Advanced Privacy Topics

COM-402: Information Security and Privacy

(slide credits: Bryan A. Ford, Sandra Siby, Kirill Nikitin, Ceyhun Alp, Linus Gasser)

Outline

- **Secure Multi-Party Computation**
- Homomorphic Encryption
- Zero-Knowledge Proofs
- Private Information Retrieval

(Slide credits: Vitaly Shmatikov - Introduction to Secure Multi-Party Computation, Li Xiong - Secure Multiparty Computation, Jean-Pierre Hubaux EPFL)

Secure Multi-Party Computation (SMC)

- General framework for describing computation between parties who do not trust each other.
- Example: elections
  - N parties, each one has a “Yes” or “No” vote
  - Goal: determine whether the majority voted “Yes”, but no voter should learn how other people voted
- Example: auctions
  - Each bidder makes an offer
  - Offer should be committing! (can’t change it later)
  - Goal: determine whose offer won without revealing losing offers

More Examples

- Example: distributed data mining
  - Two companies want to compare their datasets without revealing them
  - For example, compute the intersection of two lists of names
- Example: database privacy
  - Evaluate a query on the database without revealing the query to the database owner
  - Evaluate a statistical query on the database without revealing the values of individual entries
  - Many variations

Observations

- In all cases, we are dealing with distributed multi-party protocols
  - A protocol describes how parties are supposed to exchange messages on the network.
- All of these tasks can be easily computed by a trusted third party
- *The goal of secure multi-party computation is to achieve the same result without involving a trusted third party.*

How to Define Security?

- Must be mathematically rigorous
- Must capture all realistic attacks that a malicious participant may try to stage
- Should be “abstract”
  - Based on the desired “functionality” of the protocol, not a specific protocol
  - Goal: define security for an entire class of protocols
- Defining security for general scenarios is hard!
- To tackle this, we consider the **Ideal vs Real World approach**

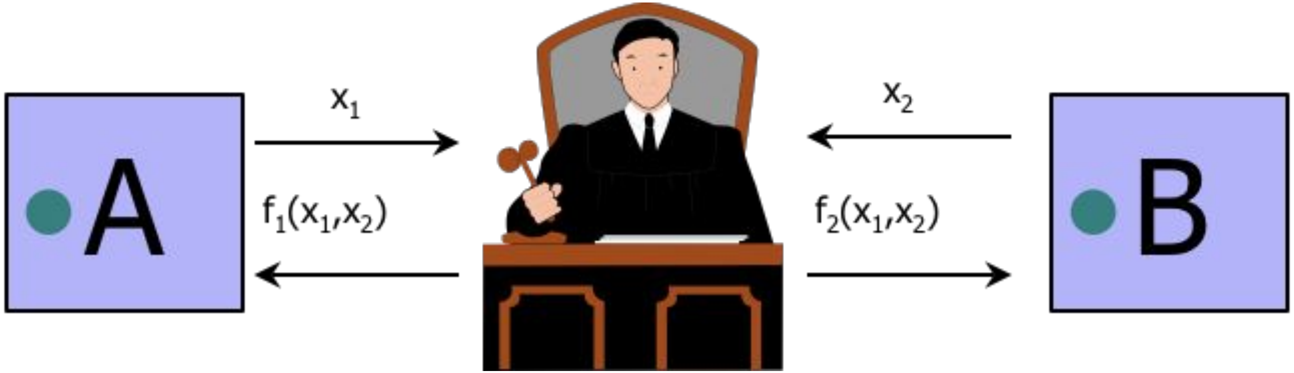
Ideal vs Real World

- Scenario: n mutually distrustful parties have inputs $(x_1, x_2, \dots, x_n)$. They wish to compute $y = f(x_1, x_2, \dots, x_n)$.
- **Ideal world:** Parties send inputs to an external trusted party that performs the computation for them.
- **Real world:** There is no trusted party. All parties run a protocol amongst themselves and learn y at the end.

Intuitively, we want the real world protocol to behave "as if" a trusted third party collected the parties' inputs and computed the desired functionality.

A secure real world scenario

- A protocol is secure if it **emulates** an ideal setting where the parties hand their inputs to a "trusted party," who locally computes the desired outputs and hands them back to the parties.
  [Goldreich-Micali-Wigderson 1987]
- A protocol is secure if any attack on a real protocol can be carried out in the ideal model.



Adversary Models

- Some of protocol participants may be corrupt
  - If all were honest, would not need secure multi-party computation
- Semi-honest (aka passive; honest-but-curious)
  - Follows protocol, but tries to learn more from received messages than she would learn in the ideal model
- Malicious
  - Deviates from the protocol in arbitrary ways, lies about her inputs, may quit at any point

Correctness and Privacy

- How do we argue that the real protocol “emulates” the ideal protocol?
- **Correctness**
  - All honest participants should receive the correct result of evaluating function f
  - Why? Because a trusted third party would compute f correctly
- **Privacy**
  - All corrupt participants should learn no more from the protocol than what they would learn in ideal model
  - What does corrupt participant learn in ideal model?
    - His/her input (obviously) and the result of evaluating f

Simulation

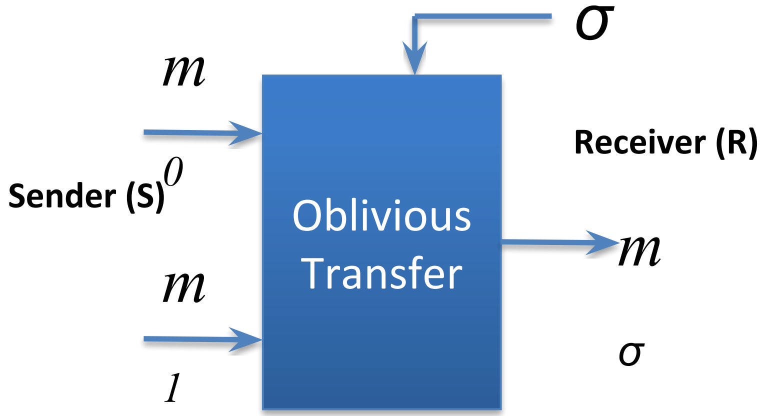
- Corrupt participant's **view** of the protocol = record of messages sent and received
    - In the ideal world, view consists simply of his input and the result of evaluating f
- How to argue that real protocol does not leak more useful information than ideal-world view?
- Key idea: **simulation**
    - If real-world view (i.e., messages received in the real protocol) can be simulated with access only to the ideal-world view, then real-world protocol is secure
    - Simulation must be **indistinguishable** from real view

SMC Building Blocks: Oblivious Transfer (OT)

- It was shown (by Kilian in 1988) that by using an implementation of oblivious transfer, and no other cryptographic primitive, it is possible to construct any secure computation protocol.

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 receives m_σ and learns nothing about $m_{(1-\sigma)}$



Oblivious Transfer - Example

- Receiver generates public keys P_σ and $P_{1-\sigma}$. It knows the decryption key for P_σ but not for $P_{1-\sigma}$. It sends the keys to the Sender.
- Sender encrypts message m_0 with P_0 and m_1 with P_1. It sends the results to the Receiver.
- Receiver can decrypt only m_σ.
  - If $\sigma = 0$, Receiver can decrypt m_0 since it has the decryption key for $P_{0.}$ It cannot decrypt m_1.
- Sender does not know anything about σ . It knows only the public keys.
- Receiver can decrypt only one message since it has only one decryptor key.
- Secure against semi-honest adversary.
- Generalization to 1-out-of-k OT
  - Generate k public keys. Decryption key is known for 1 key and unknown for k-1 keys.

Feasibility of SMC

m: number of parties
t: bound on the number of corrupted parties

- For t<m/3, SM protocols can be achieved for any function
- For t<m/2, SM protocols can be achieved for any function assuming that all parties have access to a broadcast (bcast) channel
- For t>=m/2, SM protocols without fairness or guaranteed output delivery can be achieved assuming that the parties have access to a bcast channel (Holds only in the computational setting)

Challenge: efficiency of the protocol, especially on large data sets

SMC implementations

Companies

- Sharemind by Cybernetica
  - Data analysis platform
  - Based on SMC concepts
- Unbound (formerly known as Dyadic)
  - Secure key sharing

Others

- SCAPI - Secure Computation API
- Fairplay and its second version FairplayMP

Outline

- Secure Multi-Party Computation
- **Homomorphic Encryption**
- Zero-Knowledge Proofs
- Private Information Retrieval

Slide credits: Vinod Vaikuntanathan (U. Toronto) and Shai Halevi (IBM T. J. Watson Research Center), Jean-Pierre Hubaux (EPFL)

Goal

- Wouldn't it be nice to be able to:
  - Keep my data in the cloud encrypted...
  - **While still allowing the cloud to search/sort/edit data on my behalf**
  - Encrypt my queries to the cloud...
  - **While still allowing the cloud to process them**
  - Cloud returns encrypted answers that I can decrypt

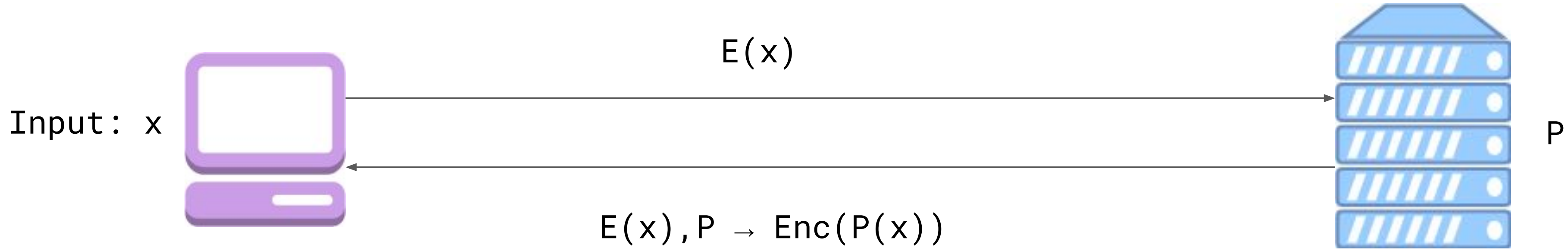
Delegate processing of data without giving away access to it

Example 1: Private Google Search

- **Client:** encrypt the query, send to Google
  - Google does not know the key, cannot “see” the query
- **Google:** encrypted query → encrypted results
  - You decrypt and recover the search results

Example 2: Private Cloud Computing

- **Client:** send encrypted input `E(x)`
- **Server:** execute program P on `E(x)`, return result



Some Notations

- An encryption scheme: $(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$
    - Plaintext space = $\{0,1\}$
    - $\mathtt{KeyGen}(\$) \rightarrow (\mathtt{pk}, \mathtt{sk})$
    - $\mathtt{Enc}_{\mathtt{pk}}(\mathtt{b}) \rightarrow \mathtt{c}$, $\mathtt{Dec}_{\mathtt{sk}}(\mathtt{c}) \rightarrow \mathtt{b}$
- Semantic security [GM'84]
    - $(\mathtt{pk}, \mathtt{Enc}_{\mathtt{pk}}(0)) \approx (\mathtt{pk}, \mathtt{Enc}_{\mathtt{pk}}(1))$
    - $\approx$ means indistinguishable by efficient algorithms

[GM'84] S. Goldwasser and S. Micali, Probabilistic encryption, Journal of Computer and System Sciences, 28:270-299, 1984

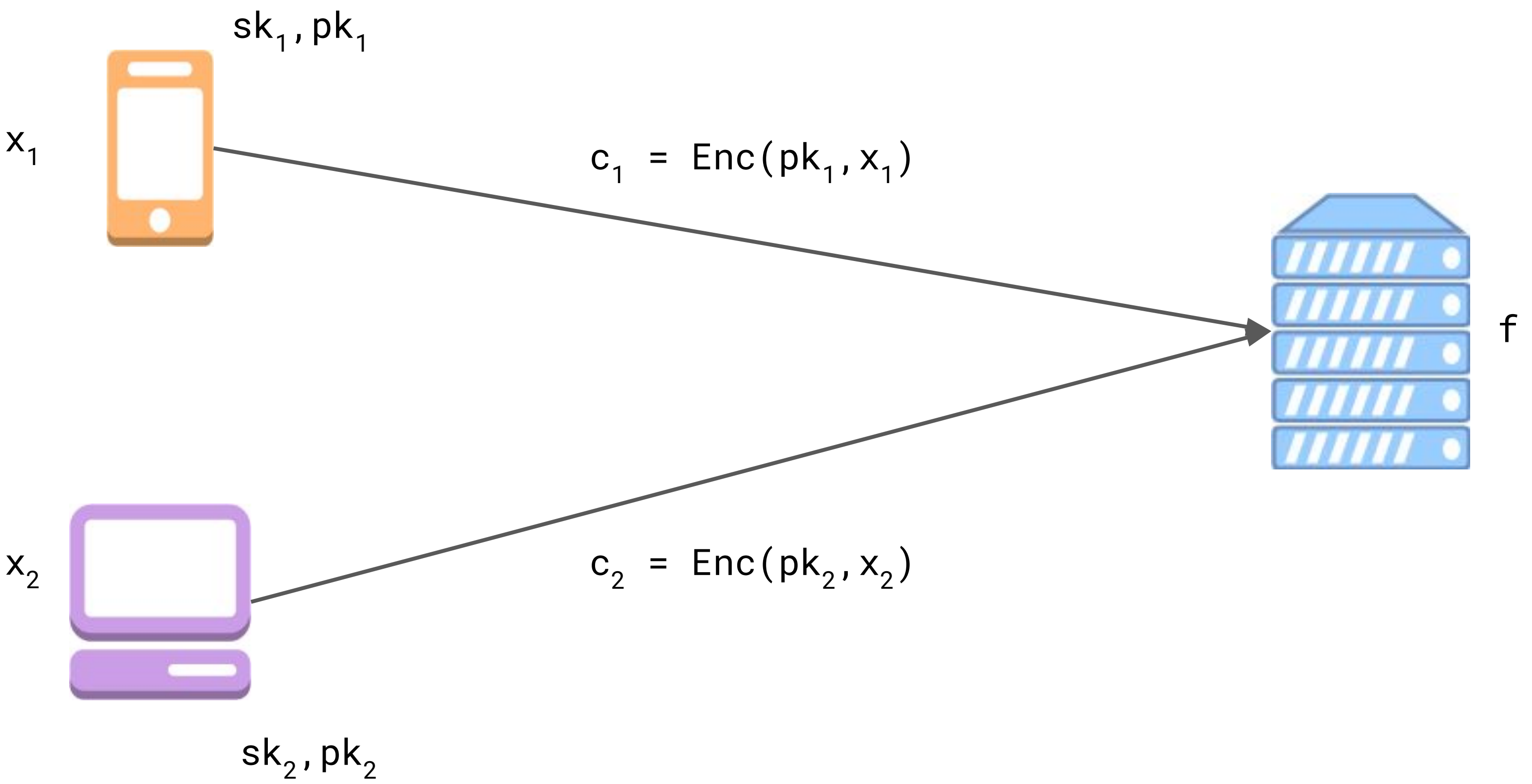
Fully Homomorphic Encryption (FHE)

- $\texttt{H = \{KeyGen,Enc,Dec,Eval\}}$
  - $\texttt{Eval}_{pk}\texttt{(f,c)}\rightarrow\texttt{c*}$
- **Homomorphic:** $\texttt{Dec}_{sk}\underbrace{\texttt{(Eval}_{pk}\texttt{(f,Enc}_{pk}\texttt{(x)))}}_{\texttt{c*}}\texttt{ = f(x)}$
  - (“Fully” Homomorphic: for every function f)
  - $\texttt{Enc}_{pk}\texttt{(f(x)), Eval}_{pk}\texttt{(f,Enc}_{pk}\texttt{(x))}$ can differ
    - As long as both distributions decrypt to $\texttt{f(x)}$
- **Compact:** $|\texttt{Eval}_{pk}\texttt{(f, Enc}_{pk}\texttt{(x))}|$ independent of the complexity of $\texttt{f}$
- **Function-private:** $\texttt{Eval}_{pk}\texttt{(f,Enc}_{pk}\texttt{(x))}$ hides $\texttt{f}$
- **Security:** Semantic security

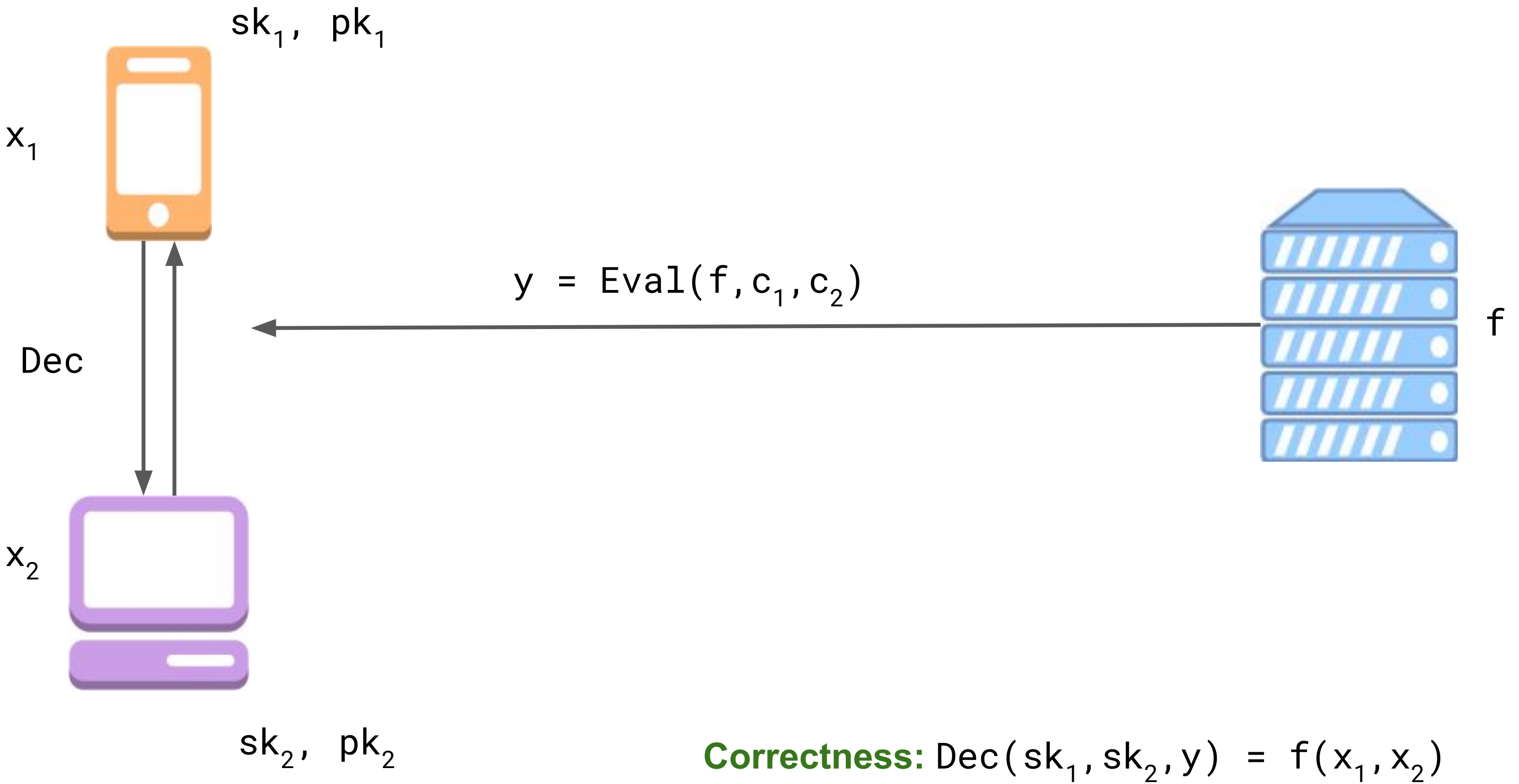
Fully Homomorphic Encryption (FHE)

- First defined: “Privacy homomorphism” [RAD’78]
  - Their motivation: searching encrypted data
- Limited variants
  - RSA & El Gamal: multiplicatively homomorphic
  - GM & Paillier: additively homomorphic
- Examples
  - RSA: $x^e \bmod N \rightarrow c$ and $c^d \bmod N \rightarrow x$
    - $x_1^e \times x_2^e = (x_1 \times x_2)^e \bmod N$
  - GM84: $\mathrm{Enc}(0) \in_R QR$, $\mathrm{Enc}(1) \in_R QNR$ (in Z_N^*)
    - $\mathrm{Enc}(b_1) \times \mathrm{Enc}(b_2) = \mathrm{Enc}(b_1 \oplus b_2) \bmod N$
- Big breakthrough → [Gentry’09]
  - First construction of FHE
  - Using algebraic number theory & “ideal lattices”

Multi-key FHE

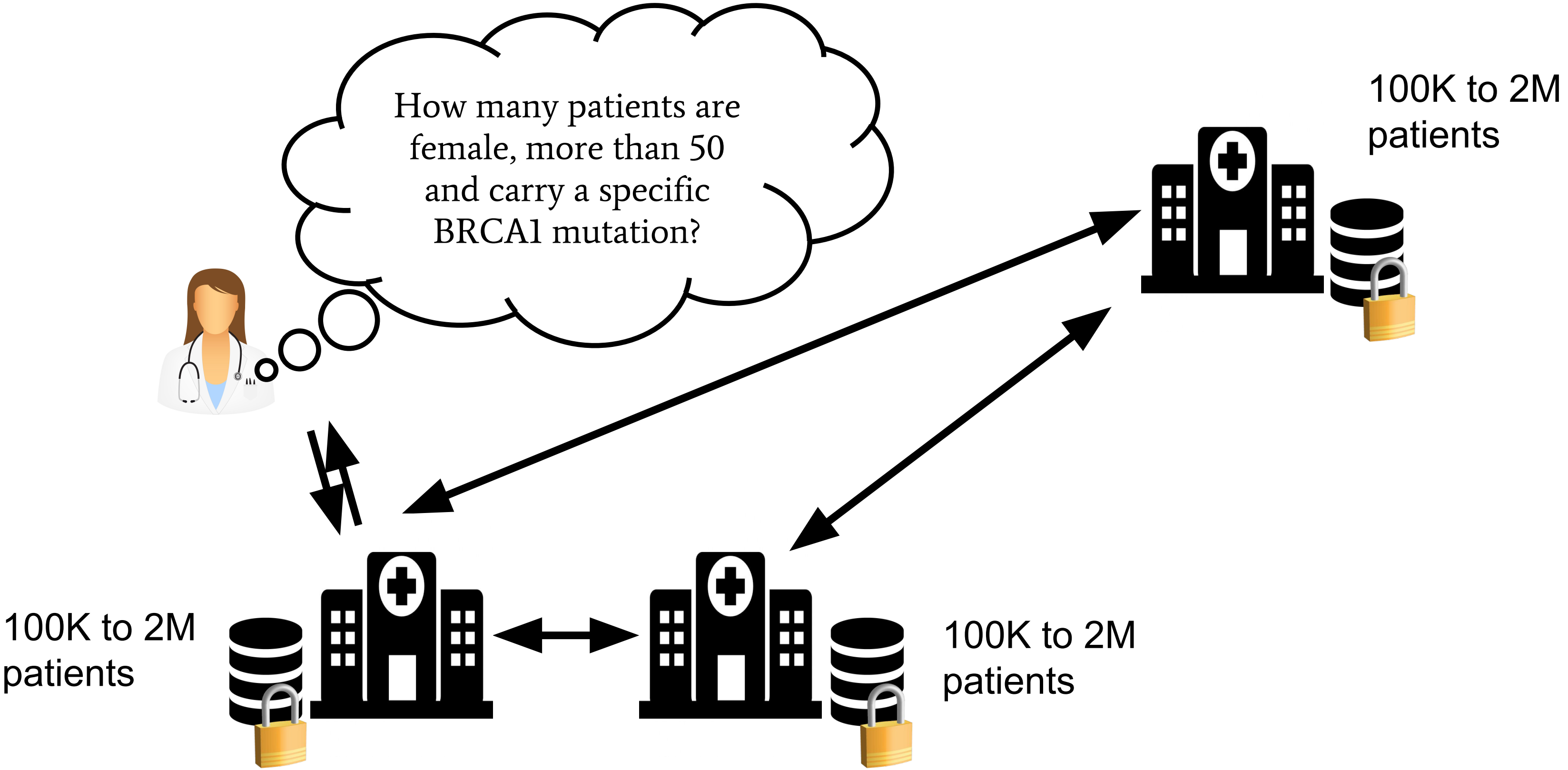


Multi-key FHE



Correctness: $\mathrm{Dec}(sk_1, sk_2, y) = f(x_1, x_2)$

UnLynx: Secure and Privacy-Conscious Medical Data Sharing

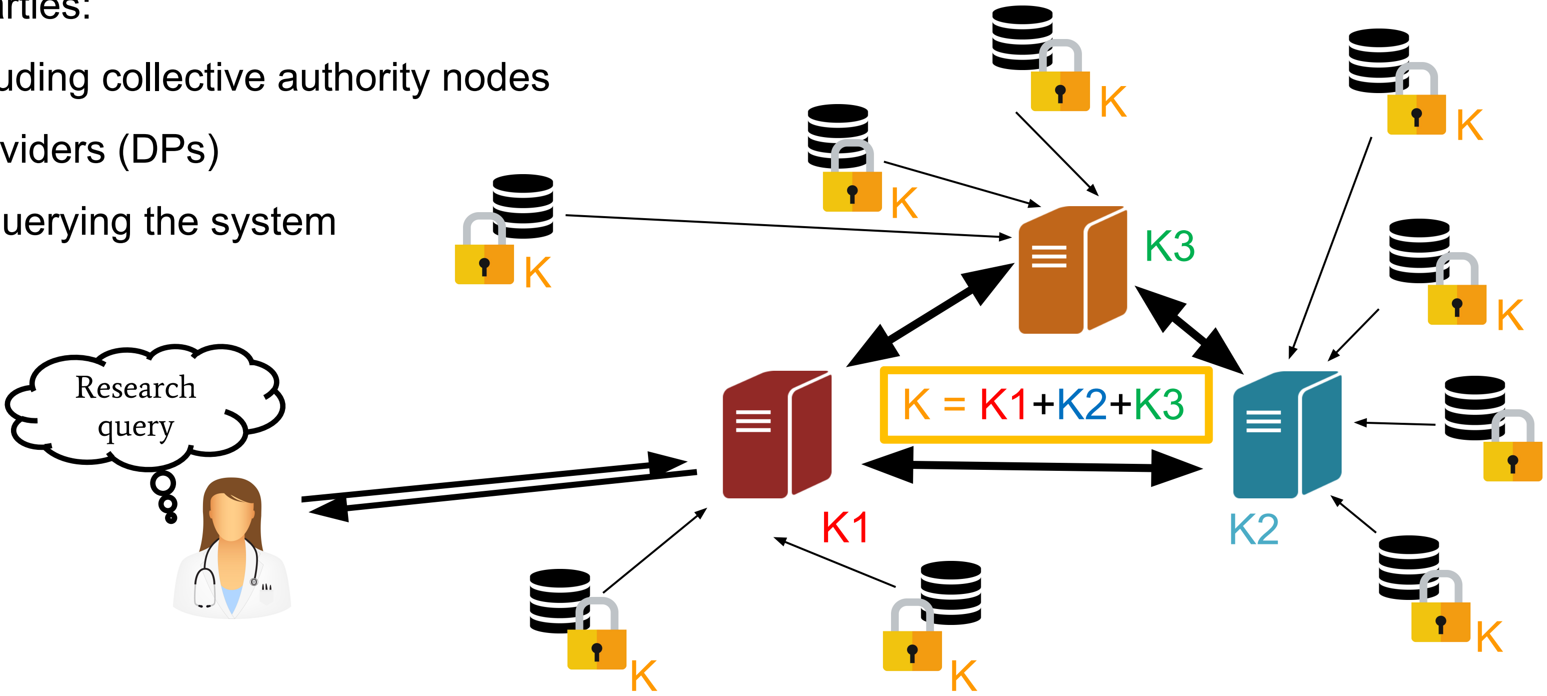


UnLynx: Privacy-conscious sharing/processing of securely distributed data

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Involved parties:

- Non-colluding collective authority nodes
- Data providers (DPs)
- Clients querying the system



Security Guarantees

0. Trust is split among multiple servers

Collective key

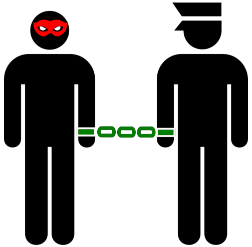
1. Data is always encrypted

Encryption

2. Results alteration is detected

ZKP

3. Servers guilty of results alteration are identifiable



ZKP

4. Data providers' privacy is protected (a data provider cannot be linked to its data)

Neff Shuffle

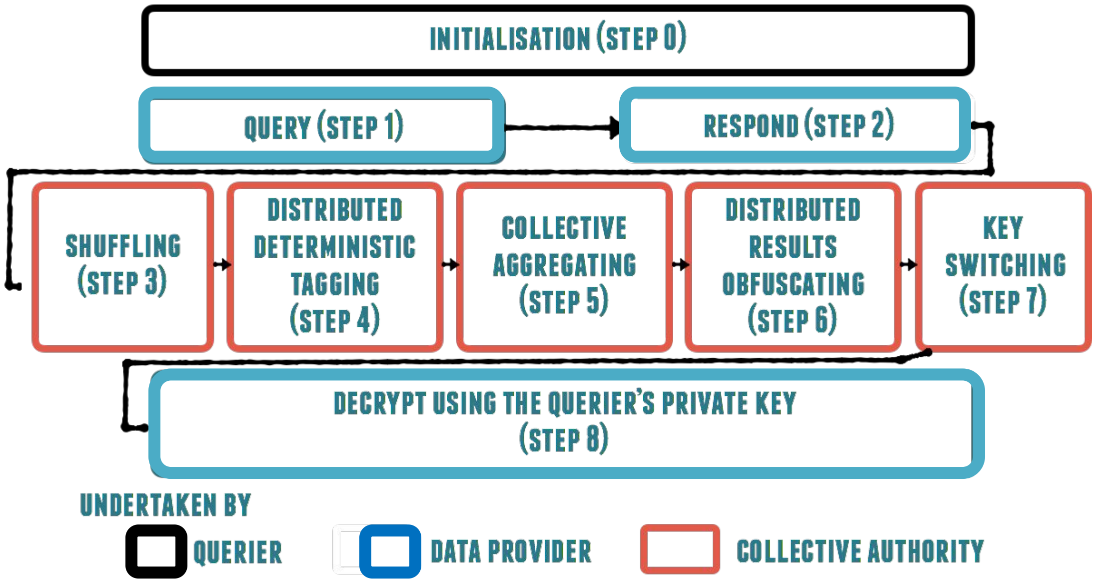
5. Data providers' data is never disclosed to another party

Encryption + Differential Privacy

Does not ensure data integrity (assume DPs are honest-but-curious) at the data providers or protect against (D)DoS attacks

UnLynx - Workflow

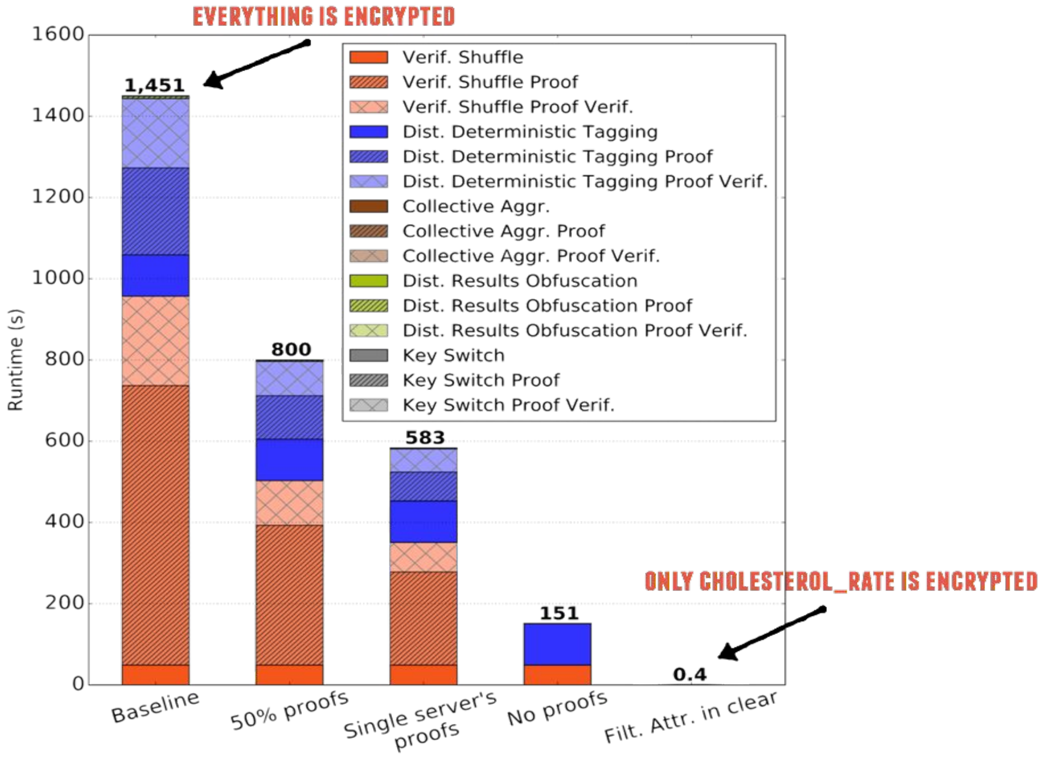
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE



- **Shuffling:** break link between data and data providers.
- **Distributed Deterministic Tagging:** permits to group/filter the responses.
- **Collective Aggregating:** aggregation of all responses.
- **Distributed Results Obfuscating:** Addition of noise to query results in order to ensure differential privacy
- **Key Switching:** transform the data encryption from the collective authority public key to the researcher key without decrypting.

Correctness of every computation can be verified with Zero Knowledge Proofs

Performance Evaluation



QUERY

SELECT AVG (cholesterol_rate)
FROM DP_1,...,DP_{20}
WHERE age in [40:50]
AND race=Caucasian
GROUP BY gender

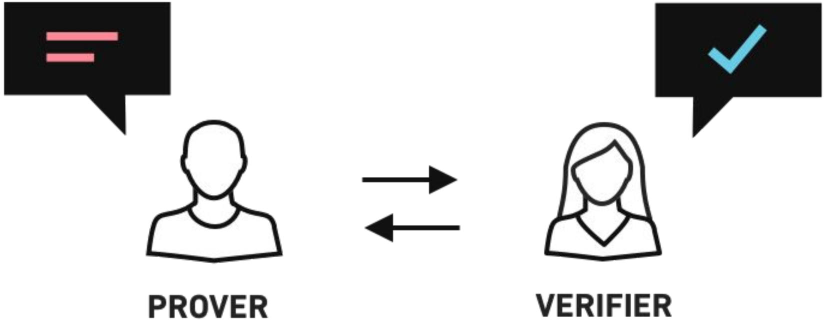
Outline

- Secure Multi-Party Computation
- Homomorphic Encryption
- **Zero-Knowledge Proofs**
- Private Information Retrieval

Zero-Knowledge Proofs

- First proposed in 1985 by S. Goldwasser, S. Micali and C. Rackoff in their paper “The knowledge complexity of interactive proof systems”:

  A zero-knowledge protocol is a method by which one party (the prover) can prove to another party (the verifier) that something is true, without revealing any information apart from the fact that this specific statement is true.



Credit: Lukas Schor

Motivation

- Audit the books of a bank for solvency without access to account values

Customer	Balance
Alice	10Ƀ
Bob	1.5Ƀ
Eve	0.5Ƀ

Is it really 12Ƀ that you have in total?

Motivation

Other possible examples:

- Audit the blockchain of privacy-preserving cryptocurrencies, e.g. ZCash, to prevent double-spending
- Charging customers for energy consumption without learning how much they have consumed (practically, whether they have paid everything or not)
- Proving identity during authentication without revealing it

Properties

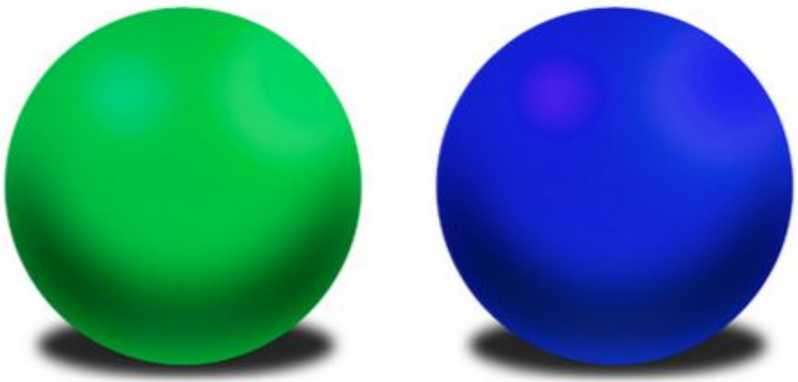
1. **Completeness**: If the input is true, the honest verifier will eventually be convinced by an honest prover

2. **Soundness**: If the input is false, no cheating prover can convince the honest verifier that it is true, except with some small probability

3. **Privacy**: The input can not be obtained by any other party

Intuition: Two balls and a colour-blind friend

- You want to prove to your colour-blind friend that two balls are differently-coloured, but not which one is which
- The friend hides the balls behind his back, randomly switches them and asks you whether he has switched
- You can know the answer by looking at the colour of a presented ball



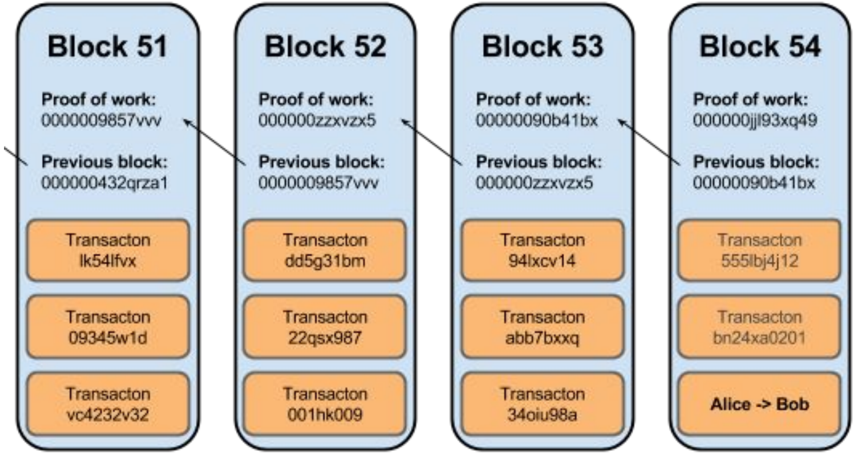
Intuition: Two balls and a colour-blind friend

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- Your friend might think that you were just lucky and is not yet completely convinced that both balls have indeed different colors
- Zero-knowledge proofs solve this problem by repeating the experiment over and over again, reaching any probabilistic level of proof that is desired
- More recently non-interactive ZKP saw the light: ZKSnarks, Bulletproof. But they are often very costly to produce on the prover’s side.

Zero-Knowledge Proofs on Blockchains

- In regular blockchains, the details of each transaction are visible to every other party in the network
  - Great for auditability
  - Problem for privacy

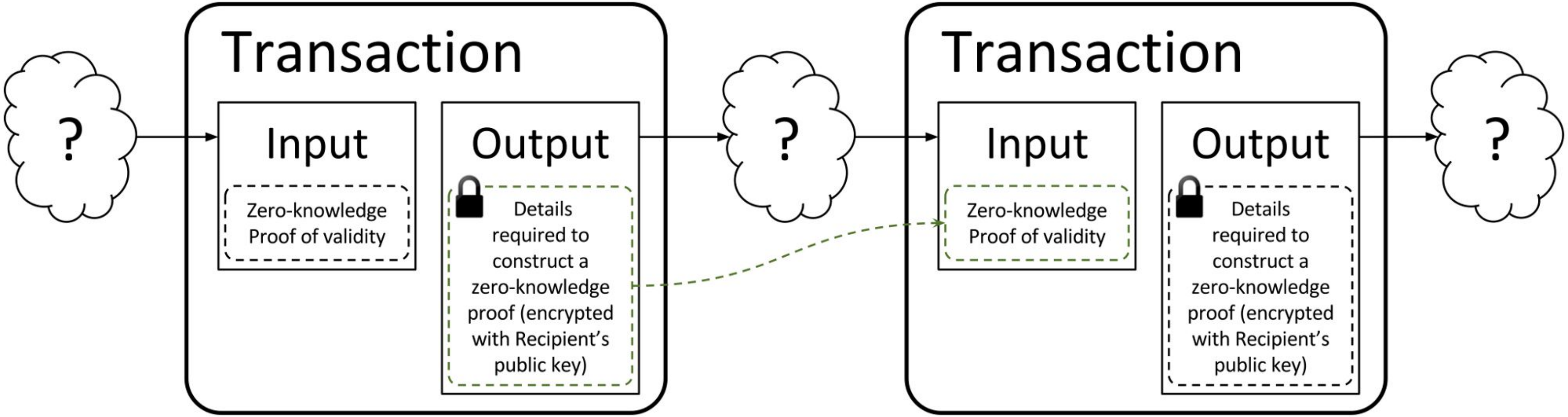


Source: [Bitcoinist]

Zero-Knowledge Proofs on Blockchains

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- Zero-knowledge protocols enable the transfer of assets across a distributed blockchain network in a privacy-preserving way



Credit: [Lukas Schor]

Zero-Knowledge Proofs on Blockchains

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- Zcash is a cryptocurrency that offers privacy and selective transparency of transactions

ZCASH

- Encrypts the contents of shielded transactions

- Uses a zero-knowledge proof construction called a zk-SNARK to maintain a ledger of balances without disclosing the parties or amounts involved

Zcash Details

- Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs)
- Proofs are used to verify the validity of a transaction
- The sender constructs a proof such that
  - The input values sum to the output values
  - The sender proves that they have the private spending keys of the input
  - The private spending keys of the input notes are cryptographically linked to a signature over the whole transaction

Zcash Details

- A cryptographic commitment instead of Bitcoin's UTXO

```
Commitment = HASH(recipient address, amount, rho, nonce)
```

- rho is a unique secret number
- To spend a transaction, the sender publishes a nullifier with rho from an existing commitment that has not been spent, and provides a zero-knowledge proof demonstrating that they are authorized to spend it

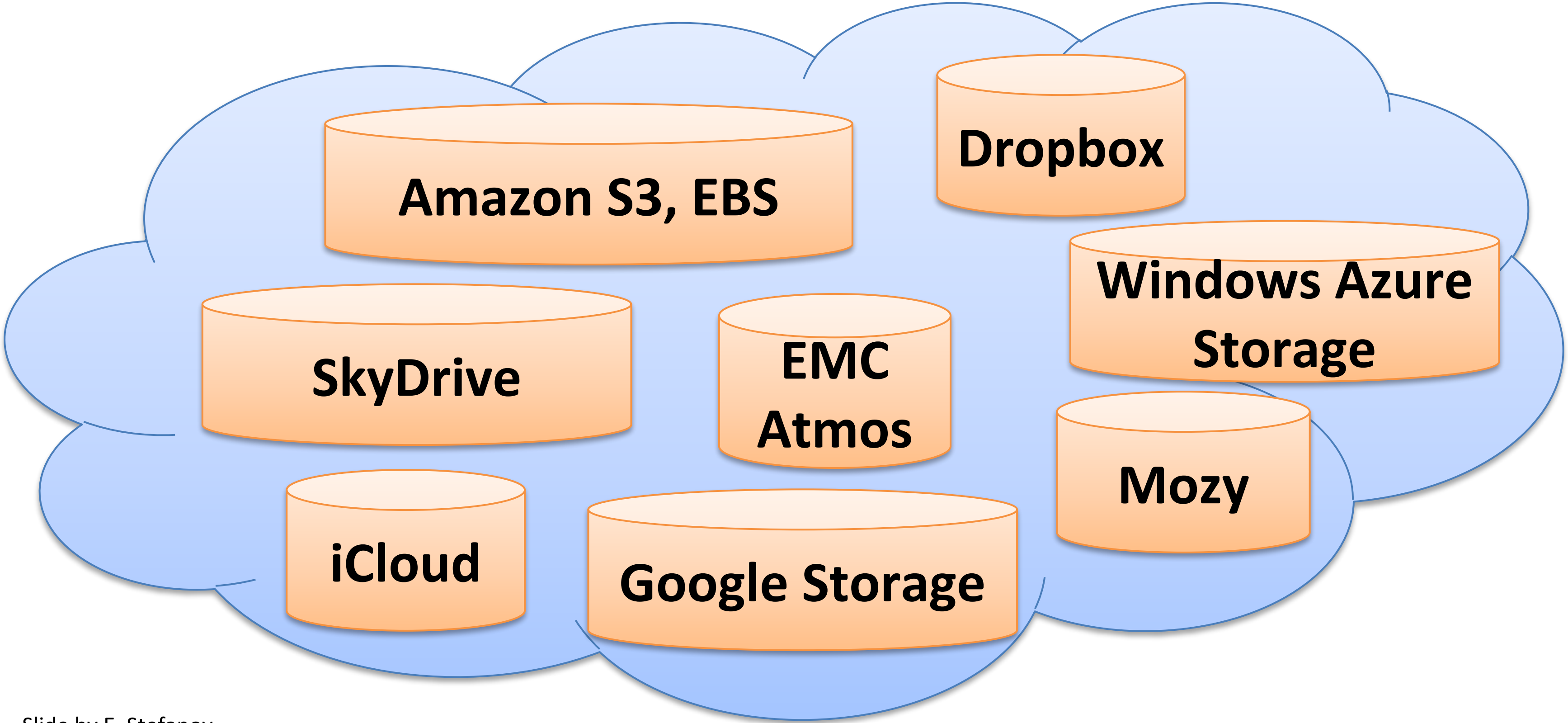
```
Nullifier = HASH(spending key, rho)
```

Outline

- Secure Multi-Party Computation
- Homomorphic Encryption
- Zero-Knowledge Proofs
- **Private Information Retrieval**

(Slide credits: Jean-Pierre Hubaux, Advanced Topics on Privacy Protection)

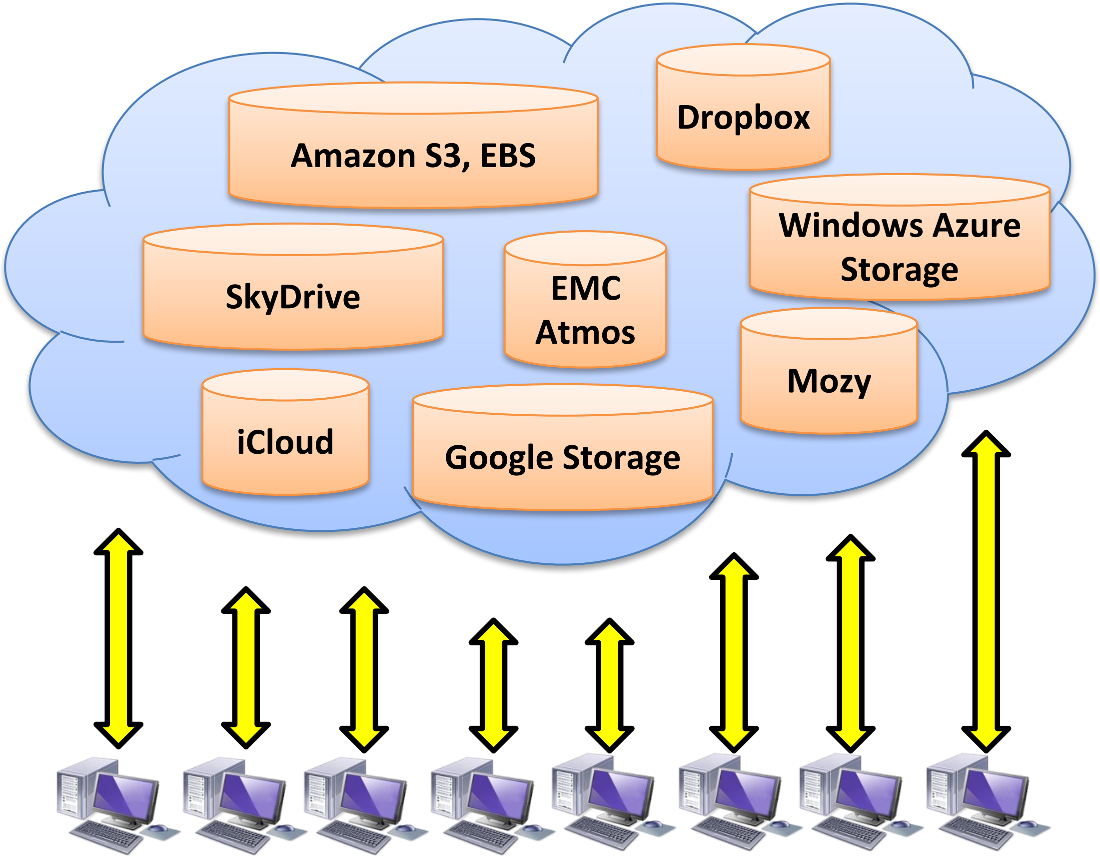
Cloud Storage



Slide by E. Stefanov

Cloud Storage

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE



Can we **TRUST** the cloud?

Slide by E. Stefanov

Data Privacy

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- Data privacy is a growing concern
    - Large attack surface (possibly hundreds of servers)
    - Infrastructure bugs
    - Malware
    - Disgruntled employees
    - Big brother
- Hence, many organizations encrypt their data

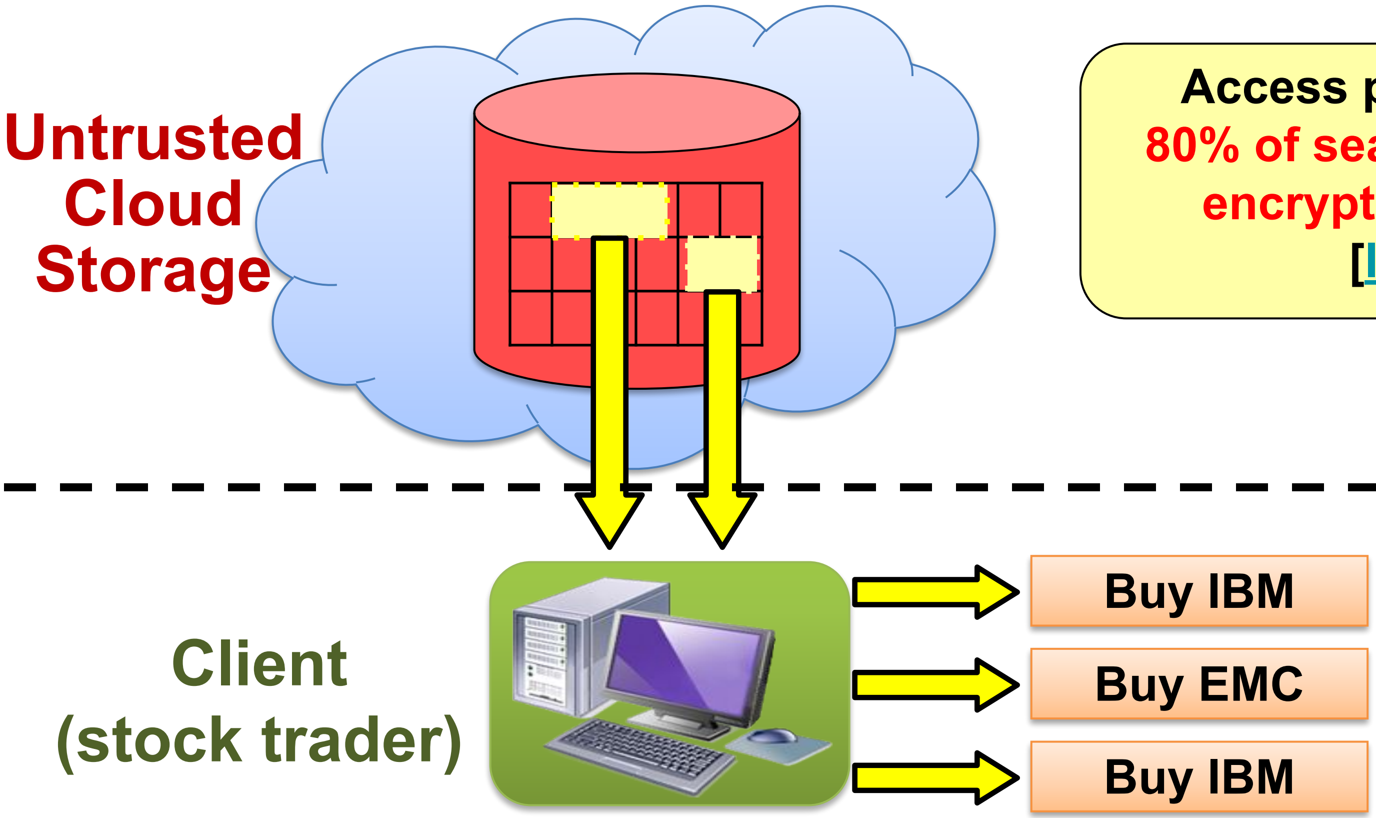
Encryption is not always enough

Access patterns

can leak sensitive information.

Example Attack by Pinkas & Reinman

Access patterns leak:
80% of search queries to encrypted database
[IQK12]



Slide by E. Stefanov

Security for Outsourced Storage

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

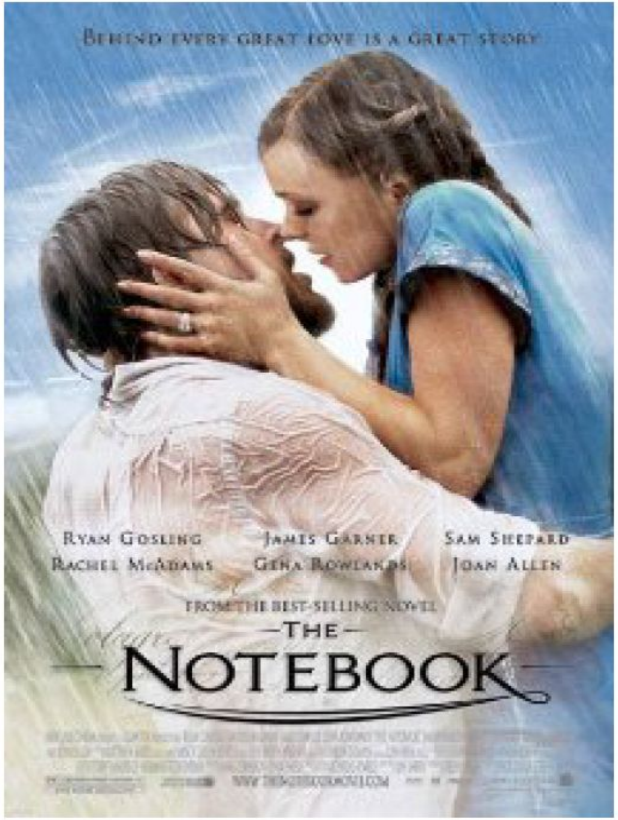
- **Confidentiality**
  - Encrypt
- **Integrity**
  - MAC & Sign
  - Merkle tree
- **Reliability**
  - Redundancy
  - Proofs of retrievability (PoR)
- **Access privacy?**
  - **Private Information Retrieval (PIR)**
  - **Oblivious RAM (ORAM)**

A Real-World Example

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

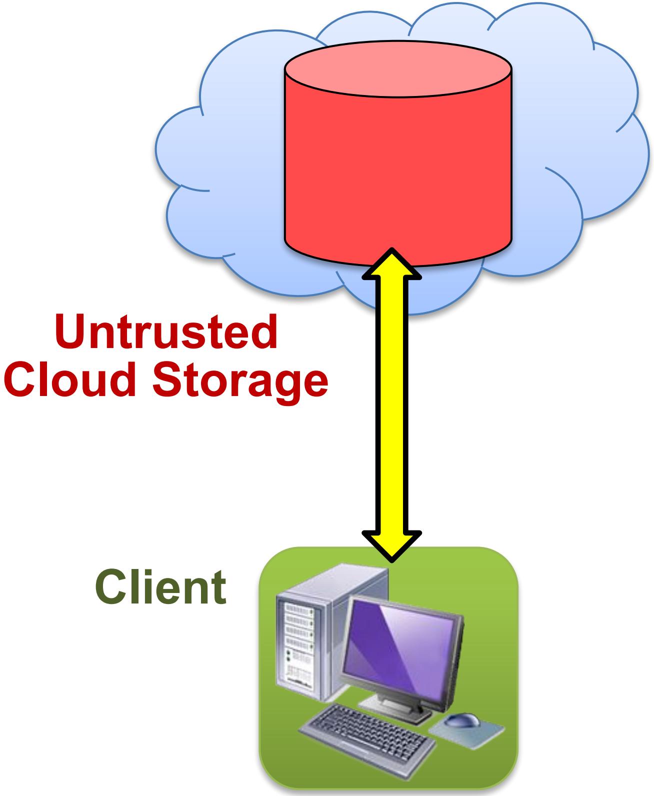
Suppose there is a movie database and I want to find information on the movie *The Notebook*.

I don't want **the database operator** to know about my interest in this movie.



Private Information Retrieval (PIR)

- **Goal:** Protect privacy of **user's queries**
- The database does not learn the query terms or responses
- Proposed by Chor et al. [CKGS95]



Private Information Retrieval (PIR)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

But…

How to do this?

Trivial Solution

Untrusted Cloud Storage

Impractical

$O(N)$ bandwidth overhead

N is the number of records in the database

Clien
(stock tr

Buy IBM

IT-PIR vs cPIR

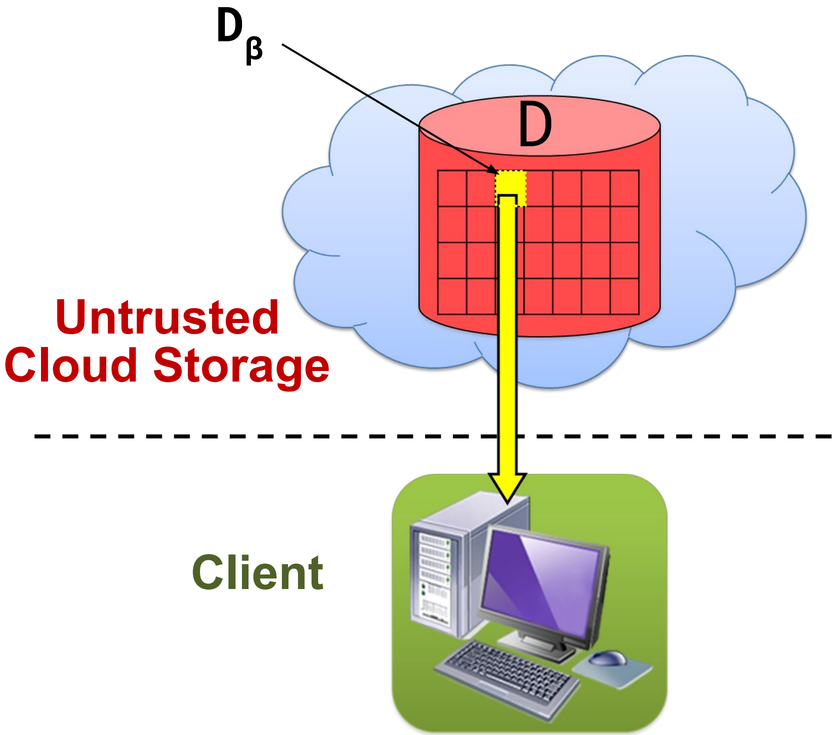
- **Information Theoretic PIR (IT-PIR)**
  - Non-colluding L servers
  - Each server holds a copy of the database
  - **Perfectly secure** if some number of these servers are not colluding
- **Computational PIR (cPIR)**
  - Single database-server
  - Uses cryptographic techniques to encrypt the user’s query
  - The security of cPIR relies on the security of the underling encryption
  - Privacy is ensured **only against computationally-bounded attackers**

IT-PIR: the Goal

Database D with blocks D_1, …, D_r

Goals:

- Retrieve D_β from the database without leaking β
- Do this without downloading the entire database



IT-PIR: Goldberg's Scheme [Gol07]

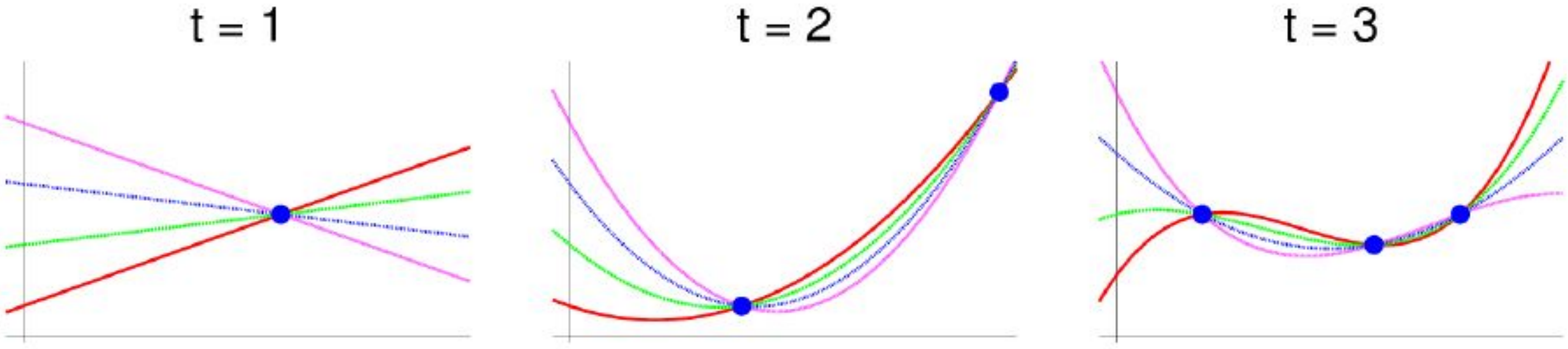
- Database D can be represented as an **r** x **s** matrix
- $\mathbf{D}_\beta = \mathbf{e}_\beta \cdot \mathbf{D}$

$$D_\beta = \overbrace{\begin{bmatrix} 0 & \dots & 1 & \dots & 0 \end{bmatrix}}^{r} \; \underbrace{\begin{pmatrix} D_1 & & \\ & D_\beta & \\ & & D_r \end{pmatrix}}_{s} \Big\} r$$

All 0s except a 1 for the **β** coordinate

- The (single) "1" of $\mathbf{D}_\beta$ allows the user to select a single row in database D
- The user hides the position of the "1" to the database by means of Shamir's scheme (next slide)

Shamir Secret Sharing (reminder) [Sha79]

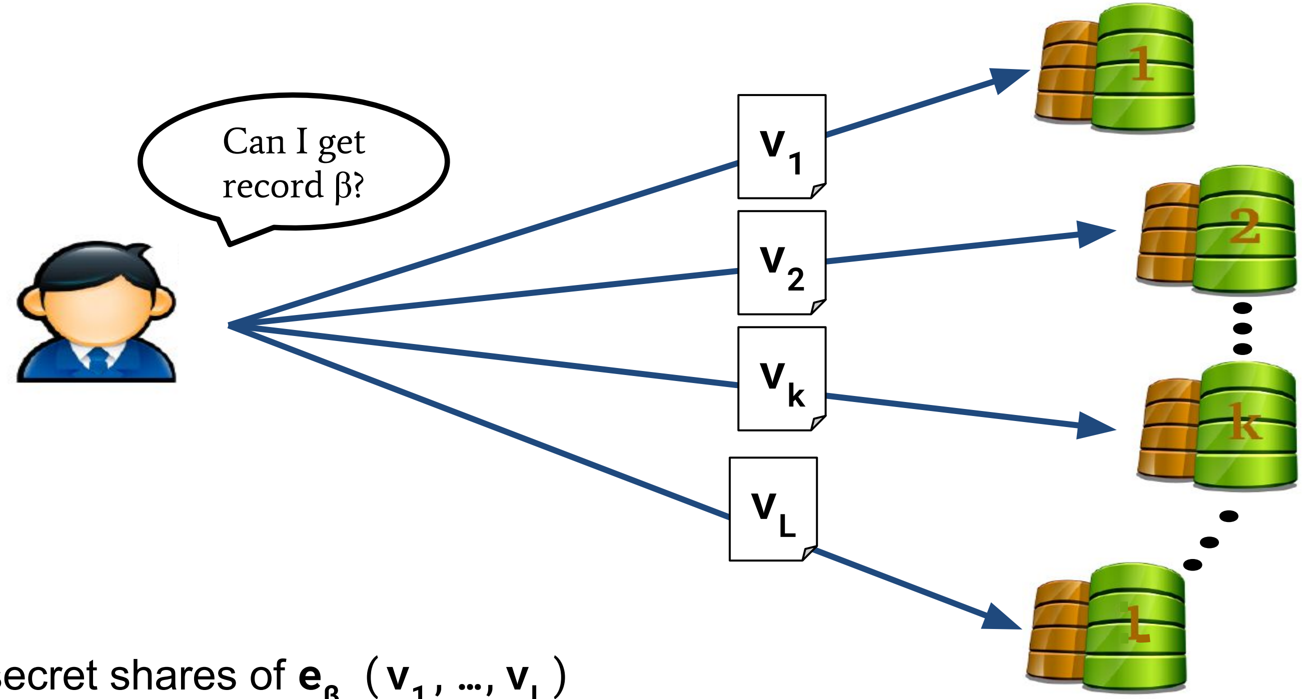


(Simplified version, just to convey the intuition)

Construction:

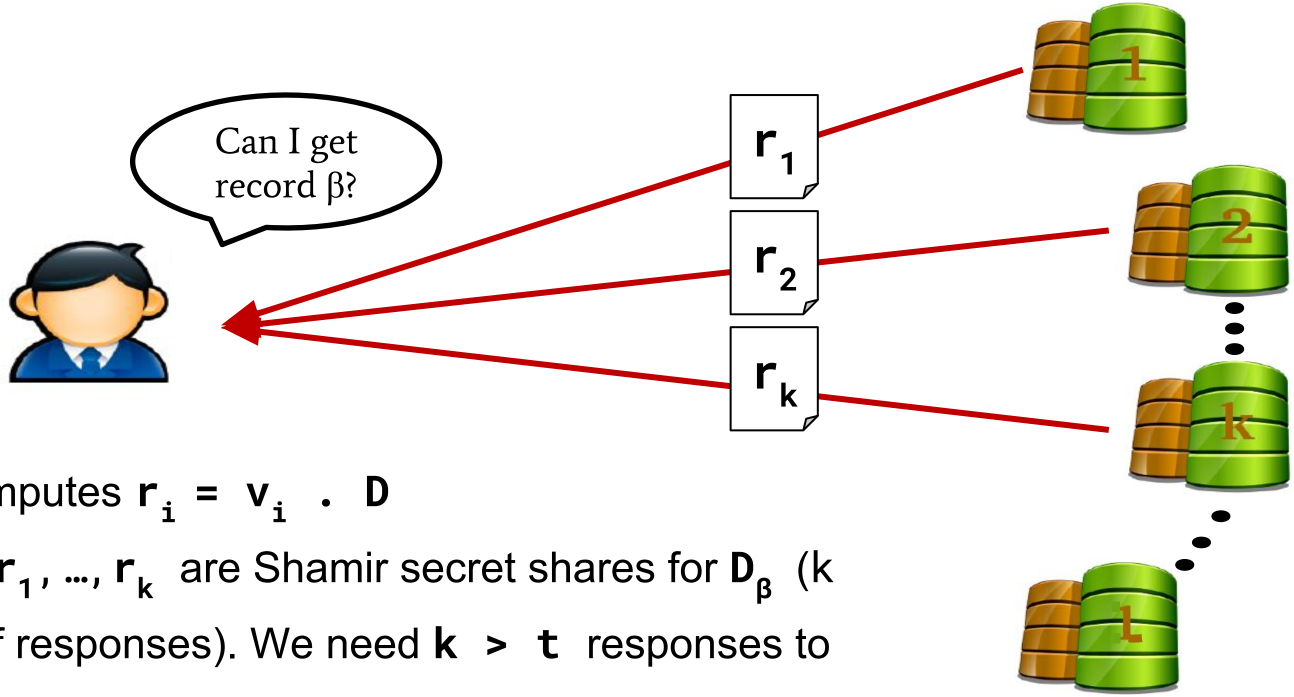
- Assume the presence of `L` parties, we pick a random point (the secret) in a field and a polynomial of degree `t` s.t. the secret is the *y-axis* intercept of that polynomial and `L ≥ t+1`
- We then pick `L` random points on this polynomial and each party is provided with one
- If we know at most `t` points we cannot reconstruct the secret.
- There is only 1 polynomial of order `t` going through the `t+1` points

IT-PIR: Goldberg's Scheme (ctd.)



Generating Shamir secret shares of $\mathbf{e}_\beta$ ($\mathbf{v}_1, \ldots, \mathbf{v}_L$) and send one to each server

IT-PIR: Goldberg's Scheme (ctd.)



- Each server computes $\mathbf{r_i = v_i \cdot D}$
- The responses $\mathbf{r_1, \ldots, r_k}$ are Shamir secret shares for $\mathbf{D_\beta}$ (k is the number of responses). We need $\mathbf{k > t}$ responses to reconstruct the secret

Goldberg's Scheme: Discussion

- We assume that no more than `t` servers are colluding.
- The system will work even if some of the servers do not respond, thanks to the intrinsic redundancy of Shamir's scheme.
- The robustness problem: how many servers' responses do we need to receive to still be able to recover that database block?

IT-PIR: Robustness

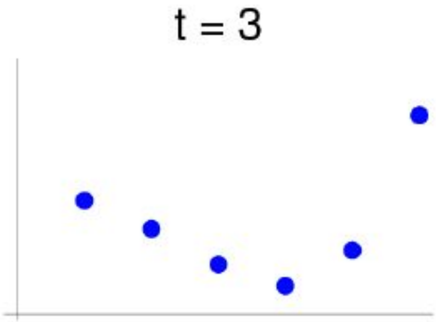
- **Robustness problem**: how many servers' responses do we need to be able to recover a database block?

- Multi-server PIR protocols tolerant of **non-responsive** or **malicious/colluding** server are called **robust** or **Byzantine robust**

- An **L**-server system that can operate where only **k** of the servers respond, **v** of the servers respond incorrectly, and which can support up to **t** colluding server without revealing the client's query is called ***"t-private v-byzantine robust k-out-of-L PIR"*** [DGH 2012]

IT-PIR: Robustness

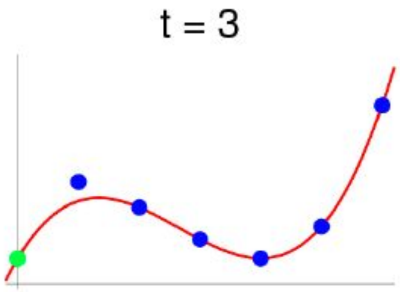
What happens if some of the responses (say v of k) are wrong?



The Shamir secret shares are a **Reed-Solomon codeword** encoding the polynomial.

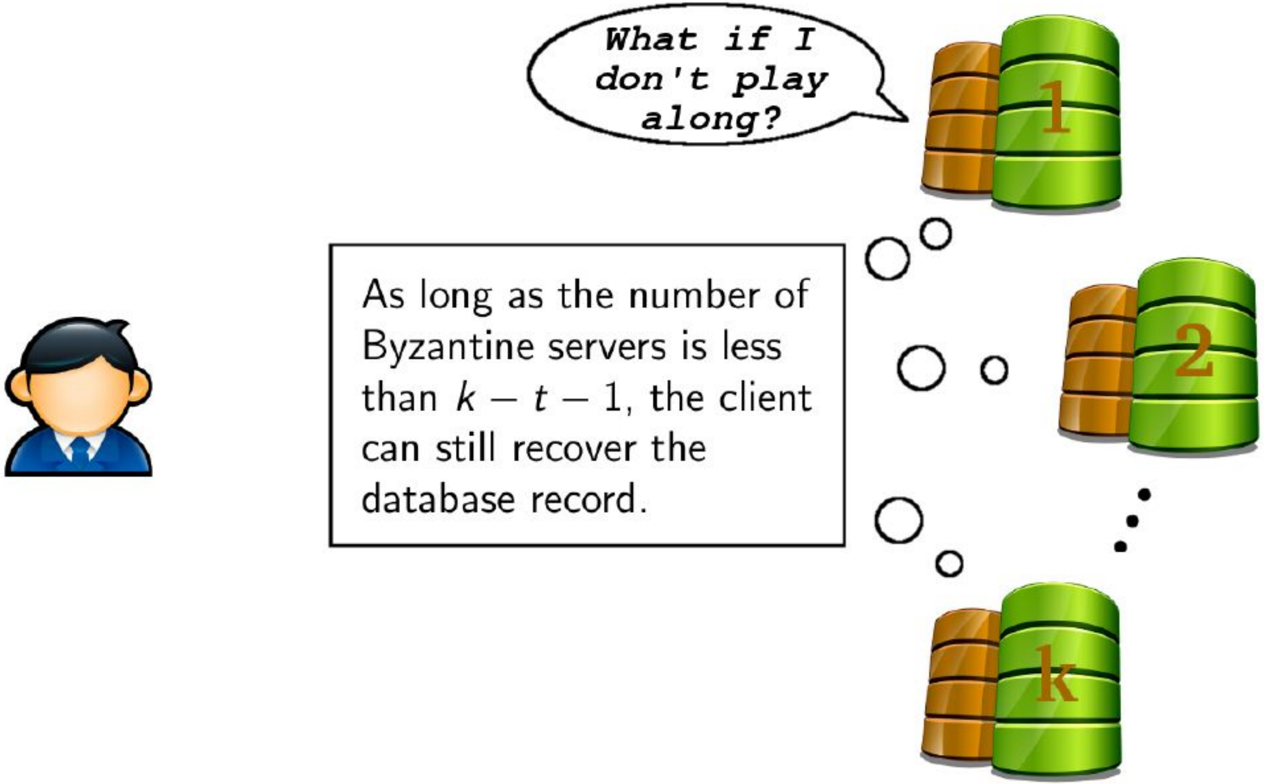
IT-PIR: Robustness

We can use **Reed-Solomon decoding algorithms** to find all polynomials of degree at most t that miss at most v of the responses. One of these polynomials is the correct one.



The **Byzantine robustness** of Goldberg's scheme is the bound on v.

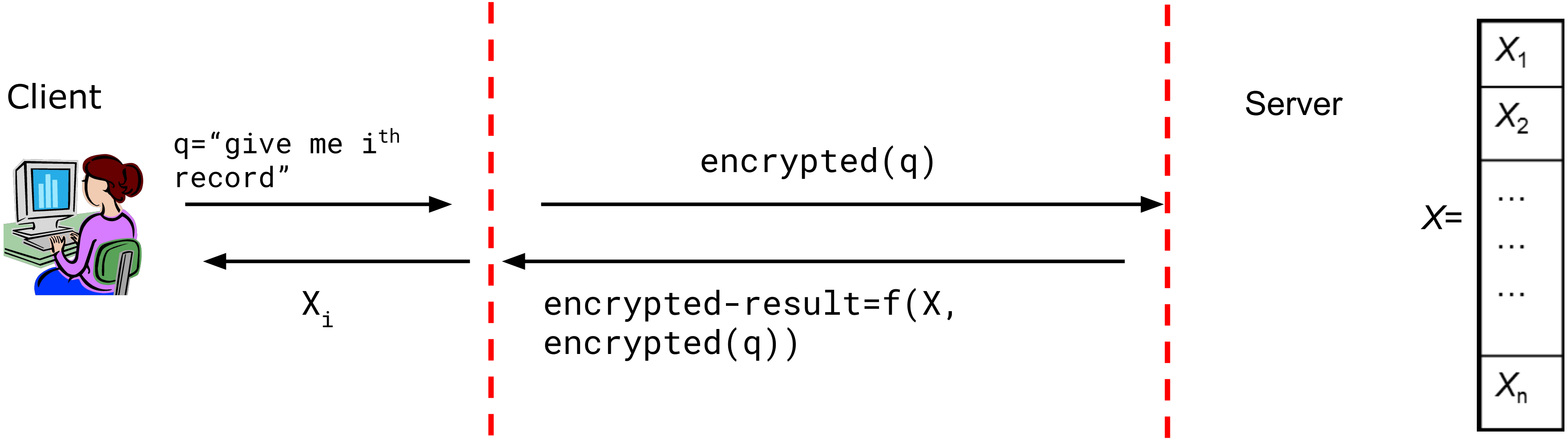
IT-PIR: Robustness



As long as the number of Byzantine servers is less than $k - t - 1$, the client can still recover the database record.

Computational PIR (cPIR)

- User privacy is related to the (assumed) intractability of a mathematical problem.
- **Principle:** Achieve computationally complete privacy by applying cryptographic computations over the entire public data



Computational PIR (cPIR)

- Some schemes use Quadratic Residuosity Assumption (QRA) as the computationally hard problem – determining whether a number is quadratic residue in a given group
- The Basic Scheme [KO97] is one example
- But this can also be done using Elliptic Curves (P256, P521, Ed25519)

Conclusion on PIR

- Protects the database user against a curious DB manager
- Comes in two flavors:
    - Computational
    - Information-Theoretic
- In both cases, leads to very substantial overhead

Conclusion

Different cryptographic tools exist to save privacy, but all of them are quite expensive:

- SMC - the gold standard, but up to now way to expensive
- HE - allows for some simple calculations in a fast way
- ZKP - getting a boost through the work in the ZCash project
- PIR - used in the Geneva CH-vote project