

Crypto for PETs – Part 2

Jorge Cuellar

WS 18-19

Anonymity is

- ▶ lack of identification
- ▶ lack of leakage
- ▶ lack of traceability
- ▶ lack of distinguishability

Moreover:

- ▶ this lack of identification, linkability, etc. should hold
 - ▶ independently of any information the attacker may have



But what information could have an attacker?

- ▶ Example:
 - ▶ Consider the information (taken from a very large hospital):

Age	Diagnose
25	Cancer
37	...
...	...

Anonymity

In the example above

- ▶ Discuss: Is this data anonymous?
- ▶  Find a context where it is not anonymous
 - ▶ And allows you to find the diagnose of a person
- ▶  Delete the age column in the table above
 - ▶ Is **now** the data anonymous?
 - ▶ What extra information could someone have
 - ▶ which allows him
 - ▶ to find the diagnose of a person, from the DB?
- ▶ It is easy to create
 - ▶ trivial protocols
 - ▶ that provide "anonymity"
 - ▶ consider for instance the protocol that discloses nothing



Information as a change in probability

- ▶ We go not define information precisely
 - ▶ but we define:
- ▶ "Event F has no information about event E "
 - ▶ which means:
 - ▶ if I know wheather F happens or not
 - ▶ this tells me **nothing** about
 - ▶ wheather E happens or not
- ▶ More precisely,
 - ▶ the probability that event E happens
 - ▶ does not change, adding the information F :
 - ▶ $\text{Prob}[E] = \text{Prob}[E | F]$



Information as a change in probability

- ▶ Note that
 - ▶ " F has no information about E "
 - ▶ is the same as F and E are independent
 - ▶ and therefore E has no information about F



Information as a change in probability


- ▶ Be careful: Even if " F has no information about E "
- ▶ There still will probably be some "a-priori" information I
 - ▶ Or – in other words – some context or situation
 - ▶ (and in this context we gain some "a-priori knowledge")
 - ▶ and F has information about E under the information I
 - ▶ $\text{Prob}[E | I] = \text{Prob}[E | I \wedge F]$
- ▶ This makes privacy very difficult
 - ▶ and in a certain sense impossible
 - ▶ If you want to disclose some very innocent-looking information F
 - ▶ that you think is not privacy-relevant
 - ▶ Still, under some (perhaps strange) context
 - ▶ the information released will provide information E
 - ▶ which is clearly personal information



Information as a change in probability

- ▶ Assume you want to publish the result:
 - ▶ $E = \text{"smoking produces cancer"}$
 - ▶ (... and assume that nobody knew that)
 - ▶ (Or, assume your research shows
 - ▶ "eating green bananas produces cancer")
- ▶ Does this information tell anything about
 - ▶ the chances that $F = \text{"Peter Pan has cancer"}$?
- ▶ No, if you do not know wheather Peter Pan smokes
 - ▶ (or eats green bananas)
- ▶ But in a context where you know that he smokes
 - ▶ then E has information about F

Anonymity in Applications

- ▶ In each of the following applications
 - ▶ anonymity seems to be competing or in conflict with
 - ▶ the main functionality of the application
 - ▶ which depend on "identifiers"
 - ▶ security
 - ▶ lawful interception
 - ▶ accountability
 - ▶ trust and reputation
 - ▶ billing
 - ▶ routing
- ▶  For each one examples of the list above
 - ▶ find/explain a conflict situation with anonymity

e-voting, Secrecy

Is it possible to design a system

- ▶ to vote from home, via Internet?



Discussion: What are the requirements?

Secrecy (Privacy), Authorization, Uniqueness

A natural requirement is Secrecy:

Voters want to hide from **anybody** how they voted

- ▶ including the people organizing the votation
- ▶ or counting the votes
- ▶ This is a basic right in any modern democracy

Authorization

Only eligible (authorized) persons are able to vote

- ▶ The system must be able to verify that voters are eligible

Uniqueness

No voter should be able to vote more than once



Registration, Authentication, Eligibility

Registration process

During the registration process

- ▶ the authorities verify the Identification and Eligibility (Authorization)
 - ▶ of the voter
- ▶ The voter identifies himself to vote
 - ▶ with respect to the registration list or database
 - ▶ the voter has to show some legal identification document
- ▶ The voter registration is to be done in person
 - ▶ as a result of the registration the voter obtains a document or token
 - ▶ that shows that he is authorized to vote



Accessibility

Eligible voters must be able to vote

- ▶ The system must be easily accessible to the voter
 - ▶ example: if done via a PC at home the SW
 - ▶ must be easy to find, download, and verify and
 - ▶ must run on most common operating systems

Voters should not require special skills

- ▶ or the system should not intimidate the voter
 - ▶ to ensure "Equality of Access to Voters"

The system must be available

- ▶ without waiting "too long"



Availability

Availability

DoS Resilience: Protection against

- ▶ accidental and malicious denial of service attacks

Redundancy: Redundant communication paths, DBs, etc

- ▶ so that availability is ensured



Reliability (Accuracy)

Accuracy

Voting systems should record the votes correctly

- ▶ As intended by the voter
- ▶ The system shall record and count all the votes and
 - ▶ shall do so correctly
- ▶ Each (correctly cast) vote gets counted



Usability: Convenience, UI Friendliness

Convenience

Voters should be able to cast their votes quickly

- ▶ in one session

User-Interface

The system must be user-friendly

- ▶ from the voters' point of view

The system shall provide an **easy-to-use** user-interface

- ▶ It shall not disadvantage any candidate while displaying the choices
 - ▶ say, by requiring the user to scroll down to see the last few choices



Usability: Transparency, Documentation and Assurance, Cost-effectiveness

Transparency

- ▶ Voters should be able to possess a
 - ▶ general knowledge and understanding of the voting process

Documentation and Assurance

- ▶ The design, implementation, and testing procedures
 - ▶ must be well documented so that the voter-confidence in the election process is ensured

Cost-effectiveness

- ▶ Election systems should be affordable and efficient



Protection against incorrect casting

Voter Confirmation (during voting)

- ▶ Voter shall be able to confirm clearly how his vote is being cast
 - ▶ and must have chance to modify his vote before he commits it

No Over-voting

- ▶ The voter shall be prevented from
 - ▶ choosing more than one candidate / answer
 - ▶ This is different from "Vote only once" (Uniqueness, below)

Under-voting

- ▶ The voter may receive a warning of not voting
 - ▶ but the system must not prevent undervoting



Data Integrity

- ▶ Once recorded, each vote cannot be tampered with in any manner
 - ▶ i.e votes should not be modified, forged or deleted without detection
- ▶ Votes should not be able to be modified
 - ▶ without detection
 - ▶ Any detected modification should be corrected to the original value

System Integrity

- ▶ The system cannot be re-configured during operation



Fairness

- ▶ no early results can be obtained
 - ▶ which could influence the remaining voters



Verifiability, Accountability

Individual verifiability

- ▶ Every voter can verify that her vote was counted correctly

Accountability

- ▶ Ensure that system operations are logged
- ▶ Election records must be reliable and demonstrably authentic

Universal Verifiability (Auditability)

- ▶ The system (and in particular, the logs) must allow open audit
 - ▶ to verify that
 - ▶ only valid votes were counted
 - ▶ and the published outcome really is the sum of all the votes



"4-Eyes" Principle

Distribution of Authority

- ▶ The administrative authority shall not rest with a single entity
- ▶ The authority shall be distributed among multiple administrators
 - ▶ who are known not to collude among themselves
 - ▶ e.g different political parties



Voter Anonymity

Ensure that votes must not be associated with voter identity

- ▶ No one should be able to determine how any individual voted



Coercion-resistance, Receipt-freeness

Attackers are prevented from monitoring the voters' systems

- ▶ even if the voters themselves want to allow such monitoring
- ▶ to prevent cases of wide scale coercion or vote selling

No Receipts

- ▶ Voters should not obtain a receipt or other information
 - ▶ that would enable them to prove to somebody how they voted
 - ▶ this would facilitate vote selling or coercion



System Disclosability

System Disclosability

Open-Source

- ▶ The core of the system shall be open-source
 - ▶ Software used should be open to public inspection and auditing



Simplicity, Certification, Mainainability,

Simplicity

- ▶ The system shall be designed to be as simple as possible
 - ▶ complexity is an enemy of security

Testing and Certification

- ▶ The system should be tested by experts
 - ▶ in particular for security considerations

Mainainability

- ▶ The system must be easy to maintain and manage



Robustness, Security

Robustness

- ▶ Systems should work robustly, even in the face of numerous failures
- ▶ Election systems should work robustly
 - ▶ without loss of any votes
 - ▶ even in the face of numerous failures
 - ▶ including failures of voting machines
 - ▶ and total loss of network communication

Secure

- ▶ The system shall be developed in a manner that
 - ▶ ensures there is no malicious code or bugs



Other Non-Functional Reqs

Flexibility

- ▶ Equipment should support a variety of ballot formats

Certiability

- ▶ Systems should be testable against essential criteria

Transparency

- ▶ Voters should be able to possess a
 - ▶ general understanding of the whole process

Cost-effectiveness

- ▶ Systems should be affordable and efficient





In short: Secrecy

Secrecy

It is infeasible to find out which voter has submitted which vote

Either

- ▶ the votes are never seen in clear, or
- ▶ it is unknown which vote belongs to which voter

Secrecy should also be satisfied for

- ▶ partial information on votes, and for
- ▶ relation between votes of several voters



In short: Anonymity

Anonymity

- ▶ It is infeasible to find out
 - ▶ whether or not a particular voter
 - ▶ has participated the vote
- ▶ This requirement can hardly be achieved by electronic voting schemes
 - ▶ unless some physical or organizational assumptions are



In short: Eligibility, No Double-Voting

Eligibility

- ▶ Only entitled voters are able to submit a vote
 - ▶ Or: the votes of unauthorized voters are not counted

No Double-Voting

- ▶ Every entitled voter can cast only one single vote



In short: Validity, Correctness

Validity

- ▶ Only valid votes are counted

Correctness

- ▶ The tally that pops up at the end of the vote
 - ▶ is the correct sum of all valid votes



In short: Verifiability

Local Verifiability

- ▶ Every voter can verify whether his vote is -is included correctly in the published tally

Global Verifiability

- ▶ Anyone can verify that all valid votes have been counted
 - ▶ and that the published tally is correct



In short: Receipt-Freeness

Receipt-Freeness

A secure voting scheme should disable the voters from selling their vote


- ▶ It is inevitable that a voter can accept money for promising his vote
 - ▶ but the voting scheme should prevent that the voter receives money
 - ▶ only in case he keeps his promise
- ▶ the voting scheme should not give the voter
 - ▶ any means to prove which particular vote he has cast

A Receipt-Free Voting Protocols

is (by definition) one that does not allow the voter

- ▶ to prove the cast vote

Routing

- ▶ Assume a user
- ▶ queries a database or a web service
 - ▶ Then "the system"
 - ▶ which is composed of many routers and tables
 - ▶ needs to figure out
 - ▶ how to route back the answer of the query to the caller
- ▶ In most plausible implementations, the "system"
 - ▶ "knows" your IP-(or MAC-) address to answer correctly
- ▶  Give at least two different implementation of "anonymous routing"

Routing

- ▶ One of the main approaches for
- ▶ implementing anonymity is to
 - ▶ "encrypt and mix" (or securely obfuscate) the information
 - ▶ say: the IP-address
 - ▶ and distribute it in small pieces
 - ▶ in a way that the parties in the protocol do not know
 - ▶ even if they collude
 - ▶ which node (which node IP-address)
 - ▶ is querying which server


e-petitions

- ▶ Assume a set of authorized users
- ▶ say, EU nationals above a certain age
 - ▶ are allowed to "sign" a petition asking for
 - ▶ Legalization of marihuana, or
 - ▶ Prohibition of the release of genetically manipulated organisms in agriculture, or
 - ▶ Permit the euthanasia in certain cases
- ▶ A citizen would like to vote
 - ▶ in favour or in against a petition
 - ▶ but he hesitates because
 - ▶ he does not want to risk that
 - ▶ his employer (or his parents, etc) will know
 - ▶ his private position in this sensitive issue

e-Toll (on highways)

- ▶ A user wants to use a system of highways
 - ▶ over a period of time
 - ▶ say, a month
 - ▶ and would like to get a bill for his usage
 - ▶ but **nobody** should know which trips he has done
- ▶ The problem here is that
 - ▶ the car should be somehow "identified"
 - ▶ the bill should be sent to the correct address

e-money

- ▶ A user engages in eCommerce
 - ▶ to buy or sell physical or virtual goods
- ▶ but all parties are required
 - ▶ each one to obtain only the **bare minimum information**
 - ▶ they need
 - ▶ for the payment and the delivery of the goods
- ▶  Sketch a "solution" that
 - ▶ uses a perfectly trusted, trustworthy, available T3P
 - ▶ and assume each party has a secure channel to it

Attribute-based access control

A system would like to

- ▶ enforce access control rules
 - ▶ of the type
 - ▶ "EU citizens over 21 years are entitled to access the service"
 - ▶ but user does not want the user to be traceable or
 - ▶ that that more information is leaked than the bare minimum
- ▶ Exercise: again: sketch a "T3P Ideal Solution"

Anonymity via a T3P: the ideal solution



The "T3P Ideal Solution" (or "Ideal Solution")

- ▶ Is a trivial way of constructing a solution for
 - ▶ any of those applications
 - ▶ preserving anonymity exactly as required
 - ▶ as follows:
- ▶ Assume there is an ideal trusted third party (T3P) that
 - ▶ would never be at risk of being corrupted
 - ▶ is perfectly trustworthy
 - ▶ it acts exactly as specified
 - ▶ does nothing more, nothing less
 - ▶ and all parties indeed agree
 - ▶ to trust it
 - ▶ are willing to work with it



Anonymity via a T3P: the ideal solution

- ▶ Further assume the parties have
 - ▶ unrestricted access to the T3P
 - ▶ whenever they need it
 - ▶ and they can communicate with the T3P
 - ▶ over **secure channels**
 - ▶ which provide timeliness, integrity and confidentiality



Anonymity via a T3P: the ideally secure solution

- ▶ Under those conditions it is not difficult
 - ▶ to implement any of the mentioned applications
 - ▶ each party P_i sends his input x_i to the T3P
 - ▶ the T3P calculates the results that each party really needs
 - ▶ and sends back to each of the parties exactly
 - ▶ this required information, not more



Anonymity via a T3P: the ideally secure solution

- ▶ A recurrent theme in PETs design
 - ▶ is how to implement efficiently systems
 - ▶ that can be "simulated" (in some formal sense)
 - ▶ by the ideal solution
 - ▶ but that do not rely on a trusted third party
 - ▶ but rather on cryptography-based PET building blocks

Anonymity in Communication: Problem

- ▶ It is easy to trace
 - ▶ user's actions and interactions with the world wide web
- ▶ Solutions are based on the idea
 - ▶ to have all users on of the network
 - ▶ blend into a *crowd*, so that
 - ▶ it is difficult to ascertain who is talking to whom

Anonymity Goals: Properties

- ▶ As a property on communication systems
 - ▶ there are 3 types of anonymity properties
 - ▶ that can be provided
- ▶ Sender anonymity
- ▶ Receiver anonymity
- ▶ Unlinkability of sender and receiver

Anonymity Goals: Properties

- ▶ Sender anonymity
 - ▶ means that the identity of the party
 - ▶ who sent a message is hidden,
 - ▶ while its receiver (and the message itself) might not be
- ▶ Receiver anonymity
 - ▶ similarly means that the identity of the receiver is hidden
- ▶ Unlinkability of sender and receiver
 - ▶ means that the sender and receiver
 - ▶ cannot be identified as communicating with each other
 - ▶ . . . although each can be identified as participating
 - ▶ in some communication

Anonymity Goals: Attackers

- ▶ A second aspect of anonymous communication
 - ▶ is: against which type of attackers
 - ▶ should we protect ourselves
- ▶ The attacker might be of different types (next slide)

Anonymity Goals: Attackers

A honest but curious participant

- ▶ That is, he tries to gain information from data that he gets
 - ▶ without actively or passively leaving his role

An eavesdropper (passive attacker)

- ▶ That is, he observes
 - ▶ messages sent and received over the network
 - ▶ also those not meant for him
 - ▶ he can not subvert the crypto nor use keys that he doesn't know
 - ▶ he can not create messages, nor change the messages on the fly

Anonymity Goals: Attackers

An active attacker

- ▶ He has, beyond the capabilities of the eavesdropper
 - ▶ also create new messages, manipulate, redirect or suppress messages

A collaboration of participants and network attackers

- ▶ consisting of some senders, receivers, and other parties
 - ▶ or variations of these

How Crowd Works

► Overview of Crowds

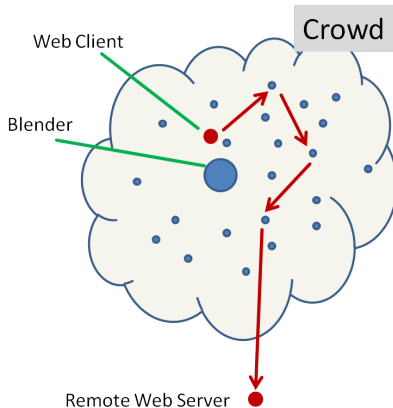


Figure: The blender application administrates user membership

How Crowd Works

Every member of the crowd runs a web proxy

- ▶ Called jondo (pronounced "John Doe")
 - ▶ to indicate "a faceless participant in the crowd"
- ▶ The user starts the jondo on the user's computer
 - ▶ the jondo contacts a server called the blender
 - ▶ to request admittance to the crowd
- ▶ The blender reports
 - ▶ to this jondo the list of current crowd members
 - ▶ plus other information to participate in the crowd

How Crowd Works

The jondo is the web proxy for all protocols

- ▶ Any request from the browser
 - ▶ is sent inside the users PC directly to the jondo

The first time the jondo receives a user request from the browser

The jondo initiates the

- ▶ **Establishment of a random path of jondos**
 - ▶ that carries the users transactions to and from their intended web servers

Path Generation

Path Identity

- ▶ A path is identified by each jondo (in this path)
 - ▶ using a *local path id*
- ▶ When a jondo receives a request marked with path id
 - ▶ from its predecessor in a path
 - ▶ it replaces path id with a different path identifier, that he stores in translate-table

Path Generation

The first jondo in the path

- ▶ chooses randomly a jondo from the crowd
 - ▶ forwards the request to it

When the second (or third, etc) jondo receives the request

- ▶ either
 1. forwards the request to a randomly chosen peer
 2. with a low probability,
 - ▶ submits the request to the end server
- ▶ Random walk, biased in favor of forwarding

How Crowd Works

Subsequent requests

- ▶ Initiated at the same jondo follow the same path
- ▶ Once established, the path remains static (for a session/ for a day/ ...)
 - ▶ the server replies along the same path as the requests
 - ▶ in reverse

Each jondo cannot tell whether its predecessor

- ▶ Initiated the request or
 - ▶ just forwards it from another jondo

How Crowd Works

Paths can be established

- ▶ per request or
- ▶ during a global join commit, say, once per day

It is difficult to ascertain

- ▶ which member of a crowd is
 - ▶ actually making web requests:
- ▶ no single party knows the entire path
 - ▶ that any member uses to establish external connections

Anonymity Goals: Degrees of Anonymity

- ▶ The degree of anonymity
 - ▶ can be viewed informally as a continuum
- ▶ For simplicity,
 - ▶ we describe this continuum
 - ▶ with respect to sender anonymity,
 - ▶ but it can naturally be extended to
 - ▶ receiver anonymity and unlinkability

Degrees of Anonymity

Absolute Privacy

- ▶ The attacker cannot perceive
 - ▶ the presence of communication
- ▶ The attacker/observer can not distinguish
 - ▶ situations in which somebody sent communication
 - ▶ from those in which nobody did
- ▶ That is, sending a message results
 - ▶ in no observable effects for the attacker

Degrees of Anonymity

Beyond Suspicion

- ▶ Though the attacker can see evidence that
 - ▶ a message was sent by somebody
- ▶ but the sender appears
 - ▶ no more likely to be the originator of the message
 - ▶ than any other potential sender in the system

Degrees of Anonymity

Probable Innocence

- ▶ the attacker has reasons to believe
 - ▶ the sender is more likely to be responsible than
 - ▶ other potential sender
- ▶ but from the attacker's point of view
 - ▶ the sender appears more likely
 - ▶ to be **not** the originator than to be the originator

Possible Innocence

- ▶ There is a nontrivial probability that the real sender is someone else

Degrees of Anonymity

Exposed

- ▶ The attacker is aware of who the sender is

Provably Exposed

- ▶ the attacker can
 - ▶ 1. identify (by some identifier) the sender of a message, and
 - ▶ 2. prove to others that this is the identity of the sender

Types of Attackers

- ▶ Local Eavesdropper
 - ▶ An attacker who can observe the communication to and from a user's machine
- ▶ Collaborating Crowd Members
 - ▶ members of the crowd network who
 - ▶ pool their information and
 - ▶ may deviate from the prescribed protocol
- ▶ End Server
 - ▶ the web server to which the web transaction is directed

Sender vs. Receiver Anonymity

- ▶ Sender Anonymity
 - ▶ The identity of the party who sent a message is hidden
- ▶ Receiver Anonymity
 - ▶ The identity of the receiver of a message is hidden
- ▶ Unlinkability of Sender & Receiver
 - ▶ Though sender and receiver can be identified as communicating
 - ▶ they cannot be identified as communicating with each other

Tossing a Coin over Phone: Sketch

- ▶ **Commit Phase** Alice chooses $b_A \leftarrow \{0, 1\}$
 - ▶ "puts b_A inside a blob", secured by a key k
 - ▶ $(b_?, k) \leftarrow \text{commit}(b_A)$
 - ▶ Alice sends $b_?$ to Bob
 - ▶ but Bob is unable to extract information about b_A
- ▶ **Response Phase**
 - ▶ Bob responds with his choice b_B
 - ▶ Bob wins if $b_B = b_A$
- ▶ Alice now knows who won,
 - ▶ and Alice may now say "Sorry, Bob, you lost!"
 - ▶ But Bob may not be sure . . .
 - ▶ Why should he trust her?
- ▶ We need some kind of proof: the
 - ▶ **Opening** or **Verification Phase**
 - ▶ later

Tossing a Coin over Phone: Commit

- ▶ Alice chooses a random $b_A \leftarrow \{0, 1\}$
- ▶ and runs a **non-deterministic** "commit algorithm" which produces:
 - ▶ a "blob" $b_?$, and
 - ▶ a key k to "open" the blob:
 - ▶ $(b_?, k) \leftarrow \text{commit}(b_A)$
- ▶ The commit algorithm often
 - ▶ first chooses a random key in a keyspace
 - ▶ $k \leftarrow \mathcal{K}$
 - ▶ and then uses the key to construct $b_?$
- ▶ Alice
 - ▶ keeps k secret
 - ▶ as long as the value of b_A is "hidden"
 - ▶ and sends $b_?$ to Bob
- ▶ In this way
 - ▶ she has committed herself to the bit
 - ▶ $b_A = 0$ or $b_A = 1$:
 - ▶ she will not be able to modify her choice later

Tossing a Coin over Phone: Commit

- ▶ It is important that Alice chooses her bit *randomly*
 - ▶ If it is not, then we must impose
 - ▶ rather strong and awkward assumptions
- ▶ To see this, assume Alice did not choose his bit randomly,
 - ▶ but according to an algorithm that she "has"
 - ▶ How can you guarantee that Bob – somehow –
 - ▶ happens to use the same algorithm?
- ▶ We want that the commit protocol, from the point of view of Alice
 - ▶ is secure, that is, no matter what algorithm B uses
 - ▶ B should have a $.5 + \epsilon$ chance of winning
 - ▶ where ϵ is negligible

Tossing a Coin over Phone: Commit

- ▶ Bob sees $b_?$
 - ▶ but from this
 - ▶ he can not infer the value of b_A and moreover
 - ▶ $b_?$ has no "visible information" about b_A
- ▶ Nevertheless $b_?$ is a commitment on b_A
 - ▶ $b_?$ "hides" b_A and may only be opened
 - ▶ to this value

Tossing a Coin over Phone: Response

- ▶ Bob now chooses a bit
 - ▶ $b_B = 0$ or $b_B = 1$ and
 - ▶ sends b_B to Alice
- ▶ Now Alice sends to Bob
 - ▶ her bit b_A
 - ▶ ...but to convince Bob
 - ▶ Alice must now **prove** that indeed her choice was b_A
 - ▶ for this Alice presents also the secret k
 - ▶ which can be used by Bob to "open the blob" $b_?$
 - ▶ or, in other words, to "verify" the bit of Alice

Tossing a Coin over Phone: Verification

- ▶ Bob runs an deterministic "open" or "verify" protocol
 - ▶ with input $(b_?, b_A, k)$
 - ▶ and output Boolean = $\{T, F\}$
- ▶ If $\text{verif}(b_?, b_A, k)$ produces T
 - ▶ then Bob is sure
 - ▶ that Alice's choice is indeed b_A
- ▶ For this **verif** must satisfy the condition:
 - ▶ $(V) : \text{verif}(b_?, b_A, k) = \text{verif}(b_?, b'_A, k') = T \Rightarrow b_A = b'_A$



Example of Commit / Verify

- ▶ Let us divide the odd numbers in two types:
 - ▶ T+1: those of the form $(4n + 1) = \{1, 5, 9, \dots\}$
 - ▶ T-1: those of the form $(4n - 1) = \{3, 7, 11, \dots\}$
- ▶ Note that the product of two numbers of type T+1 is also T+1
 - ▶ and the product of two T-2 numbers is also T+1
- ▶ Alice chooses randomly two primes: p, q
 - ▶ of a certain large size (= length, when written as decimal or binary)
 - ▶ both of type T+1, or both T-1
 - ▶ depending on his chosen bit $b_A \leftarrow \{0, 1\}$
- ▶ and sends $b? = p \cdot q$ to Bob

Example of Commit / Verify

- ▶ Since Bob can not factorize $b_? = pq$ to check the type
 - ▶ of the factors ($\text{type}(p)=\text{type}(q)$)
 - ▶ and there is no known efficient algorithm
 - ▶ to decide this type
 - ▶ we say that $b_?$ has no "visible information" about the type
 - ▶ that is, about b_A

Example of Commit / Verify

- ▶ In our example: $k = (p, q)$, the factors of $b_?$
- ▶  Explain how the algorithm
 - ▶ $\text{verif}(b_?, b'_A, k')$
 - ▶ works
- ▶  Show here that
 - ▶ (V) : $\text{verif}(b_?, b_A, k) = \text{verif}(b_?, b'_A, k') = T \Rightarrow b_A = b'_A$

Commitment Scheme

Is a pair of algorithms (commit, verify):

$(m?, k_v) \leftarrow \text{commit}(m)$, which

- ▶ given a message $m \in \{0, 1\}^n$
- ▶ outputs a commitment (or "blob") $m?$
 - ▶ and a (verification) key $k_v \in \{0, 1\}^\ell$

$\text{verify}(m?, m, k_v)$ which

- ▶ takes
 - ▶ a commitment $m?$,
 - ▶ a (candidate) message m , and
 - ▶ a candidate key k_v
- ▶ outputs T or F

With the following two properties: (next slide)

Commitment Scheme Properties

Hiding

The information $m_?$ gives an adversary no significant advantage for

- ▶ guessing correctly m

Binding

If $(m_?, k_v) \leftarrow \text{commit}(m)$

- ▶ then $\text{verif}(m_?, m, k_v) = \text{True}$

It is unfeasible to find $m' \neq m$ and $k'_v \in \{0, 1\}^\ell$ such that

- ▶ $\text{verif}(m_?, m', k'_v) = \text{T}$

Thus, $m_?$ can only be opened as the original m

It is unfeasible to open it as a different m'

- ▶ But it is possible that there exists (in a mathematical sense):
 - ▶ $m' \neq m, k'_v$ with $\text{verif}(m_?, m', k'_v) = \text{T}$
 - ▶ but it is unfeasible to find them

Notes

- ▶ In some cases the 'commit' operation is implemented as a sequence:
 - ▶ first generate a key k_c (for commitment)
 - ▶ which is associated somehow to a verification key k_v
 - ▶ think of (k_c, k_v) as a private/public key pair
 - ▶ or that they are simply equal
 - ▶ then use the key k_c to generate the commit:
 - ▶ $m_? \leftarrow \text{commit1}(m, k_c)$
- ▶ We do not assume that the messages m are bits

Perfectly Binding Commit Schemes

It is also reasonable that commit protocols exist for which

- ▶ it is really impossible (!) to find:
 - ▶ $m' \neq m, k'$ with $\text{verif}(m, m', k') = T$
 - ▶ 'impossible' means: there are no such $m' \neq m, k'$

Then the Commitment Scheme is called Perfectly Binding

- ▶ in that case, we do not need a separate verification procedure
 - ▶ (see next slides)

Def: A Perfectly Binding Commitment Scheme

Is a polynomial time algorithm 'Com' that takes

- ▶ a message $m \in \{0, 1\}^n$ and
- ▶ a key $k \leftarrow \{0, 1\}^\ell$
 - ▶ ℓ , the length of the key may depend (polynomially)
 - ▶ on n , the length of m

and outputs a commitment $m_?$

- ▶ with the following properties (next slide)

Def: A Perfectly Binding Commitment Scheme (cont)

Hiding:

Any ppt algorithm A distinguishes the distributions

- ▶ $\mathcal{D}\{Com(m_0, r) \mid k \leftarrow \{0, 1\}^\ell\}, \mathcal{D}\{Com(m_1, r) \mid k \leftarrow \{0, 1\}^\ell\}$
 - ▶ with a negligible probability

Binding:

For all $m_0, m_1 \in \{0, 1\}^n, k_0, k_1 \in \{0, 1\}^\ell$, if $m_0 \neq m_1$

- ▶ then $Com(m_0, k_0) \neq Com(m_1, k_1)$

Symmetric Ciphers


- ▶ A cipher over $(\mathcal{M}, \mathcal{C})$ is a pair of efficient algorithms $(\mathcal{E}, \mathcal{D})$
 - ▶ "efficient": polynomial time

$$\mathcal{E} : \{\mathcal{K} \times \mathcal{M}\} \rightarrow \mathcal{C}$$

$$\mathcal{D} : \{\mathcal{K} \times \mathcal{C}\} \rightarrow \mathcal{M}$$

- ▶ With the following consistency condition:

$$\forall k \in \mathcal{K} \forall m \in \mathcal{M} \mathcal{D}(k, \mathcal{E}(k, m)) = m$$

- ▶ \mathcal{E} is often *randomized* but \mathcal{D} is always deterministic
- ▶  Explain why (or when) \mathcal{E} should be *randomized*
 - ▶ Hint: discuss the perils of
 - ▶ using the same key on the same message twice
 - ▶ on an example

Integrity

- ▶ Suppose Alice sends Bob a message m
 - ▶ Bob wants to ensure
 - ▶ m was truly sent by Alice
 - ▶ m was not modified by anybody
- ▶ To accomplish this, Alice sends
 - ▶ besides the message m
 - ▶ or its encryption
 - ▶ $c = \mathcal{E}(k, m)$ or $c = \mathcal{E}(P_B, m)$
 - ▶ a tag t which
 - ▶ identifies the sender Alice *and*
 - ▶ shows that the message was not modified (*message integrity*)

Integrity

- ▶ The resulting message
 - ▶ containing
 - ▶ the original m or c
 - ▶ plus the tag t
 - ▶ is denoted by $S_{Alice}(m)$ (S = Signed)
- ▶ Alice creates the tag
 - ▶ using a function $Tag(m, k) = t$
- ▶ Bob verify the tag using a function
 - ▶ $Verif(m, k, t) \in \{0, 1\}$
 - ▶ 1 means "true", etc
- ▶ In the case of a secret shared key:
 - ▶ $t = T(m, k)$
- ▶ In the case of a private / public keys:
 - ▶ $t = T(m, p_A)$


Integrity

- ▶ If Eve intercepts c, t
 - ▶ when passing through the insecure connection,
 - ▶ she will not be able to modify c
 - ▶ undetected without knowing k
- ▶ The standard definition of security for integrity:
 - ▶ Threat model: "Adaptive chosen-message attack"
 - ▶ the attacker can induce the sender
 - ▶ to authenticate messages of the attacker's choice
- ▶ The security goal is usually called
 - ▶ "Existential unforgeability":
 - ▶ Attacker should be unable to
 - ▶ forge a valid tag on any message
 - ▶ not authenticated by the sender
- ▶ Exercise: Express this security notion as a game

Signatures: Signed Messages

- ▶ In the case of public-key cryptosystems
 - ▶ integrity can be provided by a process called *signing* a message
- ▶ Suppose Alice has the key pair (P_A, p_A)
 - ▶ Alice creates the signed message $S_A(m) = (m, D(p_A, h(m)))$
- ▶ Using the decryption algorithm (rather than encryption)
 - ▶ of the hash of m
 - ▶ using her **private key**

Signatures: Non-Repudiation

- ▶ Signatures have some properties
 - ▶ that symmetric integrity tags do not have:
 - ▶ 1. They can be produced (correctly) only by one entity
 - ▶ 2. They can be verified by everybody
 - ▶ 3. They proof they provide is "transferable"
 - ▶ 4. They provide non-repudiation
 - ▶ (The properties are related between them)
- ▶  Explain each property



Blind signatures

- ▶ Blind signatures are employed
 - ▶ mostly in privacy-related protocols
 - ▶ or in eCash protocols
- ▶ Here, the signer and message author are
 - ▶ different parties
 - ▶ and anonymity of the author is required



Blind signatures: Physical analogy:

- ▶ A voter fills-in a ballot
 - ▶ and encloses the **completed anonymous ballot**
 - ▶ in a special envelope with carbon paper
- ▶ that has the voter's credentials written **on the outside**
- ▶ The voter gives the closed envelope to a voting official who
 - ▶ 1. verifies
 - ▶ the credentials
 - ▶ that the voter is authorized and
 - ▶ that the voter hasn't voted yet
 - ▶ 2. signs it and returns it




Blind signatures: Physical analogy:

- ▶ The ballot can be signed
 - ▶ through the envelope by the carbon paper
 - ▶ and the envelope can only contain one ballot
- ▶ The voter now retrieves
 - ▶ the signed ballot from the envelope
 - ▶ and transfers it to a new unmarked normal envelope
 - ▶ the signature has been "un-blinded":
 - ▶ the signature is valid for the un-blinded message
- ▶ Note that the signer does not see the message content
 - ▶ but a third party can later verify the signature



Blind signatures

- ▶ Blind signatures provide *unlinkability*
 - ▶ which prevents the signer (or anybody)
 - ▶ from linking the envelope signed
 - ▶ to an un-blinded version
- ▶ Another example where
 - ▶ blind signatures are used is digital cash
- ▶  Explain how this might work using blind signatures!



Blind RSA signature

Recall "textbook RSA":

- ▶ Encryption/Decryption $m =$ message, $c =$ cypher text
 - ▶ $c = \mathcal{E}(m) \equiv_n m^e$
 - ▶ $m = \mathcal{D}(c) \equiv_n c^d = c^{1/e}$
- ▶ Message signing / Verification $m =$ message, $\sigma =$ signature
 - ▶ $\sigma = \mathcal{D}(m) \equiv_n m^d = m^{1/e}$
 - ▶ $m = \text{Verif}(\sigma) \equiv_n \sigma^e$



Blind RSA signature

- ▶ The "blind signature" is a two party protocol:
 - ▶ 1. Alice creates a "blinding factor" r , randomly
 - ▶ sends to Bob $(r^e m) \bmod n$
 - ▶ 2. Bob computes $(r^e m)^{1/e} = r \cdot m^{1/e} \bmod n$
 - ▶ 3. Alice divides by the blinding factor r ,
 - ▶ obtaining $m^{1/e} \bmod n$, the original message, but signed by Bob
- ▶ For general use, this method has two problems:
 - ▶ Bob does not know what he is signing
 - ▶ Alice can get Bob to sign anything,

Private communication channels and off-the-record

- ▶ The purpose is to keep conversations confidential
 - ▶ like a private conversation in real life
 - ▶ or **off the record**, for example
 - ▶ to protect the sources in journalism
- ▶ Cryptography tools
 - ▶ usually produce "transferable proofs"
 - ▶ or "verifiable transcripts"
 - ▶ which can be later used
 - ▶ to reconstruct the communication events
 - ▶ and the identities of the participants
- ▶ Off-the-Record Messaging (OTR)
 - ▶ is a cryptographic protocol that can be used in
 - ▶ interactive conversations and has
 - ▶ the following properties (next slide)

Private communication channels and off-the-record

Authentication Within the conversation,

- ▶ the recipient can be sure that
- ▶ a message is coming from the *claimed* person

Deniable authentication After the conversation

- ▶ anyone is able to forge a message
 - ▶ to appear to have come from one of the participants
 - ▶ in the conversation

Forward secrecy Messages are only encrypted

- ▶ with temporary per-message symmetric keys
- ▶ negotiated using a key agreement, like Diffie-Hellman
 - ▶ The compromise of long-lived cryptographic keys
 - ▶ does not compromise any previous conversations
- ▶ even if an attacker is in possession of ciphertexts



Undeniable signatures

- ▶ Undeniable signature is a digital signature scheme
 - ▶ Where a signer can publish a signature of a message
 - ▶ ...but the signature reveals nothing to a recipient
 - ▶ without taking part
 - ▶ in either of two interactive protocols
- ▶ Confirmation protocol
 - ▶ which confirms that a candidate
 - ▶ is a valid signature of the message
 - ▶ issued by the signer, identified by the public key
- ▶ Disavowal protocol
 - ▶ which confirms that a candidate
 - ▶ is not a valid signature of the message
 - ▶ issued by the signer
- ▶ The result of each protocol is non-transferable



Undeniable signatures

- ▶ Motivation:
 - ▶ allow the signer to determine
 - ▶ to whom he verifies or disavows a signature
 - ▶ this is the purpose of the interactive nature
 - ▶ of the protocols



Undeniable signatures: Example

- ▶ A customer
 - ▶ wishes to gain access
 - ▶ to a safety-deposit box room in a bank
- ▶ The bank
 - ▶ requires the customer
 - ▶ to sign a time and date document
 - ▶ before access is granted
- ▶ But the customer
 - ▶ does not want the bank
 - ▶ to be able to tell anyone
 - ▶ when he has actually used those facilities
- ▶ Therefore
 - ▶ he uses an undeniable signature
 - ▶ in which verification is impossible
 - ▶ without his direct involvement



Undeniable signatures: Example

- ▶ In such an undeniable signature protocol
 - ▶ it is possible that a document is signed
 - ▶ and that the signer later
 - ▶ denies that she has signed that document
- ▶ This must be avoided
 - ▶ and this explains the term "*undeniable signature*"
- ▶ The participants of an undeniable signature protocol
 - ▶ are the signer A and the verifier B



Undeniable Signatures: Protocol Steps, high-level view

Key generation:

- ▶ A generates a secret signing key p and the corresponding public verification key \mathcal{P}

Signature generation:

- ▶ $\sigma = \sigma(m, p)$
 - ▶ where m is the document and
 - ▶ p is the secret key of A



Undeniable Signatures: Protocol Steps, high-level view



Signature verification

- ▶ Or "Signature Confirmation"
 - ▶ to interactively verify the signautes
- ▶ We assume that B knows \mathcal{P} , the public key of A
 - ▶ 1. $A \rightarrow B : m, \sigma$ message and signature
 - ▶ 2. $B \rightarrow A : \gamma$ the challenge
 - ▶ 3. $A \rightarrow B : \varrho = \varrho(\gamma, m, p)$ the response to the challenge
- ▶ B verifies:
 - ▶ $Verif(m, \sigma, \mathcal{P}, \gamma, \varrho)$
 - ▶ outputs "signature verified/confirmed" or "not verified"
- ▶ It is impossible to verify a signature
 - ▶ without the signer's participation
 - ▶ A third party is unable to verify that the signature is genuine
 - ▶ The proof provides no transferrable proof script

Undeniable Signatures: Protocol Steps, high-level view



Signature Disavowal

- ▶ But the signer A might deny that she has generated the signature:
 - ▶ A pretends to perform the verification protocol
 - ▶ but on purpose she does it incorrectly
 - ▶ in order to claim that the signature is not hers
- ▶ This is why another subprotocol is useful
 - ▶ to securely disavow
- ▶ If A disavows a signature
 - ▶ then B and A perform a disavowal subprotocol
 - ▶ (very much as the one above, for verification)
 - ▶ allowing a verifier to confirm that
 - ▶ the signature is not valid



Designated verifier signature

- ▶ Suppose two users want to exchange messages
 - ▶ in a conversation or asynchronously, per mail or USB
- ▶ They want to authenticate their messages
 - ▶ but the signatures can only be
 - ▶ verified by a single designated verifier
 - ▶ who is chosen by the signer
- ▶ Unlike in undeniable signature scheme
 - ▶ this protocol of verifying is **non-interactive**:
- ▶ The signer must
 - ▶ choose the **set of possible verifiers in advance**
 - ▶ but the signer does not participate in the verification protocol
 - ▶ (it is not an interactive proof)



Group signature

- ▶ Given a group of users
 - ▶ the members, acting as signers
 - ▶ wish to preserve their anonymity against verifiers
- ▶ But verifiers want to be sure that valid signatures
 - ▶ were produced only by existing group members
 - ▶ who can be identified, if necessary, by the group manager



Group Signatures

- ▶ Conventional Digital signatures
 - ▶ conflict with privacy, in particular with regard to
 - ▶ anonymity of signers and
 - ▶ unlinkability of issued signatures
 - ▶ Their unforgeability authenticates (and identifies) the signer as
 - ▶ the origin of the signed document
- ▶ Group signature scheme achieves authenticity and anonymity by
 - ▶ avoiding that information that would uniquely identify the signer
 - ▶ could leak during the signature verification procedure



Group signature Scheme

- ▶ Achieves authenticity and anonymity of signers
 - ▶ (against verifiers)
- ▶ Provides the ability of the group manager to identify the signer
- ▶ Verifiers are ensured that valid signatures
 - ▶ were produced by existing group members
 - ▶ who can be identified by the group manager



A group signature scheme *Sig*

- ▶ Is a tuple of algorithms $Sig = (Gen, Sig, Vf, Open, Judge)$
- ▶ $Gen(n)$: takes n , the number of users, produces
 - ▶ gpk : The group public key, for signature verification, public
 - ▶ $gmsk$: The opening key, provided to the group manager
 - ▶ $gsk[i]$: The private signing key of user i
- ▶ $Sig(gpk, gsk[i], m)$: any group member
 - ▶ using its signing key $gsk[i]$
 - ▶ can sign a message m to obtain a signature σ
- ▶ $Vf(gpk, m, \sigma)$: anyone
 - ▶ using the group public key gpk can verify a signature σ
- ▶ $Open(gpk, gmsk, m, \sigma)$: the group manager, using his master key
 - ▶ obtains a pair (i, τ)
 - ▶ i is the identity of the member who produced σ
 - ▶ τ is a proof of this claim that can be verified via the *Judge* algorithm