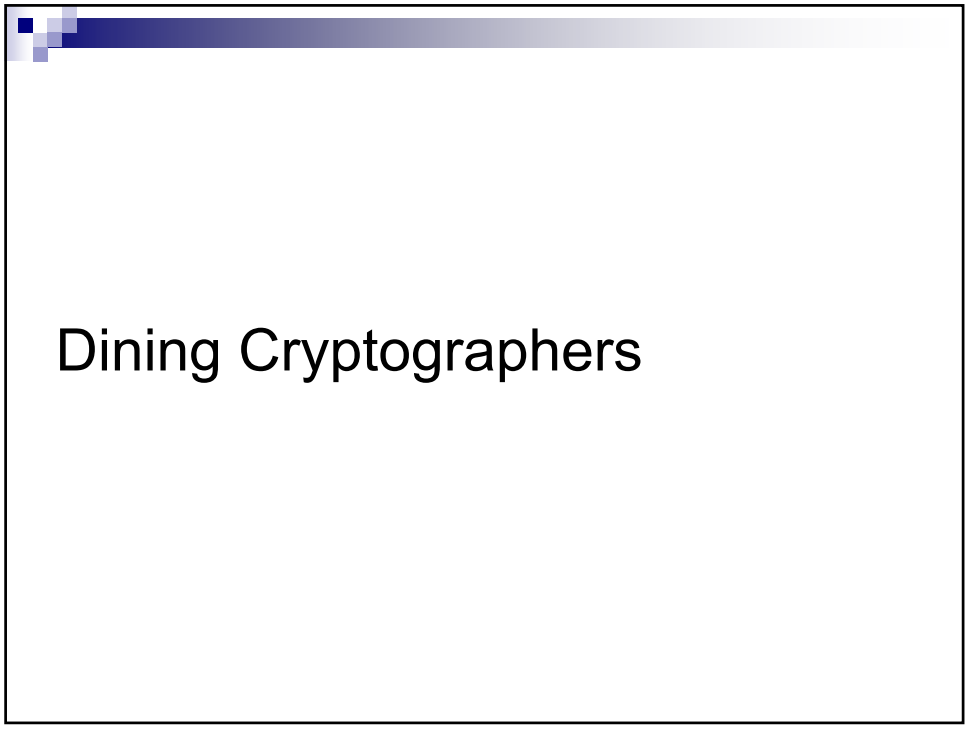


# An Untraceable E-Cash Protocol




# Discussions

- **Unforgeability**
- **Untraceability**
  - Bank cannot trace an e-cash to the withdrawing protocol
- Perfect crimes
  - Money Laundering
  - To safely get a ransom




## Dining Cryptographers

- Also introduced by Chaum
- Purpose is to release a public message in a perfectly untraceable manner

## Dining Cryptographers

- Clever idea how to make a message public in a perfectly untraceable manner
  - David Chaum. “The dining cryptographers problem: unconditional sender and recipient untraceability.” Journal of Cryptology, 1988.
- Guarantees information-theoretic anonymity for message senders
  - This is an unusually strong form of security: defeats adversary who has unlimited computational power
- Impractical, requires huge amount of randomness
  - In group of size N, need N random bits to send 1 bit

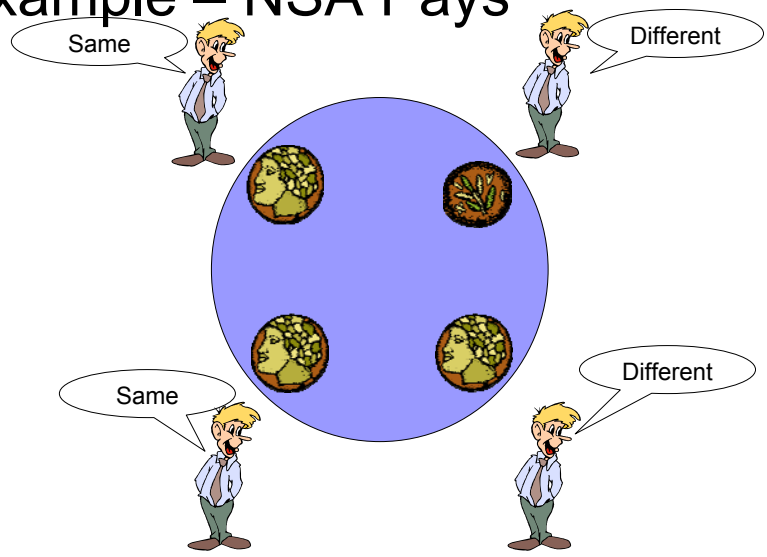
## The Protocol

- N cryptographers are having dinner
- Waiter tells them that the dinner has been paid for but they want to know whether it was one of them that paid or the NSA agent in the corner

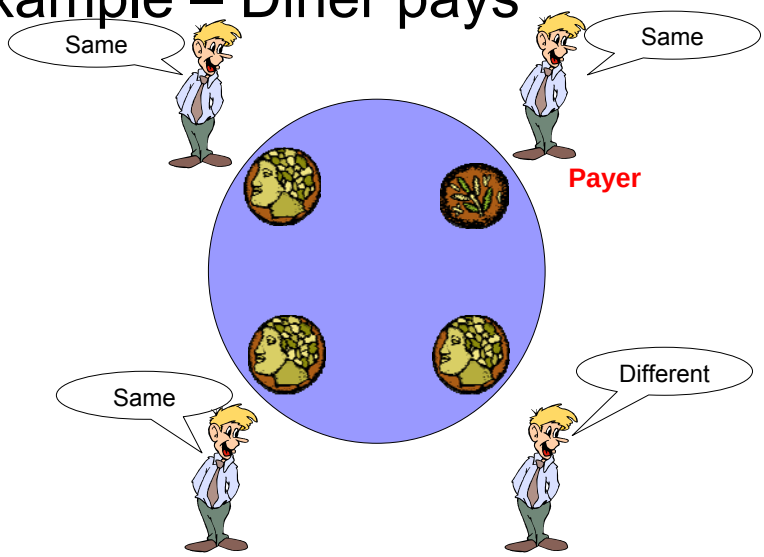
# The Protocol

1. Each diner flips a coin and shows it to his left neighbor
2. Each diner announces whether he and his neighbor's coin flips are the same or different. The payer lies
3. Odd number of "same" => NSA paper
   Even number of "same" => one the diners paid

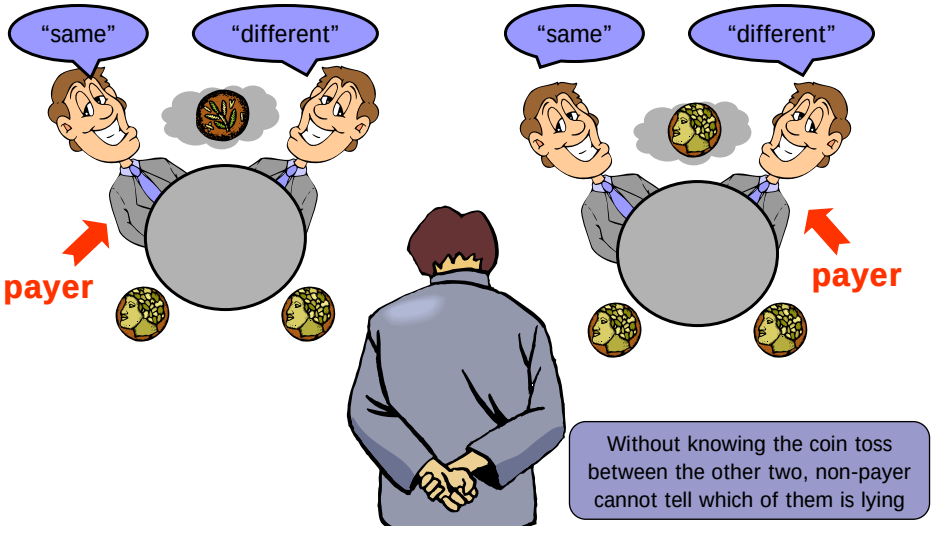
# Example – NSA Pays



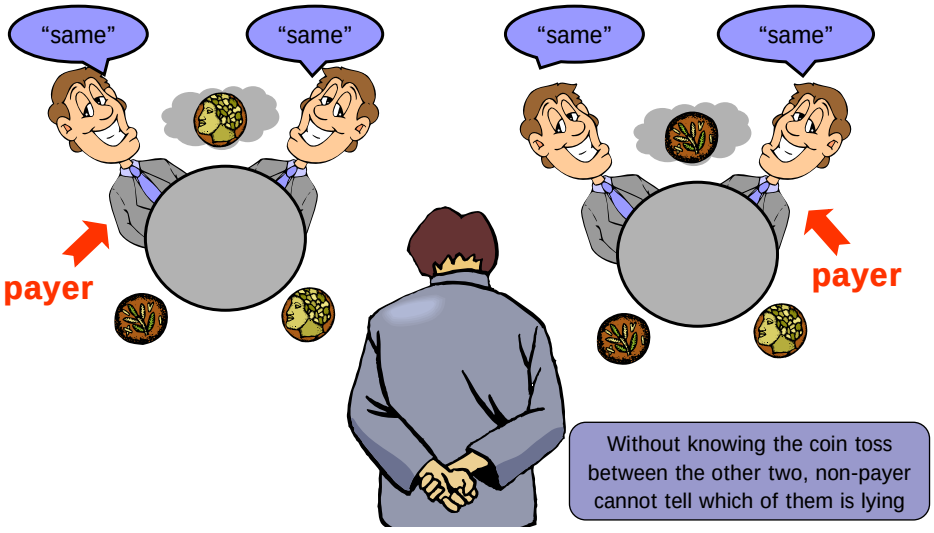
# Example – Diner pays




# Non-Payer's View: Same Coins

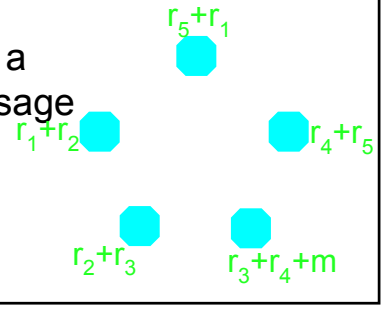


## Non-Payer's View: Different Coins




## Dining Cryptographers

- Nodes form a ring
- Each adjacent pair picks a random number
- Each node broadcasts the sum (xor) of the adjacent numbers
- The user who wants to send a message also adds the message
- The total sum (xor) is:
  $r_1+r_2+r_2+r_3+r_3+$
  $r_4+r_4+r_5+r_5+r_1+m = m$

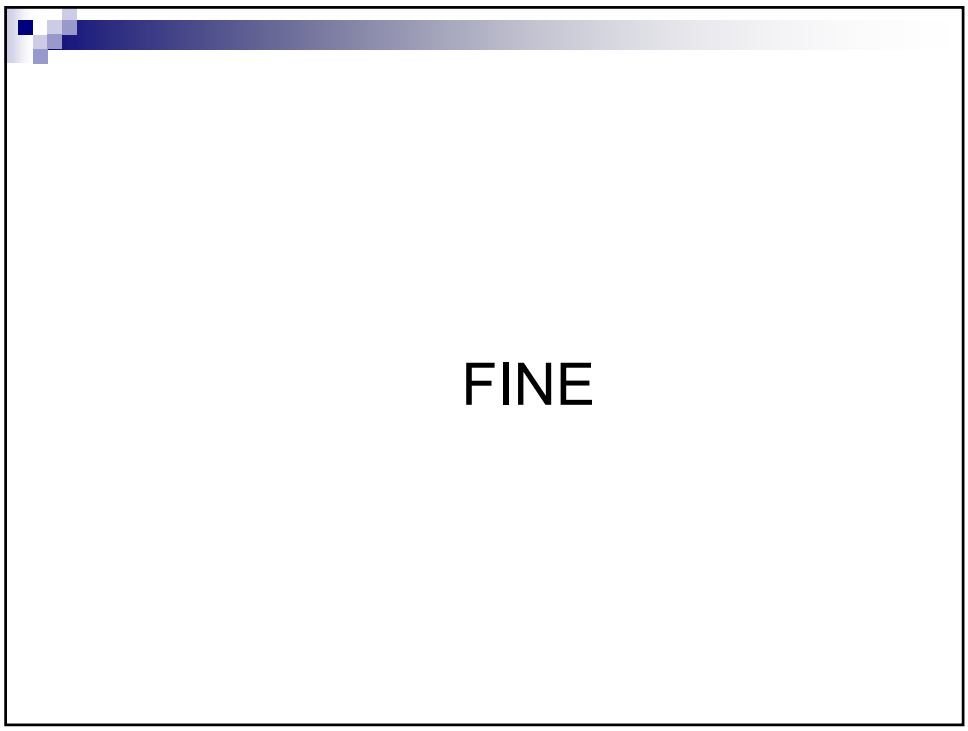


## Dinning Cryptographers

- It's impossible to tell who added m.
- Very inefficient: everyone must send the same amount of data as the real sender.

## Problems with DC

- Very Impractical
  - Only one bit sent at a time
  - Each party has to have pairwise secure channels
  - Massive communication overhead


FINE