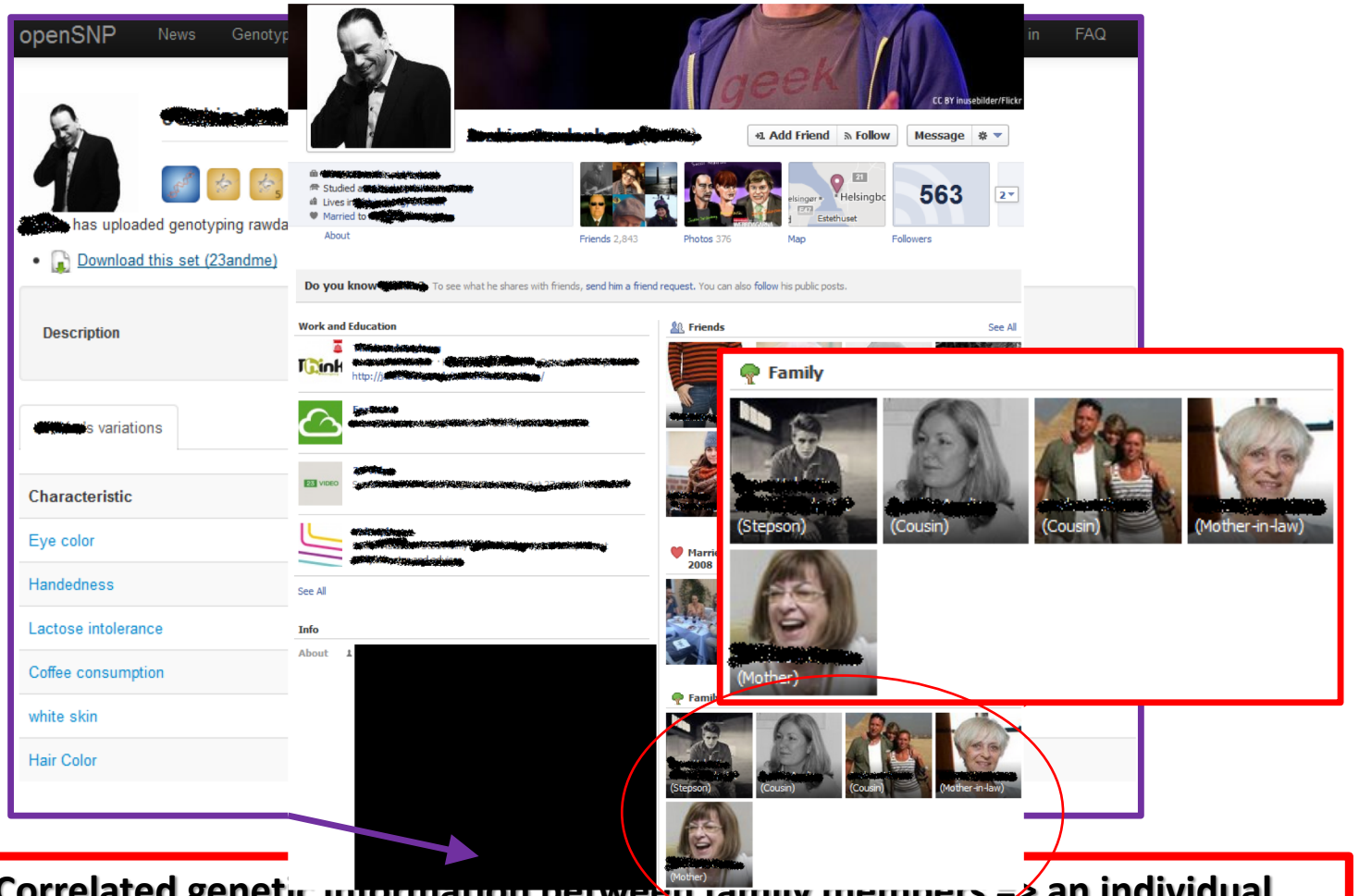
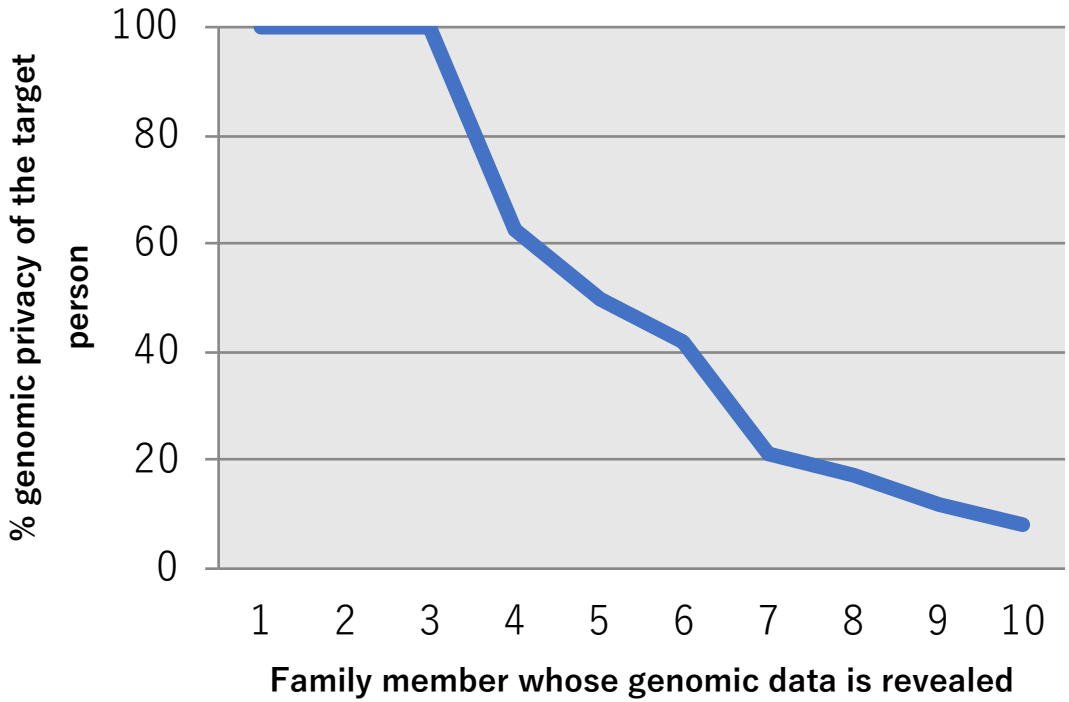
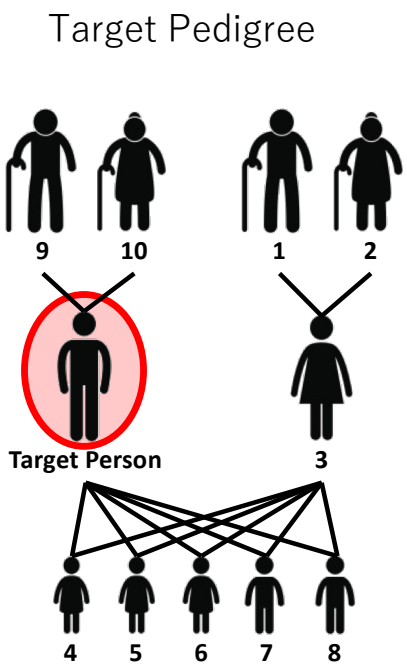
# Kin Genomic Privacy




**Correlated genetic information between family members –> an individual sharing his/her genome threatens his (known) relatives' genomic privacy**

# Quantifying Kin Genomic Privacy




M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti. Addressing the concerns of the Lacks family: Quantication of kin genomic privacy. CCS, 2013.

# Surname Inference Attack

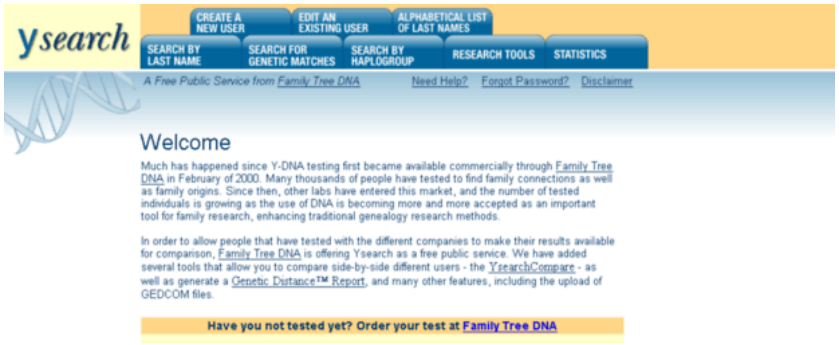
- Goals:
  - Recover the surname of sequence donors from 1000 Genome Project
  - Triangulate the identity of a sequence donor using his surname, age and state
- Using:
  - Surnames are paternally inherited in most human societies
  - Y-chromosome haplotypes in male individuals are directly inherited from the father

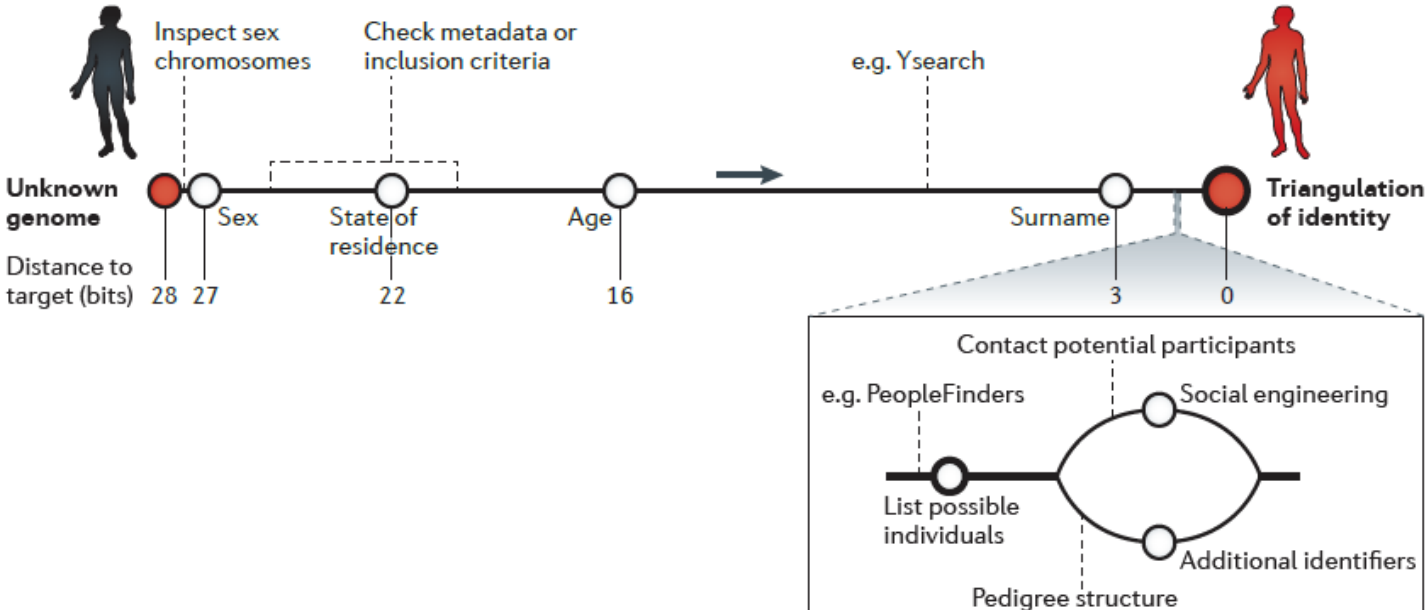
M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. Identifying personal genomes by surname inference. Science: 339 (6117), January 2013.

# Surname Inference Attack

- Surname inference
  - Profile short tandem repeats (STR) on the Y-chromosome
    - A small repeating sequence in DNA.
  - Query recreational genetic genealogy databases
  - Obtain a list of possible surnames for the sequence in question
- Identity Triangulation
  - Combine surnames with age and state
  - Triangulate the identity of the target (using US censuc database)

# Surname Inference Attack

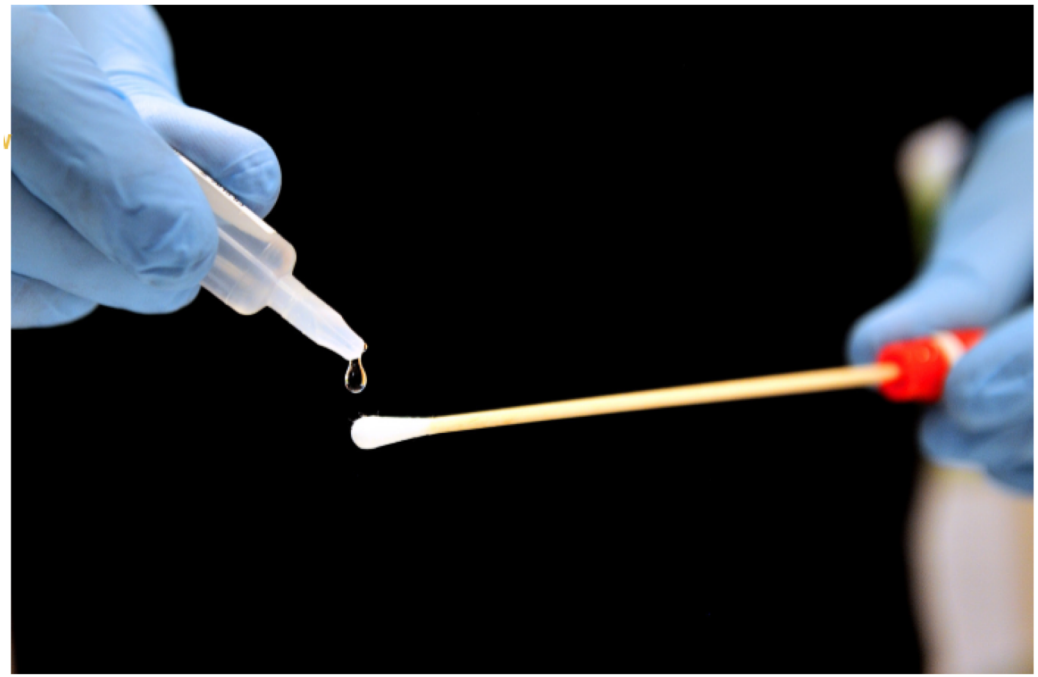


# A popular genealogy website just helped solve a serial killer cold case in Oregon

Comment

**Taylor Hatmaker** @tayhatmaker / 4 months ago



On Thursday, detectives in Portland, Ore. announced that a long-cold local murder case finally came to a resolution, 40 years after the fact.

In 1979, 20-year-old Anna Marie Hlavka was found dead in the Portland apartment she shared with her fiance and sister. According to police, she was strangled to death and sexually assaulted. Police followed a number of leads and kept tabs on the case for decades without a breakthrough.

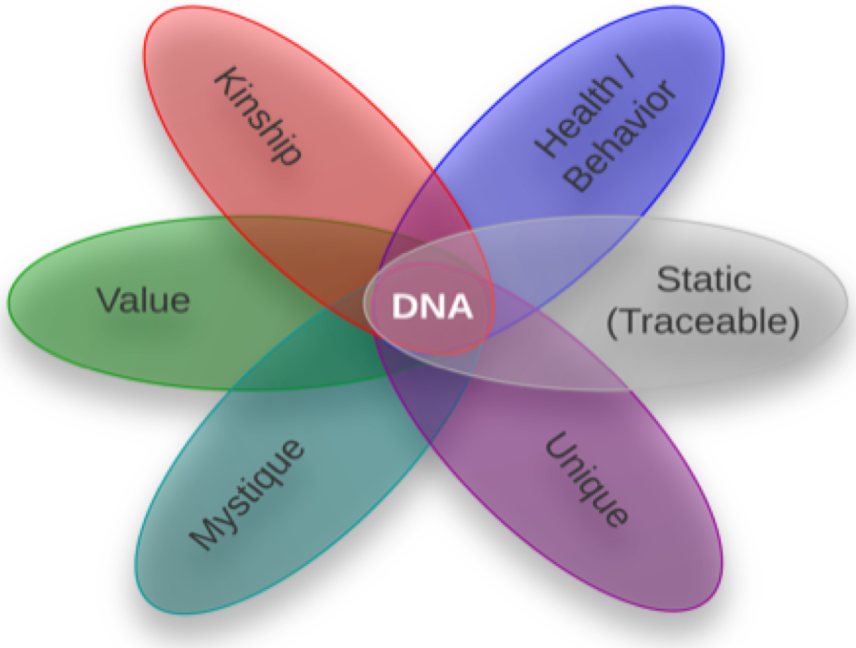
Last May, detectives with Portland's Cold Case Homicide Detail dug back into the case using the methodology made famous when investigators last year tracked down the man believed to be the Golden State Killer.

TechCrunch
January 31, 2019

# What If Genomic Data Are Leaked?

**Genomic data pose special privacy problems:**

- They are inherently identifying
- They can't be changed (as opposed to passwords)
- They have unique statistical regularities
- They contain sensitive and personal information (genetic diseases or propensity to develop certain conditions)
- Their leakage can expose individuals to genetic discrimination
- Relatives can also be affected




[Naveed et al.'15]

# Genome Privacy and Security: a Grand Challenge for Mankind

- Required **duration** of protection >> **1 century**
- (Current) **data size**: around **300 GBytes** / person
- Need sometimes to carry out computations on **millions** (if not more) of patient records
- **Noisy** data
- **Correlations**
  - within a single genome ("linkage disequilibrium")
  - across genomes (kinship, ethnicity)


- **Several "semi-trusted" stakeholders**: sequencing facilities (including Direct-to-Consumer companies), hospitals, genetic analysis labs, private doctors,…
- **Diversity of applications** (and thus of requirements): healthcare, medical research, forensics, ancestry

# Canonical Misconception about Genome Privacy and Security

Genome privacy is hopeless, because all of us leave biological cells (hair, skin, droplets of saliva,…) wherever we go

- Those cells can be collected and used for DNA sequencing
- Hence trying to secure genomes is a lost battle
- What is wrong with this reasoning?
- Collecting human biological samples and sequencing them is expensive, illegal, prone to mistakes, and non-scalable! (even if sequencing techniques keep improving)
- The medical community (research and healthcare) **should not be** the (indirect) accomplice of massive leaks of sensitive data

# Security / Privacy Requirements for Personalized Health

- Pragmatic approach, **gradual** introduction of new protection tools
- Different **sensitivity levels** of the data
- Different **access rights**
- Exploit **existing** data (electronic health records) and tools
- Be **future-proof** (no short-sighted “bricolage”)
- Awareness and enforcement of **patient consent**

# Technologies for Privacy and Security Protection

### Traditional Encryption

- Protects data at rest and in transit
- Cannot protect computation

### Homomorphic Encryption

- Protects computation in untrusted environments
- Limited versatility vs efficiency

### Secure Multiparty Computation

- Protects computation in distributed environments
- High communication overhead

### Trusted Execution Environments

- Protects computation with Hardware Trusted Element
- Requires trust in the manufacturer, vulnerable to side-channels

### Differential Privacy

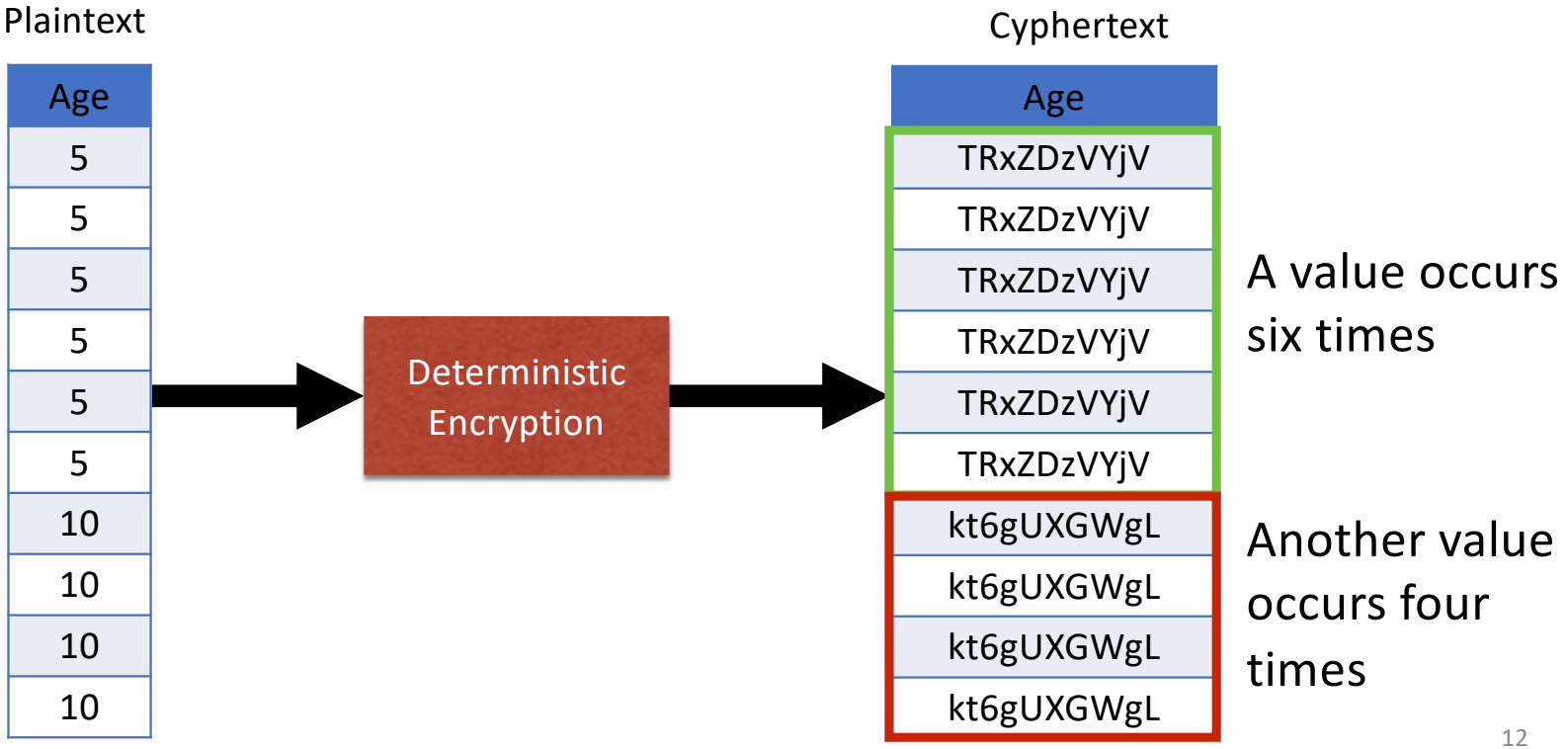
- Protects released data from inferences
- Degrades data utility (privacy-utility tradeoff)

### Distributed Ledger Technologies (Blockchains)

- Strong accountability and traceability in distributed environments
- Usually no data privacy

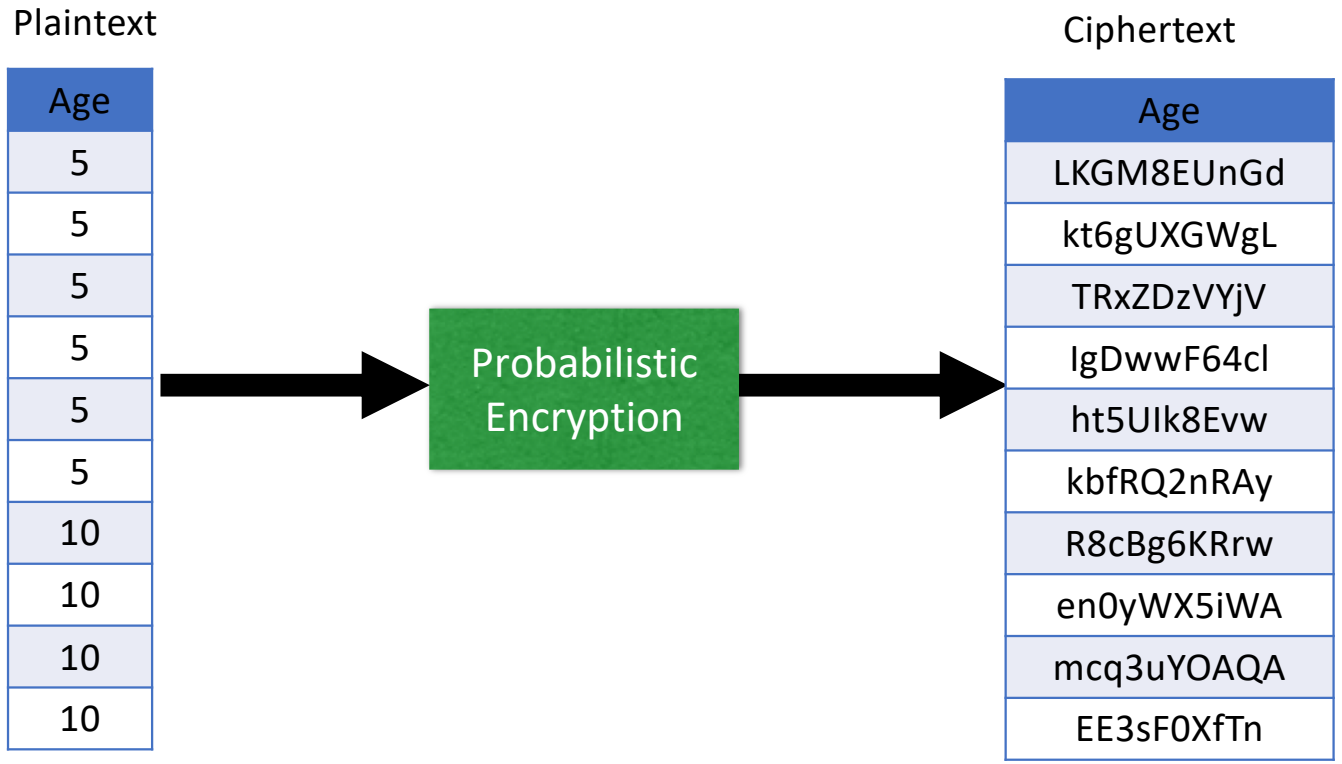
# Deterministic vs probabilistic encryption

- Deterministic encryption
  - Preserves and **leaks** equality of the the plaintext
  - More general property-preserving schemes can focus on other properties (e.g., order, format,…)



# Deterministic vs probabilistic encryption

- Probabilistic encryption
  - Random salt added to each encryption to achieve semantic security (plaintext properties are not disclosed, and equality of plaintexts cannot be determined with non-negligible advantage)
  - Problem: ciphertexts cannot be compared



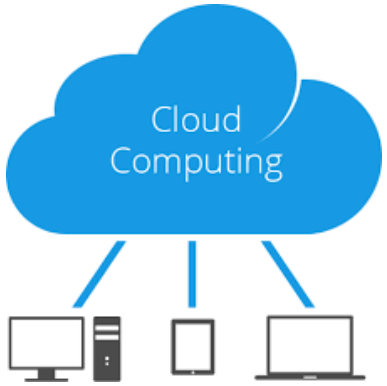
# Multi-site Studies – Where to Store the Data?

**a. Keep them at each site**

- Useful especially if the cloud is untrusted
- Better control of the data

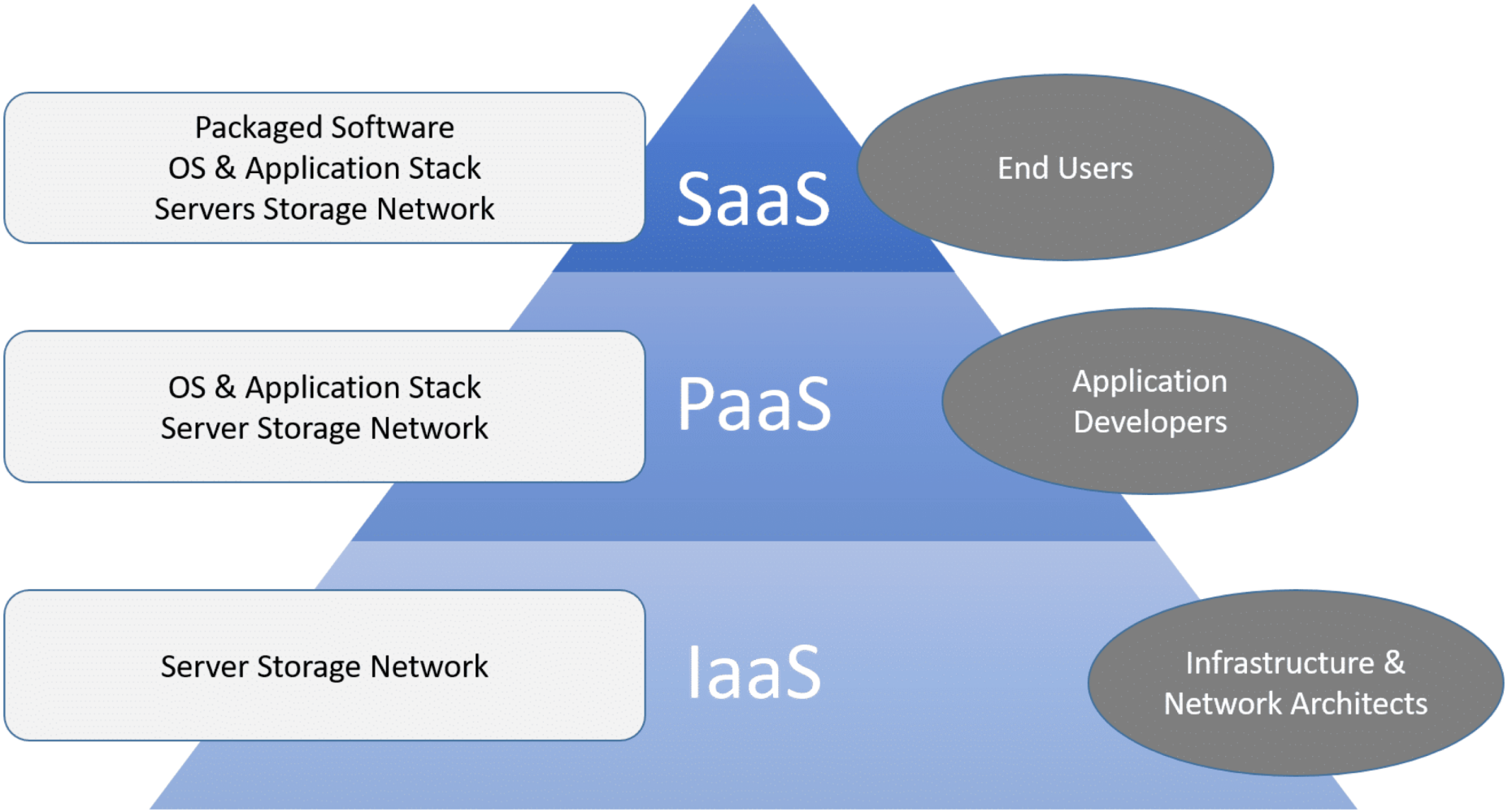
**b. In the cloud**

- Take advantage of the well-known strengths of the cloud (see next slide)

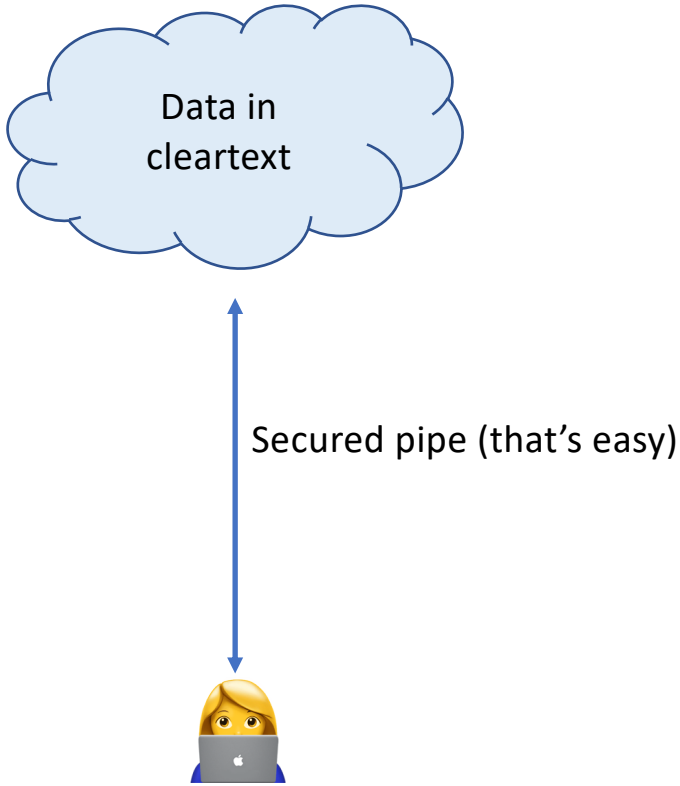



- Multiple tenants (different, usually unrelated legal entities) store their data on an infrastructure managed by another legal entity
- Benefits
  - Mutualization of storage and computation capabilities, but also of IT manpower
  - No upfront investment for the customer
  - Usually professionally managed, thus more secure
- Privacy question: is the cloud trustworthy? → encryption solutions
- Legal:
  - US/EU: Safe Harbor → Data Shield
  - US CLOUD Act (2018)
- Main global providers: Amazon (AWS), Google Cloud, Microsoft Azure

# Cloud Service Models



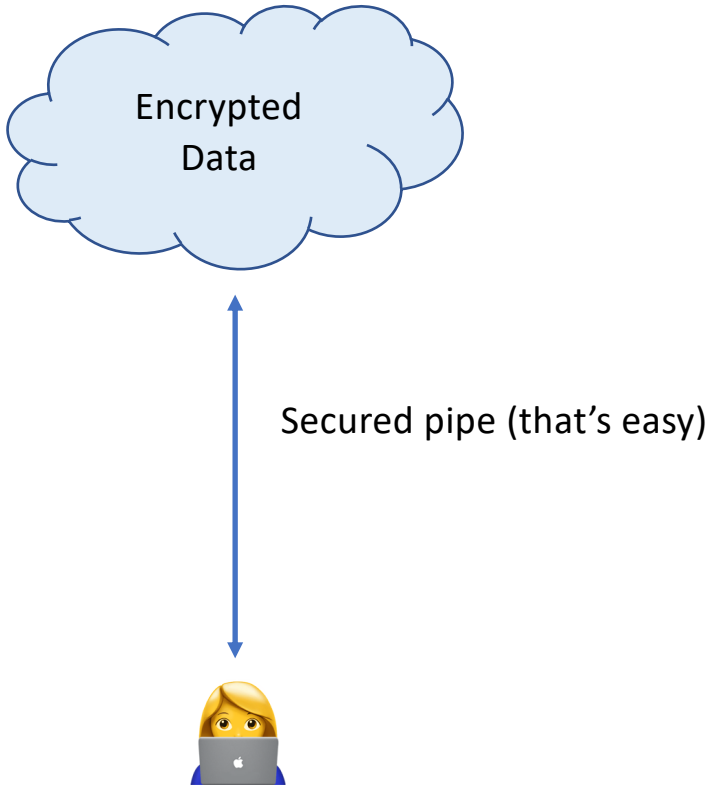
# Case 1: The Cloud is Fully Trusted – Storage in Clear Text (never happens in practice)



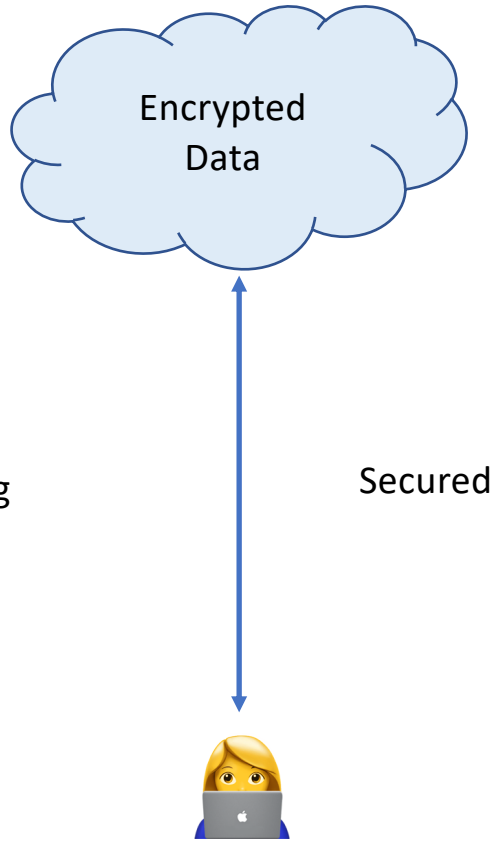

- Data sharing is easy
- Computation in the cloud is easy

# Case 1’: The Cloud is Fully Trusted – It encrypts with keys it controls

- Data sharing is easy
- Computation in the cloud is easy

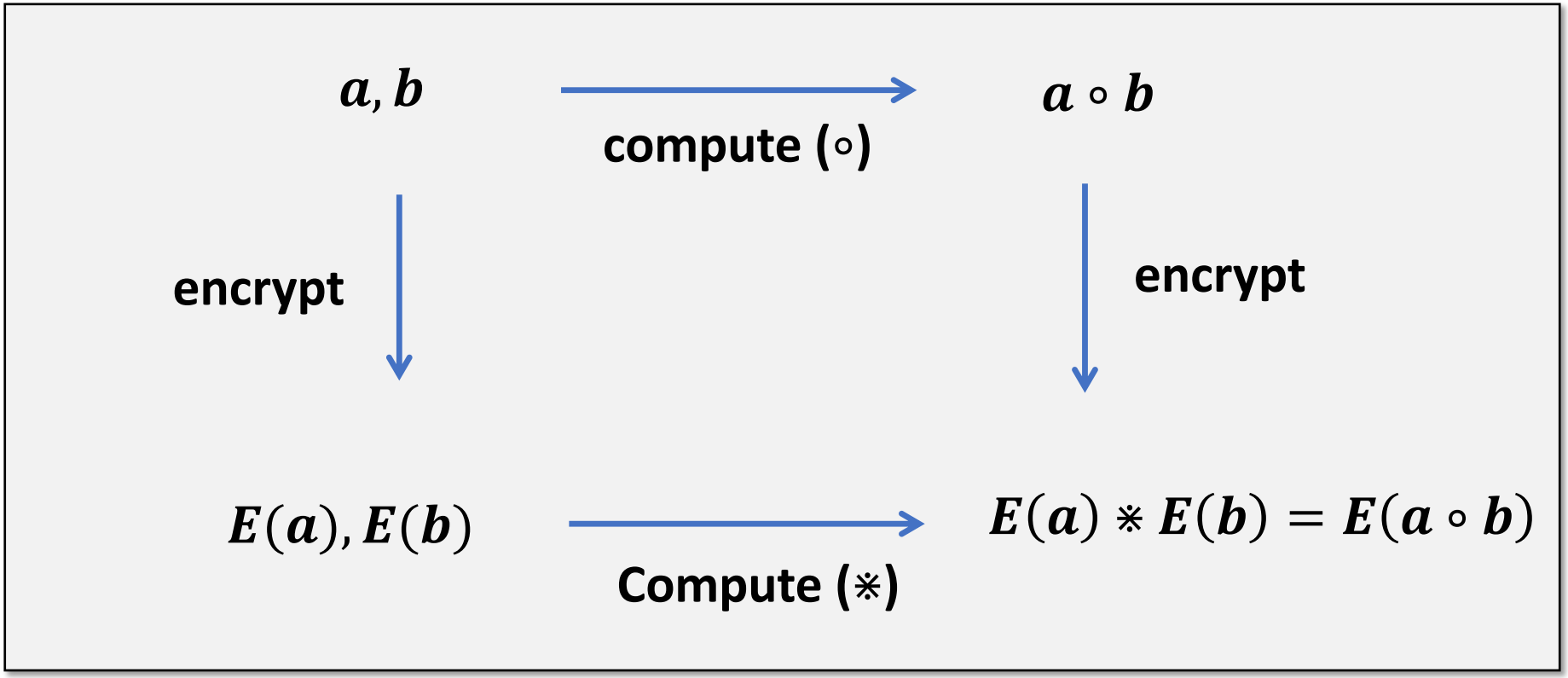


# Case 2: The Cloud Is Untrusted – The user encrypts under their own keys




- Data sharing is tricky (key management)
- Computation in the cloud is impossible
- Some of the benefits of cloud computing are thus lost
- If the user loses their keys, they lose all their data

# Homomorphic Encryption




**Homomorphic encryption enables computations directly on encrypted data.**

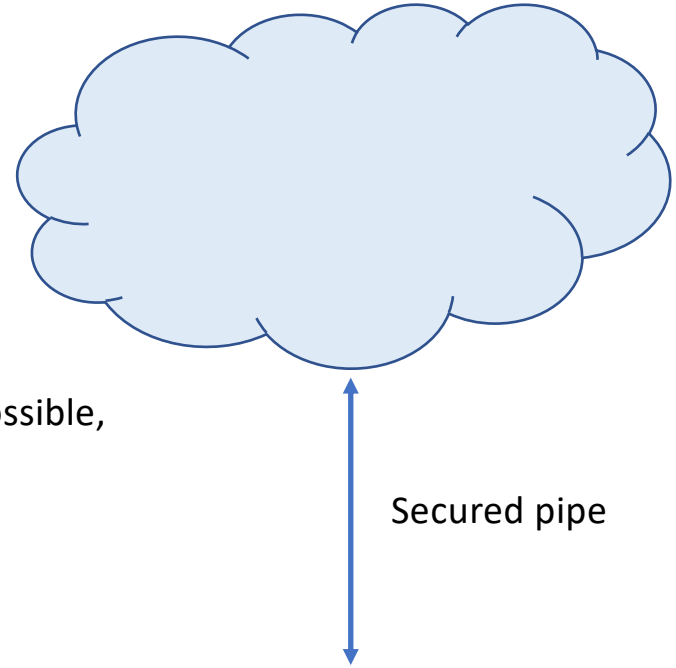
# Homomorphic encryption

- Homomorphic Encryption and Fully Homomorphic Encryption
  - (Practical/Somewhat) Homomorphic Encryption ≠ Fully Homomorphic Encryption

| Somewhat Homomorphic Encryption | Fully Homomorphic Encryption (FHE) |
|---|---|
| Can perform polynomial operations of limited degree | Can perform any operation represented as a binary circuit |
| Need to set-up security parameters tailored for the specific computation | Generic security parameters |
| Arithmetic circuits and integer inputs | Logical circuits and binary inputs |
| Constrained computational overhead (several orders of magnitude speed-up w.r.t. FHE) | Very high computational overhead |
| Expansion controlled by the ciphertext/plaintext cardinality ratio | Very high expansion |

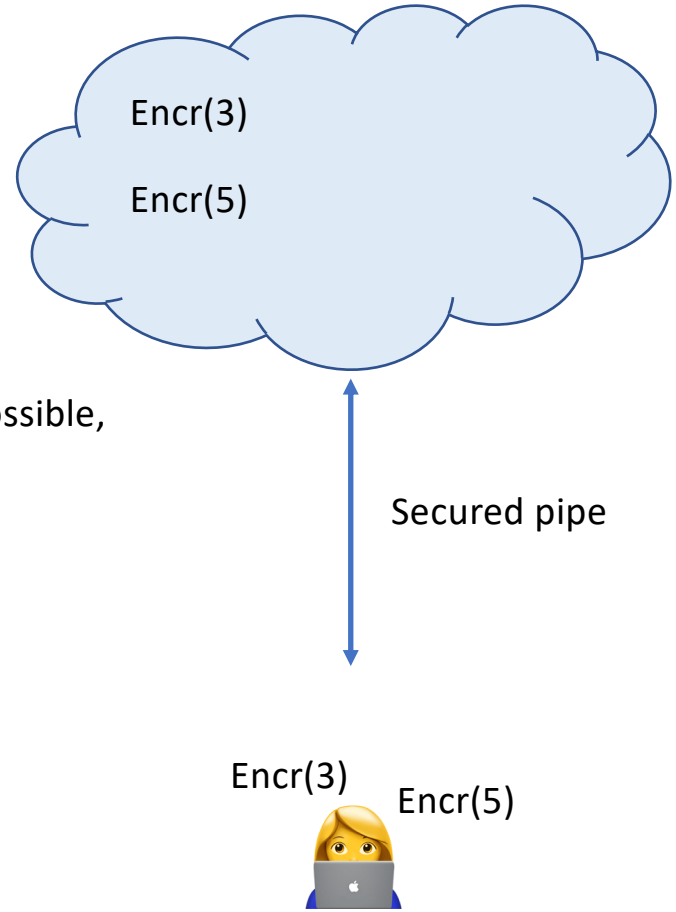
# Case 3: The Cloud is Untrusted – The user homomorphically encrypts with keys it controls (1/3)

- Data sharing is doable
- Computation in the cloud is possible, but expensive

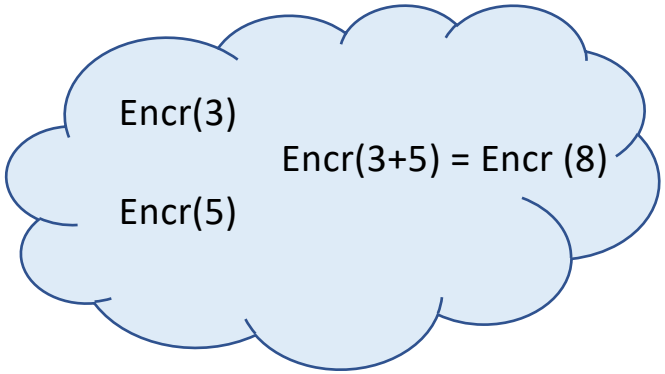


# Case 3: The Cloud is Untrusted – The user homomorphically encrypts with keys it controls (2/3)

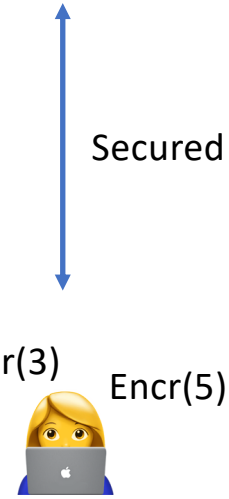
- Data sharing is doable
- Computation in the cloud is possible, but expensive



# Case 3: The Cloud is Untrusted – The user homomorphically encrypts with keys it controls (3/3)




- The cloud can make computations on encrypted data, **for which it does not know the crypto keys**
- Hence computation in the cloud is possible (albeit expensive)
- Data sharing is doable



# Multi-site Studies: Keeping the Data at Each Site

Assume Sites do not trust each other
→ Possible solution: Secure Multi-Party Computation

# Secure Multiparty Computation

Problem statement:

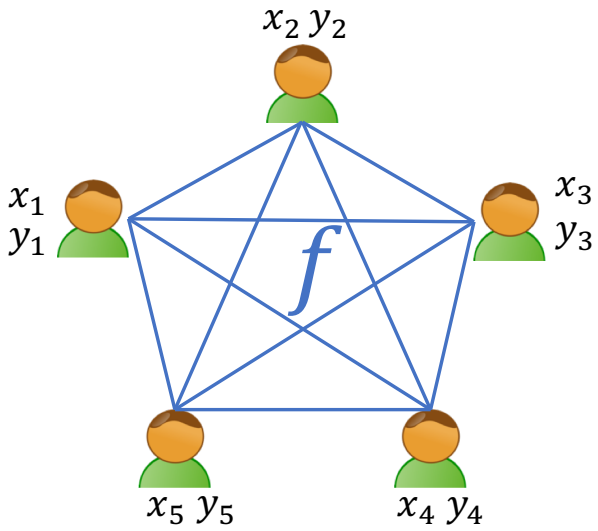
A set of players $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ would like to compute a function $f(x_1, x_2, \ldots, x_N) = (y_1, y_2, \ldots, y_N)$ of their joint inputs.

Requirements:

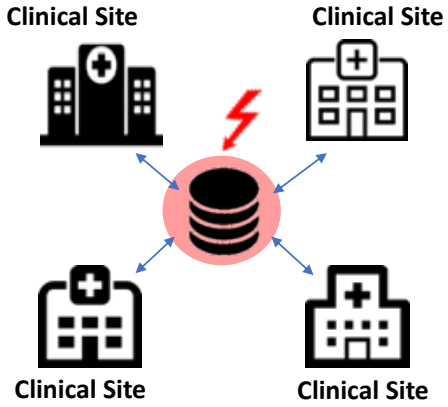
1. *Privacy*
   No party should learn anything more than its prescribed output
2. *Correctness*
   Each party is guaranteed that the output that it receives is correct

Realization:

A multiparty cryptographic protocol



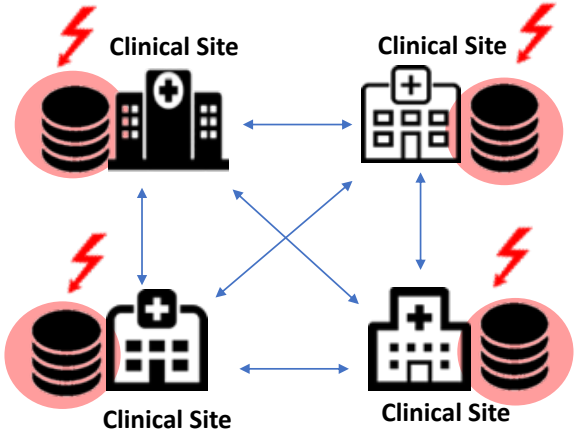
# Current approaches for medical data sharing




**Centralized approach**

e.g., Genomics England, All of Us Research Program, Human Longevity, 23andMe

- Single point of failure
- Legal, ethical and social constraints for outsourcing sensitive data

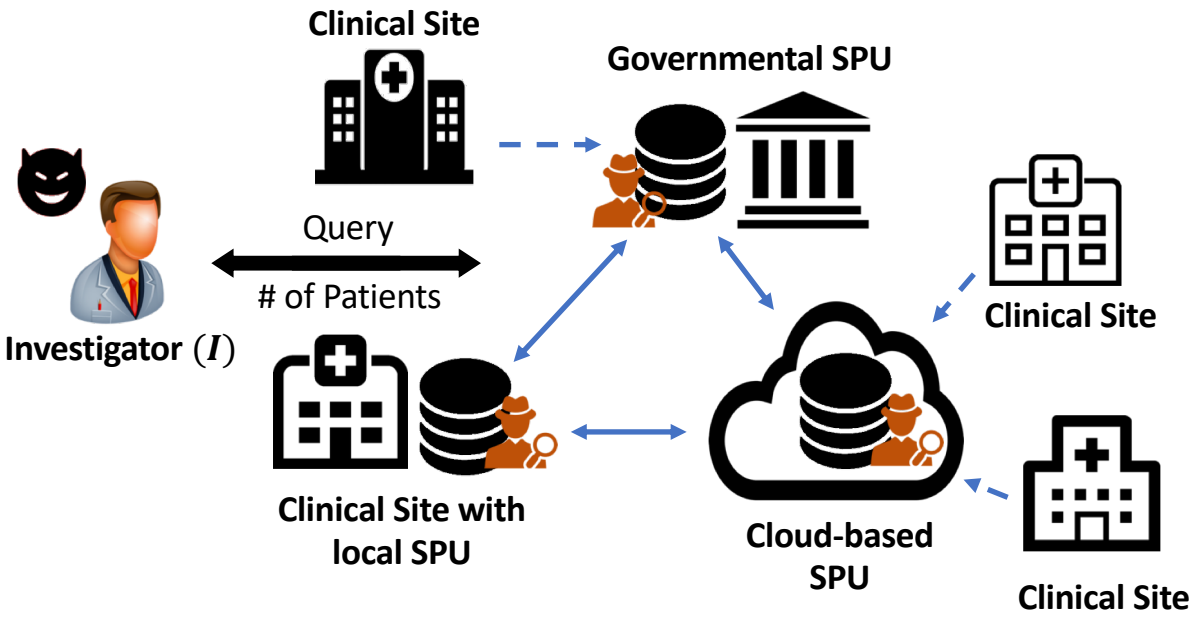



**Decentralized approach**

e.g., GA4GH Beacon, SHRINE, PCORNet

- High costs (human and technical)
- Low scalability (unsustainable on the long run)

# System and threat models




SPU: Storage and Processing Unit

**Honest-but-curious adversary:**

- honestly follows the protocol
- tries to infer sensitive data from the different steps of the protocol

**Malicious-but-covert adversary:**

- can tamper with the protocol
- tries to infer sensitive data from the query end-result

# Designing Secure Protocols

- Crucial step missing in the previous protocol design: “definition of secure”

- Components of any security definition:
  - Network Model
  - Adversarial Model
  - Expected Security Guarantees

- Without these components, it is impossible to reason about the security of a protocol

# Network Model

- Abstraction for the network
  - Message delivery
    - **Synchronous**: strict upper bound on the message propagation delay
    - **Asynchronous**: no such upper bound
  - Channel security
    - **Secret** and **authenticated** and party-to-party channels
    - **Open channels**
  - Topology
    **Fully** vs **partially** connected
- Access to “helper functionalities”
  - Authentication infrastructure
  - Common reference string (CRS): access to a common source of randomness
  - Broadcast channel
- It is common to assume a convenient setting first and gradually reduce assumptions

# Adversarial Model

- Adversary can be categorized based on multiple aspects:
  - Set of corrupted parties

    Typically a **threshold** $t$ on the number of corrupted parties
    Honest majority vs. no honest majority
  - Allowed adversarial behavior

    **Passive**: adversary that follows the protocol specification, only tries to infer information
    **Active**: adversary that deviates arbitrarily from the protocol
    **Covert**: adversary that can deviate from the protocol only if detection probability is low
  - Complexity

    **Probabilistic Polynomial-time**: computational model for secure computation
    **Quantum adversary**: has access to a quantum computer
    **Computationally unbounded**: information-theoretic model for secure computation
  - Corruption strategy:

    **Static corruption**: no change in the set of corrupted parties
    **Adaptive corruption**: adversary capable of corrupting parties during the computation
    **Proactive defense**: parties are corrupted only for a certain period of time

# Security Guarantees

- Typical set of requirements that should hold for a secure protocol:
  1) *Privacy*
     - No party should learn anything more than its prescribed output
  2) *Correctness*
     - Each party is guaranteed that the output that it receives is correct
  3) *Independence of Inputs*
     - Corrupted parties must choose their inputs independently of the honest parties' inputs
  4) *Guaranteed Output Delivery*
     - Corrupted parties should not be able to prevent honest parties from receiving their output
  5) *Fairness*
     - Corrupted parties should receive their outputs if and only if the honest parties also receive their outputs

- However:
  - How can we be sure it's exhaustive ?
  - They are subject to interpretation on a per-application basis
    - What is meant by "learning more than its own output" ? Ex. *SecureSum* with 2 players
    - What is a correct result ? Ex. Player lying on its inputs

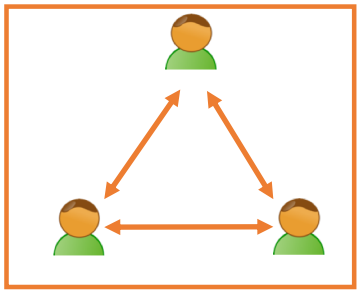
- We need a better way of formulating guarantees

# Ideal World vs. Real World

- We need a better **security definition**

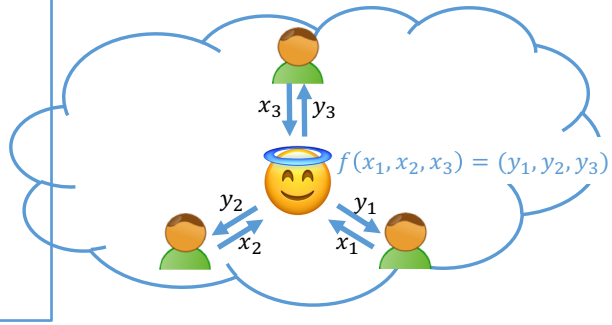
**Real world**

- No trusted third party exists
- **Real protocol**
  - Players freely interact together



**Ideal world**

- There exists an external party $P_{ideal}$ that:
  - Is trusted by all players
  - Is Incorruptible
  - Can carry out the computation
- **Ideal execution**
  1. Players send their inputs to $P_{ideal}$
  2. $P_{ideal}$ computes the outputs according to $f$
  3. $P_{ideal}$ sends each party its respective output




**Definition of security**: *a **real protocol** is said to be secure if it emulates the **ideal execution***

# Generalized Security Definition

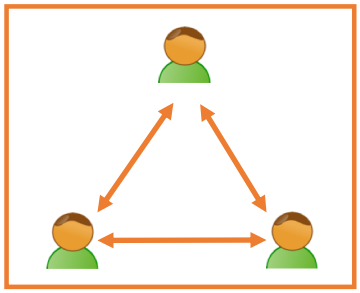
**Definition of security**: *a **real protocol** is said to be secure if it emulates the **ideal execution***

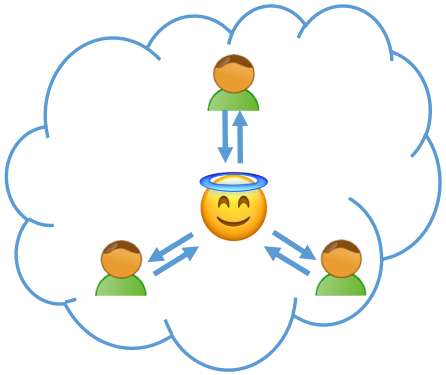
- **Alternative formulation**: *A real protocol is said to be secure if no adversary can do **more harm** in a real execution than in an ideal execution by a trusted third party*
- Implies all previous 5 security requirements in a general way while avoiding formulating them in an **ambiguous** way.

  Ex 1: Privacy Requirement for SecureSum with 2 player: even an ideal execution cannot protect inputs

  Ex 2: Correctness Requirement: even an ideal execution cannot prevent parties from lying

- Also provides a nice way of **abstracting** an MPC protocol to its **functionality**





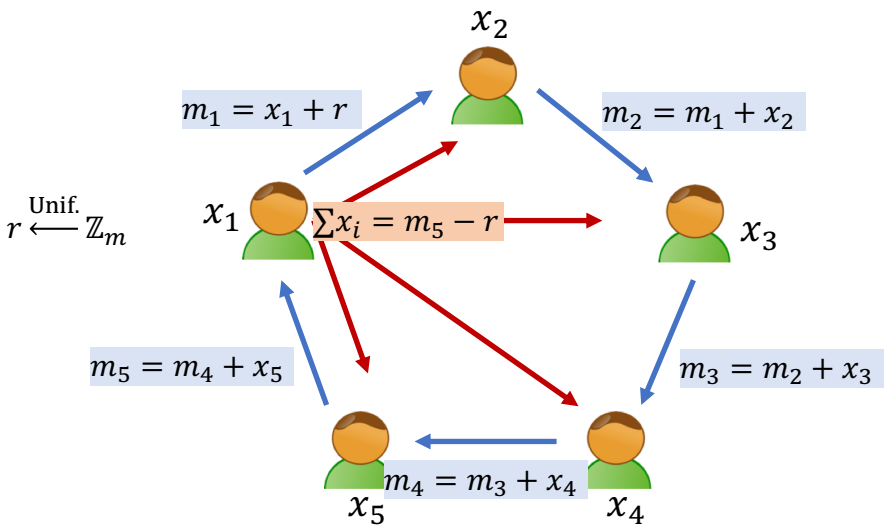
# Example: How much do 5 people collectively earn ?

Problem statement:

Players want to compute the sum of their salaries without revealing them

Realization:

- A first (bogus) attempt:
  - Assume $\sum x_i < m$
  - Consider inputs and +, - operation in $\mathbb{Z}_m$
  - The ***SecureSum*** protocol:



# Example: How much do 5 people collectively earn ?

Requirements

1. *Privacy*
   - No party should learn anything more than its prescribed output
2. *Correctness*
   - Each party is guaranteed that the output that it receives is correct

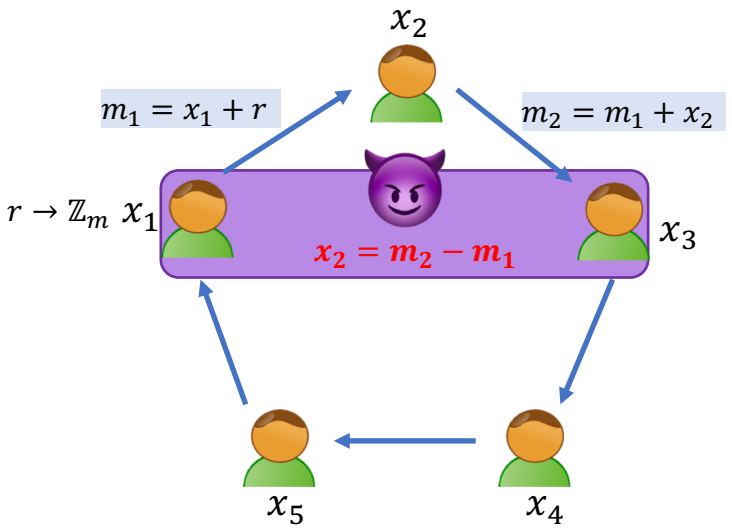
Security Concerns

- What if two players collude?
  - Privacy requirement can break
- What can the attacker do after breaking privacy?
  - Change its own input
  - Choose not to participate
  - ???

  → It's hard predict what the adversary will do !

- Fix the protocol ?
  - We have found one security flaw, **how many other** remain ?
  - Iterative approach to security: Find flaw, fix flaw, repeat

    → software engineer do that, not cryptographers
  - Even if the protocol is correct: are the requirements exhaustive ?



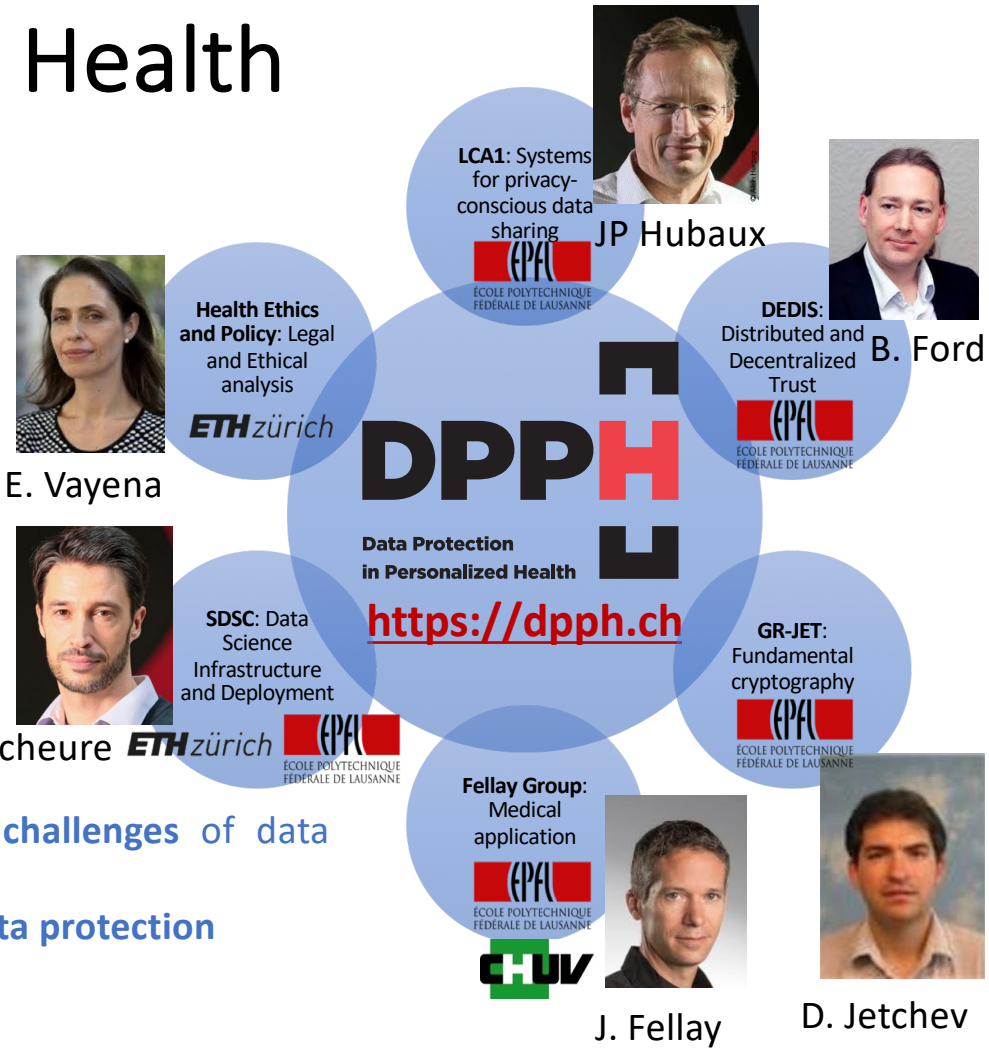
# More on Secure Multi-Party Computation (SMC)

- There exist operational tools, including SMC compilers
- Computational overhead can be an issue
- SMC can be implemented with homomorphic encryption
- Some companies active in this field:
  - Cybernetica (ShareMind product)
  - Dyadic (key management)
  - inpher (XOR for SMC and TFHE for homomorphic encryption)
  - Microsoft Research
  - Nebula Genomics

# DPPH – Data Protection in Personalized Health

- 5 research groups across the ETH domain + SDSC (Swiss Data Science Center)
- Funding: 3 Millions CHFrs
- Duration: 3 years (4/2018 - 3/2021)
- Funding Program: ETH PHRT (Personalized Health and Related Technologies)

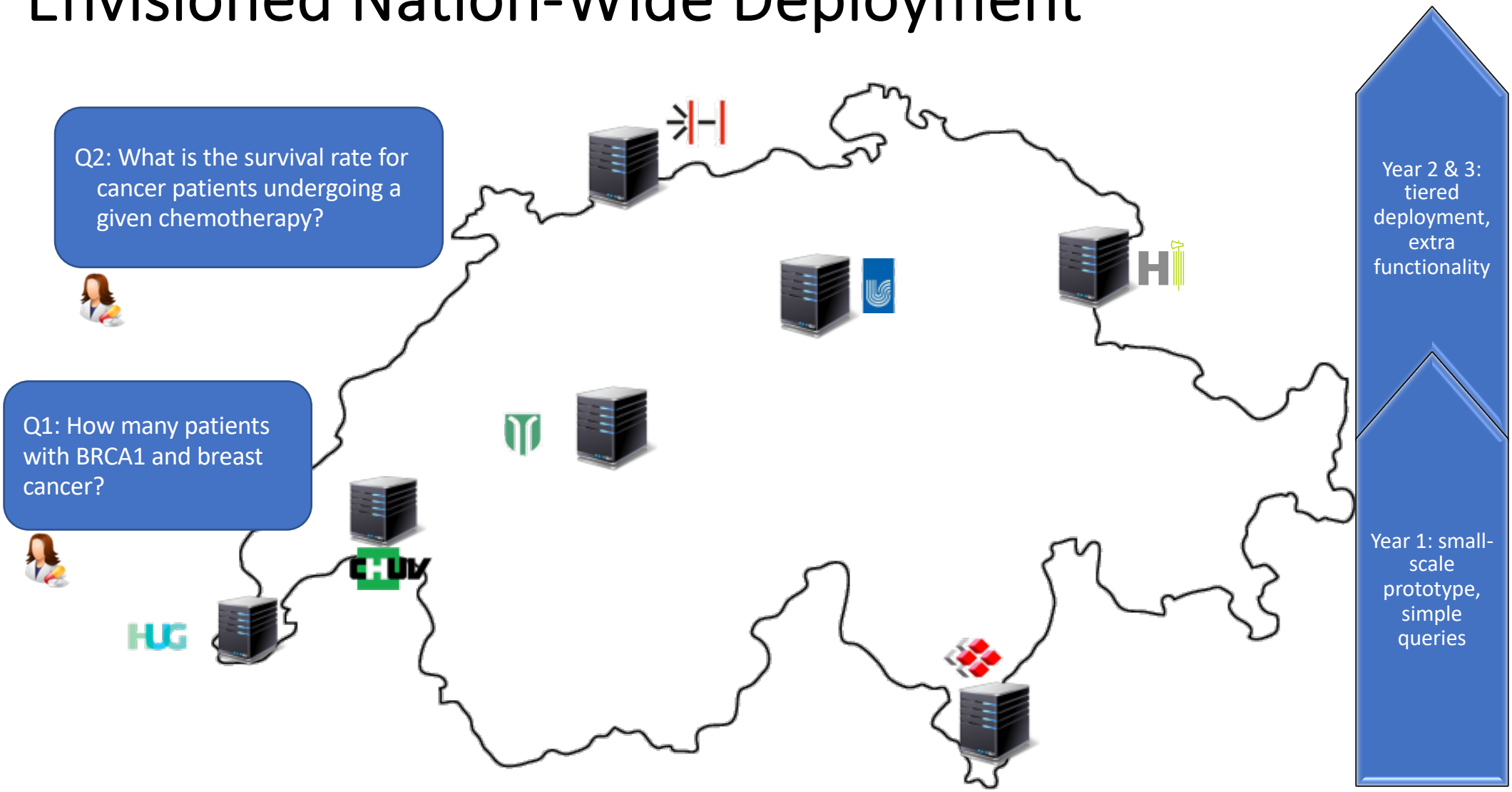
Strategic Focus Area
**Personalized Health and Related Technologies**




***Project goals***:

- Address the main **privacy, security, scalability, and ethical challenges** of data sharing for enabling effective P4 medicine
- Define an optimal **balance between usability, scalability and data protection**
- Deploy an appropriate set of **computing tools**

# Envisioned Nation-Wide Deployment



# Step 1: Secure and Decentralized Cohort Exploration