Secure Computation

CS489

Shin Yoo

Anonymity & Privacy

- Information is both treasure and liability: we want to make sure that what is seen and heard privately does not become public without our consent (Kizza, Ethics in Computing)
- Anonymity and privacy is not only to be discussed as abstract ideas but also to be implemented at the technical level.

Anonymity

- Being & remaining nameless
  - Pseudo-identity: an individual is identified by a certain pseudonym, code, or number.
  - Untraceable identity: an individual is not known by any name, including pseudo-names.
  - Anonymity with pseudo-address for correspondence: remailers and user groups.

Anonymous Remailers

- A server that receives messages with embedded instructions on where to send them next, and forwards them without revealing the sender (https://en.wikipedia.org/wiki/Anonymous_remailer)
- Pseudonym Remailer, Cyberpunk Remailer (Type I), Mixmaster Remailer (Type II), Mixminion Remailer (Type III)

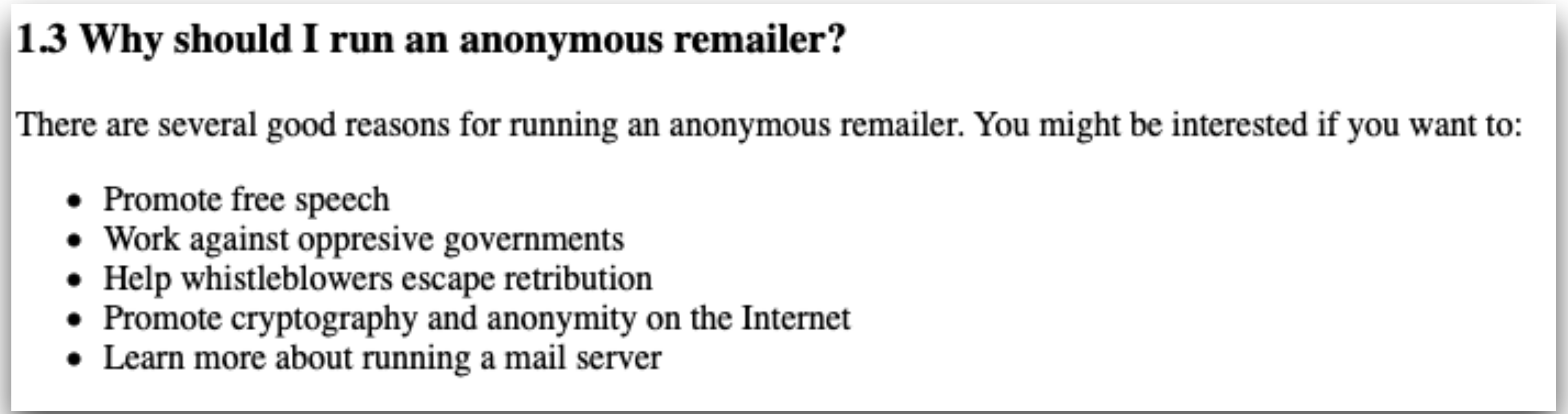
Pseudonym Remailer

- Assigns a pseudonym to each user; maintains a record of how to return messages to the real user.

- Type 1 (Cyberpunk): removes any identifying information from the PGP encrypted message and forwards it to the receiver; cannot reply unless reply address is included in the message body

- Type 2 (Mixmaster): message is relayed through MIX nodes to make it harder to trace (over SMTP)

- Type 3 (Mixminion): uses TLS to improve Type 2 (among other things)


1.3 Why should I run an anonymous remailer?

There are several good reasons for running an anonymous remailer. You might be interested if you want to:

- Promote free speech
- Work against oppresive governments
- Help whistleblowers escape retribution
- Promote cryptography and anonymity on the Internet
- Learn more about running a mail server

http://mixmaster.sourceforge.net/faq.shtml

Pros & Cons of Anonymity

- Good when a whistle-blower uses it for a good cause.
- Good in case underground spies gather information for national good.
- Good when there is intimidation and fear of reprisal.
- Bad when criminals use it for their advantage.
- Bad when anonymity hinders the resolution of some disputes.

Privacy

- Difficult to accurately define, as many aspects depend on culture, geographic location, political system, religious beliefs, etc.

- Durlak's four rights

  - Solitude: the right to be alone

  - Anonymity: the right to have no public personal identity

  - Intimacy: the right not to be monitored

  - Reserve: the right to control one's personal information

Technical Tools

- We will introduce three approaches:
  - k-anonymity: to anonymise a data set to reduce the probability of identifying an individual
  - Secure Multiparty Computation: to perform computation without revealing the input data
  - Homomorphic Encryption: to enable computation on encrypted data, without decrypting first

k-Anonymity

- Here, we follow the theoretical framework of Samarati 2001, “Protecting Respondents’ Identities in Microdata Release”, IEEE Transactions on Knowledge and Data Engineering, 13(6):1010-1027

Data Centric Applications (DCAs)

- Application based on non-trivial databases, some of which often contain personal information
- For privacy protection, there is a need to sanitise and anonymise the data before testing.
- However, there are multiple reports that only real world data can reveal certain faults.
- Clean-room approach is expensive and risky.

Medical Data Released as Anonymous

	SSN	Name	Race	DateOfBirth	Sex	ZIP	Marital Status	HealthProblem
			asian	09/27/64	female	94139	divorced	hypertension
			asian	09/30/64	female	94139	divorced	obesity
			asian	04/18/64	male	94139	married	chest pain
			asian	04/15/64	male	94139	married	obesity
			black	03/13/63	male	94138	married	hypertension
			black	03/18/63	male	94138	married	shortness of breath
			black	09/13/64	female	94141	married	shortness of breath
			black	09/07/64	female	94141	married	obesity
			white	05/14/61	male	94138	single	chest pain
			white	05/08/61	male	94138	single	obesity
•			white	09/15/61	female	94142	widow	shortness of breath

Voter List

	Name	Address	City	ZIP	DOB	Sex	Party	....
	....	....	....	....	....	....	....	....
•	Sue J. Carlson	900 Market St.	San Francisco	*94142*	*9/15/61*	*female*	democrat	....
	....	....	....	....	....	....	....	....

A linking attack: Table 1 has been anonymised by deleting SSN and Name from a medical dataset. However, joining this with the Voter registration list, we see that only one female born on 15th September 1961 lives in post code 94142.

Given a table $T\left(A_1, \ldots, A_n\right)$, a set of attributes

$$\left\{A_i, \ldots, A_j\right\} \subseteq \left\{A_1, \ldots, A_n\right\}$$

and a tuple $t \in T, t\left[A_i, \ldots, A_j\right]$ denotes the sequence of the values of $A_i, \ldots, A_j$ in $t, T\left[A_i, \ldots, A_j\right]$ denotes the projection, maintaining duplicate tuples, of attributes $A_i, \ldots, A_j$ in T. Also, $|T|$ denotes T' s cardinality, that is, the number of tuples in T.

Definition 1 (k-anonymity) . *Let $T(A_1, \ldots, A_n)$ be a table and QI be a quasi-identifier associated with it. T is said to satisfy k-anonymity w.r.t. QI iff each sequence of values in $T[QI]$ appears at least with k occurrences in $T[QI]$.*

Domain Generalisation

- Each attribute has its domain (names, age, address, etc). The set of concrete values is called the **ground**.
- We can generalise a domain: address can be generalised by dropping the street number, post code can be generalised by dropping the least significant digit, gender can be generalised by making all “human”, etc…
- If D_j is a generalisation of D_i, we write: $D_i \leq_D D_j$

$$\forall D_i, D_j, D_z \in \textbf{Dom} : D_i \leq_{\mathrm{D}} D_j, D_i \leq \mathrm{p}_z \Rightarrow$$
$$D_j \leq_{\mathrm{D}} D_z \vee D_z \leq_{\mathrm{D}} D_j$$

All maximal domains are singleton (i.e., everything is eventually generalised into a single thing).

Value Generalisation

- Concrete values also form a hierarchy.
- Value Generalisation Hierarchy (VGH) is a tree where, given $D_i \leq_D D_j$, a parent node is one of the values in D_j, and children nodes are more specific values in D_i that correspond to the parent.

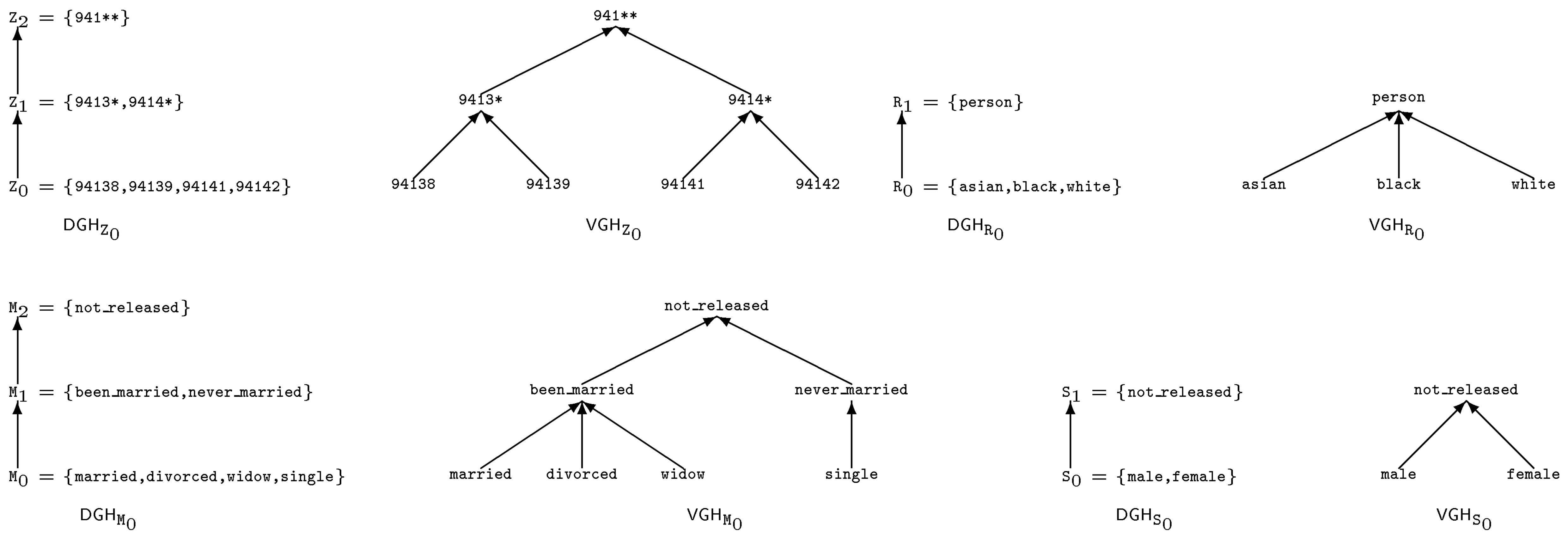


Fig. 2. Examples of domain and value generalization hierarchies.

Considering Multiple Domains

$$\begin{tikzcd}
\mathtt{Z}_2 = \{\mathtt{941{*}{*}}\} \\
\mathtt{Z}_1 = \{\mathtt{9413{*}, 9414{*}}\} \arrow[u] \\
\mathtt{Z}_0 = \{\mathtt{94138, 94139, 94141, 94142}\} \arrow[u]
\end{tikzcd}$$

$\mathsf{DGH}_{\mathtt{Z}_0}$

$$\begin{tikzcd}
 & & & \mathtt{941{*}{*}} & & & \\
 & \mathtt{9413{*}} \arrow[urr] & & & & \mathtt{9414{*}} \arrow[ull] & \\
\mathtt{94138} \arrow[ur] & & \mathtt{94139} \arrow[ul] & & \mathtt{94141} \arrow[ur] & & \mathtt{94142} \arrow[ul]
\end{tikzcd}$$

$\mathsf{VGH}_{\mathtt{Z}_0}$

$$\begin{tikzcd}
\mathtt{R}_1 = \{\mathtt{person}\} \\
\mathtt{R}_0 = \{\mathtt{asian, black, white}\} \arrow[u]
\end{tikzcd}$$

$\mathsf{DGH}_{\mathtt{R}_0}$

$$\begin{tikzcd}
 & \mathtt{person} & \\
\mathtt{asian} \arrow[ur] & \mathtt{black} \arrow[u] & \mathtt{white} \arrow[ul]
\end{tikzcd}$$

$\mathsf{VGH}_{\mathtt{R}_0}$

$$\begin{tikzcd}
 & \langle \mathtt{R}_1, \mathtt{Z}_2 \rangle & \\
\langle \mathtt{R}_1, \mathtt{Z}_1 \rangle \arrow[ur] & & \langle \mathtt{R}_0, \mathtt{Z}_2 \rangle \arrow[ul] \\
\langle \mathtt{R}_1, \mathtt{Z}_0 \rangle \arrow[u] & & \langle \mathtt{R}_0, \mathtt{Z}_1 \rangle \arrow[u] \arrow[ull] \\
 & \langle \mathtt{R}_0, \mathtt{Z}_0 \rangle \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

$\mathsf{DGH}_{\langle \mathtt{R}_0, \mathtt{Z}_0 \rangle}$

Generalisation Strategy

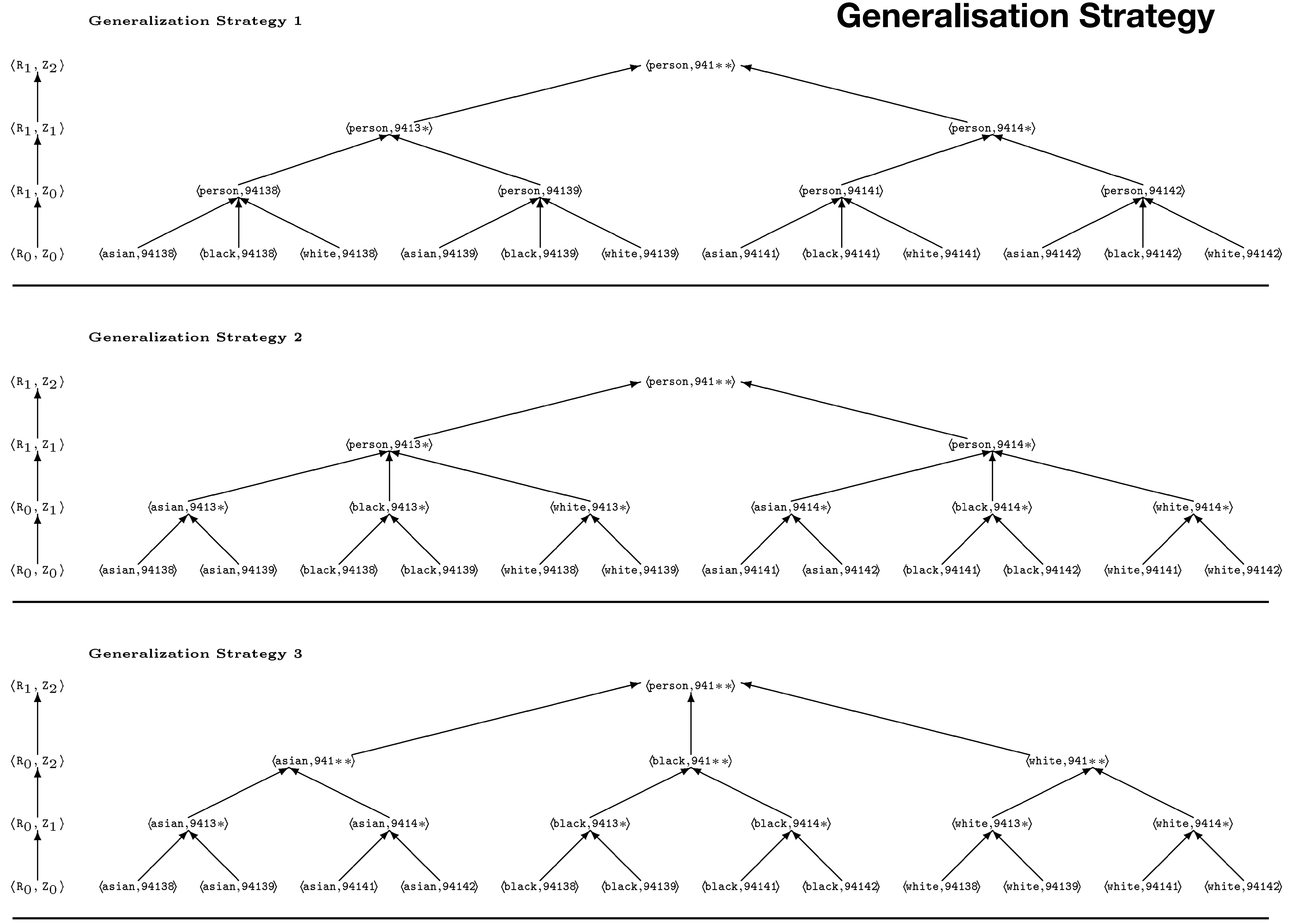


Fig. 3. Hierarchy $\mathrm{DGH}_{\langle \mathrm{R}_0,\mathrm{Z}_0\rangle}$ and corresponding domain and value generalization strategies.

Definition 1 (Generalized Table) *Let $T_i(A_1,\ldots,A_n)$ and $T_j(A_1,\ldots,A_n)$ be two tables defined on the same set of attributes. T_j is said to be a generalization of T_i, written $T_i \preceq T_j$, iff*

1. *$|T_i| = |T_j|$*
2. *$\forall A_z \in \{A_1,\ldots,A_n\} : dom(A_z,T_i) \leq_D dom(A_z,T_j)$*
3. *It is possible to define a bijective mapping between T_i and T_j that associates each tuple $t_i \in T_i$ with a tuple $t_j \in T_j$ such that $t_i[A_z] \leq_v t_j[A_z]$ for all $A_z \in \{A_1,\ldots,A_n\}$*

Race:$\mathtt{R}_0$	**ZIP**:$\mathtt{Z}_0$
asian	94138
asian	94139
asian	94141
asian	94142
black	94138
black	94139
black	94141
black	94142
white	94138
white	94139
white	94141
white	94142

PT

Race:$\mathtt{R}_1$	**ZIP**:$\mathtt{Z}_0$
person	94138
person	94139
person	94141
person	94142
person	94138
person	94139
person	94141
person	94142
person	94138
person	94139
person	94141
person	94142

$\mathsf{GT}_{[1,0]}$

Race:$\mathtt{R}_1$	**ZIP**:$\mathtt{Z}_1$
person	9413*
person	9413*
person	9414*
person	9414*
person	9413*
person	9413*
person	9414*
person	9414*
person	9413*
person	9413*
person	9414*
person	9414*

$\mathsf{GT}_{[1,1]}$

Race:$\mathtt{R}_0$	**ZIP**:$\mathtt{Z}_1$
asian	9413*
asian	9413*
asian	9414*
asian	9414*
black	9413*
black	9413*
black	9414*
black	9414*
white	9413*
white	9413*
white	9414*
white	9414*

$\mathsf{GT}_{[0,1]}$

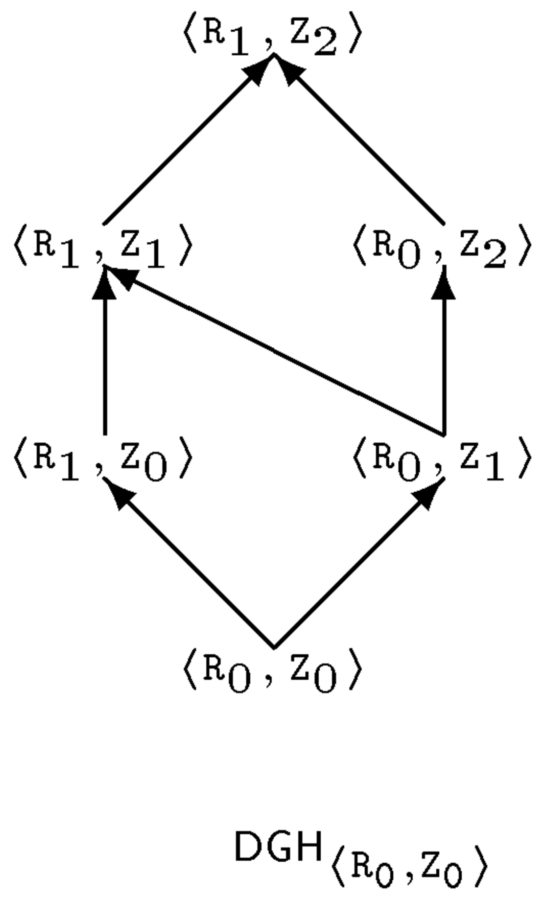
Race:$\mathtt{R}_0$	**ZIP**:$\mathtt{Z}_2$
asian	941**
asian	941**
asian	941**
asian	941**
black	941**
black	941**
black	941**
black	941**
white	941**
white	941**
white	941**
white	941**

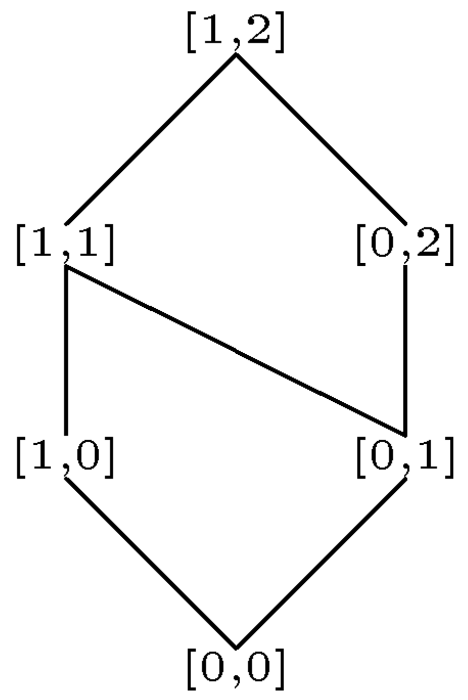
$\mathsf{GT}_{[0,2]}$

Race:$\mathtt{R}_1$	**ZIP**:$\mathtt{Z}_2$
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**
person	941**

$\mathsf{GT}_{[1,2]}$

Definition 1 (Distance Vector) *Let $T_i(A_1, \ldots, A_n)$ and $T_j(A_1, \ldots, A_n)$ be two tables such that $T_i \preceq T_j$. The distance vector of T_j from T_i is the vector $DV_{i,j} = [d_1, \ldots, d_n]$, where each $d_z, z = 1, \ldots, n$, is the length of the unique path between $D_z = \text{dom}(A_z, T_i)$ and $\text{dom}(A_z, T_j)$ in the domain generalisation hierarchy DHG_{Dz}.*





Given two distance vectors $DV = [d_1, \ldots, d_n]$ and $DV' = [d'_1, \ldots, d'_n]$, $DV \leq DV'$ iff $d_i \leq d'_i$ for all $i = 1, \ldots, n$. Moreover, $DV < DV'$ iff $DV \leq DV'$ and $DV \neq DV'$.

Definition 1 *(k-minimal generalization) Let* $T_i(A_1, \ldots, A_n)$ *and* $T_j(A_1, \ldots, A_n)$ *be two tables such that* $T_i \preceq T_j$. T_j *is said to be a k-minimal generalization of* T_i *iff:*

1. T_j *satisfies k-anonymity*

2. $\forall T_z : T_j \preceq T_z, T_z$ *satisfies k-anonymity* $\Rightarrow \neg(DV_{i,z} \leq DV_{i,j})$

Intuitively, you can also anonymise by suppressing data: “if only I could delete this row, I don’t need to generalise k-1 other rows!”

Definition 1 (Generalized Table-with Suppression) . *Let $T_i(A_1, \ldots, A_n)$ and $T_j(A_1, \ldots, A_n)$ be two tables defined on the same set of attributes. T_j is said to be a generalization of T_i written $T_i \preceq T_j$, iff:*

1. $|T_j| \leq |T_j|$
2. $\forall A_z \in \{A_1, \ldots, A_n\} : dom(A_z, T_i) \leq_D dom(Az, Tj)$
3. *It is possible to define an* injective *(sic.) mapping between T_i and T_j that associates each tuple $t_i \in T_i$ with a tuple $t_j \in T_j$ such that $t_i[A_z] \leq_v t_j[A_z]$ for all $A_z \in \{A_1, \ldots, A_n\}$*

Definition 1 (Minimal required suppression) *Let T_i be a table and T_j a generalisation of T_i satisfying k-anonymity. T_j is said to enforce minimal required suppression iff*

$$\forall T_z : T_i \preceq T_z, DV_{i,z} = DV_{i,j},$$

T_z satisfies k-anonymity $\Rightarrow |T_j| \geq |T_z|$

Find_vector

INPUT: Table $T_i = \mathsf{PT}[QI]$ to be generalized, anonymity requirement k, suppression threshold MaxSup, lattice VL_{DT} of the distance vectors corresponding to the domain generalization hierarchy DGH_{DT}, where DT is the tuples of the domains of the quasi-identifier attributes.

OUTPUT: The distance vector *sol* of a generalized table GT_{sol} that is a k-minimal generalization of $\mathsf{PT}[QI]$ according to Definition 4.3.

METHOD: Executes a binary search on VL_{DT} based on height of vectors in VL_{DT}.

```
1. $low$ :=0; $high$:=$height(\top, \mathsf{VL}_{DT})$; $sol := \top$
2. while $low < high$
   2.1 $try := \lfloor \frac{low+high}{2} \rfloor$
   2.2 $Vectors := \{vec \mid height(vec, \mathsf{VL}_{DT}) = try\}$
   2.3 $reach\_k$:= false
   2.4 while $Vectors \neq \emptyset \wedge reach\_k \neq$ true do
          Select and remove a vector $vec$ from $Vectors$
          if satisfies($vec$,$k$,$T_i$,MaxSup) then $sol$:= $vec$; $reach\_k$:= true
   2.5 if $reach\_k$ = true then $high$:= $try$ else $low$:= $try + 1$
3. Return $sol$
```

Fig. 10. An algorithm for computing a vector corresponding to a k-minimal generalization.

Secure Multiparty Computation

- Here, we mostly stick to the description in the following paper: J. Kim, M. G. Epitropakis, and S. Yoo. Learning without peeking: Secure multi-party computation genetic programming. In Proceedings of the 10th International Symposium on Search Based Software Engineering, SSBSE 2018, pages 246–261, 2018.
- There are useful references in the paper.

Secure Multiparty Computation

- Here, we examine the Garbled Circuit: there are other schemes.
- First, we will go through a very gentle introduction to RSA.
- Second, we will establish the concept of Oblivious Transfer.
- Finally, we will introduce the Garbled Circuit, with a case study.

RSA (Rivest-Shamir-Adleman)

- One of the first public-key encryption: encryption key is public, decryption key is kept secret.
- The asymmetry is based on the difficulty of factorisation of two large prime numbers.
- Let us go through a sketch of how it operates (we do not seek the full cryptographic knowledge here, although it would be preferable).

Intuition

- It is practical to find large positive integers, $e, d,$ and n, such that with modular exponentiation for all integers m (with $0 \leq m < n$):
  - $(m^e)^d \equiv m \pmod{n}$
  - But even after knowing e and n, or even m, it is extremely difficult to know d as well.

Key Generation

- Choose two prime numbers, p and q.
- Compute $n = pq$: n is later released as part of the public key.
- Compute $\lambda(n)$, Carmichael's totient function, which is the smallest integer $a^m \equiv 1(\mod n)$ for every integer a between 1 and n that is coprime to n.
  Since p, q are prime,$\lambda(n) = \lambda(p)\lambda(q) = (p-1)(q-1)$.
- Choose an integer e such that $1 < e < \lambda(n)$ and $gcd(e, \lambda(n)) = 1$.
  e forms the public key along with n.
- Determine $d \equiv e^{-1}(\mod \lambda(n))$. d is kept as the private key.

Encryption/Decryption

- Encryption
    - Turn the original message M into a large integer m such that $0 \leq m < n$.
    - $c \equiv m^e \pmod n$ (can be computed quickly)
- Decryption
    - $c^d \equiv (m^e)^d \equiv m \pmod n$

Yao’s Millionaire Problem

- Two millionaires want to decide who is the richer person.
- But they do not want to reveal how much money they have to the other.
- How do we do this?

Oblivious Transfer (OT)

- In cryptography, OT means a particular type of communication in which:
  - the sender sends multiple messages without knowing what was actually transferred, and
  - the receiver receives one of the sent messages without knowing what other messages were.

Oblivious Transfer (OT)

- Alice generates an RSA key pair and sends the public exponent e to Bob. The private exponent, d, is secret.

- Alice also generates and sends two random messages, x_0 and x_1, to Bob.

- Bob chooses $b \in \{0,1\}$, and generates a random k. Bob then sends $v = (x_b + ke) \mod n$ (i.e., encryption of k blind to x_b) to Alice.

- Alice computes $k_0 = (v - x_0)d \mod N$ and $k_1 = (v - x_1)d \mod N$. Alice knows k is one of these values, but does not know which.

- Alice sends $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$ to Bob.

- Bob decrypts m'_b because Bob knows which x_b was chosen earlier. Bob can only decrypt m'_b.

Garbled Circuit

- OT is message passing: how do we go from OT to computing generic functions?
- Answer: precompute all the answers, and transfer the table using OT 😨

Garbled Circuit for 2-input Boolean Gate

- Suppose Alice and Bob want to compute Boolean AND function using one input from each of them.
- Table (a) is the full truth table for AND gate.

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

(a)

Garbled Circuit for 2-input Boolean Gate

- Alice, the garbler, replaces 0s and 1s in the truth table with randomly generated string labels. The result is shown in (b).

a	b	c
X_0^a	X_0^b	X_0^c
X_0^a	X_1^b	X_0^c
X_1^a	X_0^b	X_0^c
X_1^a	X_1^b	X_1^c

(b)

Garbled Circuit for 2-input Boolean Gate

- Alice encrypts the output column(s) of the truth table with corresponding input labels. Alice also permutes the encrypted output rows so that the values cannot be guessed from the order (hence the name garbled).
- Alice sends the encrypted circuit to Bob, along with her inputs. For example, if Alice's input for a is 1, Alice sends X_1^a. Since Alice generated the labels randomly, Bob does not know what Alice's actual input is.

Garbled Table
$Enc_{X_0^a, X_0^b}(X_0^c)$
$Enc_{X_0^a, X_1^b}(X_0^c)$
$Enc_{X_1^a, X_0^b}(X_0^c)$
$Enc_{X_1^a, X_1^b}(X_1^c)$

(c)

Garbled Circuit for 2-input Boolean Gate

- In order to obtain the result, Bob needs the labels for his input.
- If Bob's input for b is 0, Bob asks Alice for $b = 0$ between X_0^b and X_1^b through 1-2 Oblivious Transfer, after which Alice does not know which Bob chose between X_0^b and X_1^b and Bob does not know what the other label (in our case X_1^b) is.

Garbled Table
$Enc_{X_0^a, X_0^b}(X_0^c)$
$Enc_{X_0^a, X_1^b}(X_0^c)$
$Enc_{X_1^a, X_0^b}(X_0^c)$
$Enc_{X_1^a, X_1^b}(X_1^c)$

(c)

Garbled Circuit for 2-input Boolean Gate

- Now Bob has both Alice's input, X_1^a, and his own input, X_0^b.
- Bob attempts to decrypt each row, and can only succeed with one row.
- Alice can map the random string to the actual result.
- No one knows the other's input!

Garbled Table
$Enc_{X_0^a, X_0^b}(X_0^c)$
$Enc_{X_0^a, X_1^b}(X_0^c)$
$Enc_{X_1^a, X_0^b}(X_0^c)$
$Enc_{X_1^a, X_1^b}(X_1^c)$

(c)

Extending the GC to an arbitrary function

- Any arbitrary function can be converted into a large Boolean gate circuit.
- Alice and Bob can process one Boolean gate at a time to work their way through the complicated circuit.
- Cost of privacy: encryption & decryption, I/O cost from OT, as well as the conversion of a function to a Boolean circuit

Obliv-C

- A GCC wrapper and a Domain Specific Language (DSL) that allows easier implementation of GC in high level language.
    - `obliv`: a qualifier added to variables that need to remain oblivious.
    - `obliv if`: to prevent information leak from control flow, the body of `obliv if` will be always executed, regardless of how the branch predicate evaluates. When the predicate is false, the garbled circuit ensures that the values computed inside the block are simply ignored.

```c
#include <million.h>
#include <obliv.oh>

void millionaire (void *args) {
  ProtocolIO *io = args;
  obliv int a, b;
  obliv bool result = false;
  a = feedOblivInt (io->myinput, 1);
  b = feedOblivInt (io->myinput, 2);
  obliv if (a < b) result = true;
  revealOblivBool (&io->result, result, 0);
}
```

Figure 1: An `Obliv-C` program that implements Yao's Millionaires' Problem taken from Zahur et al. [37].

https://github.com/samee/obliv-c/

ML under GC

- We tried Genetic Programming (okay, not the most classical ML algorithm) under GC.
  - 2-party scenario: two sides have halves of data, and need to combine them to learn something useful, but do not want to share the data
  - 1-party scenario: data holder wants to learn something from the data without necessarily revealing the data

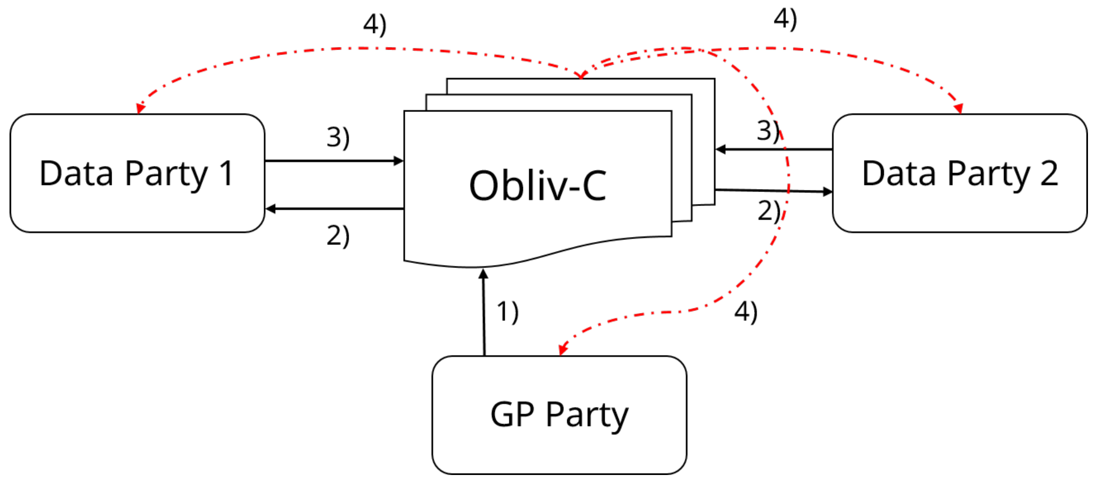


Figure 2: The multiparty dataholder scenario when there are two data parties and one GP party: 1) GP party generates the SMC program, and 2) sends it to each party. 3) Each party enters their input, and 4) the SMC program computes and all parties get the results.

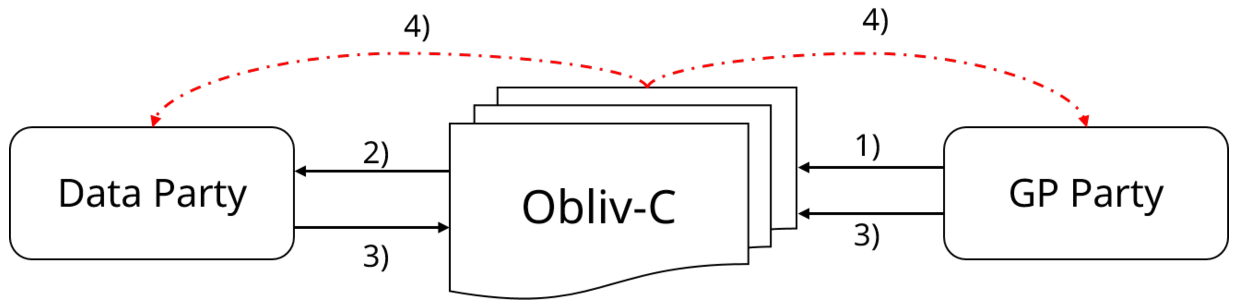


Figure 3: The singleparty dataholder scenario when there are one data party and one GP party: 1) GP party generates the SMC program, and 2) sends it to the data party. 3) The data party enters its input, whereas GP party enters nothing, and 4) the SMC program computes and all parties get the results.

It totally works. Computation is precise: in fact, we reported a bug in `obliv-c` implementation because only a specific operation differed from normal computation. But:

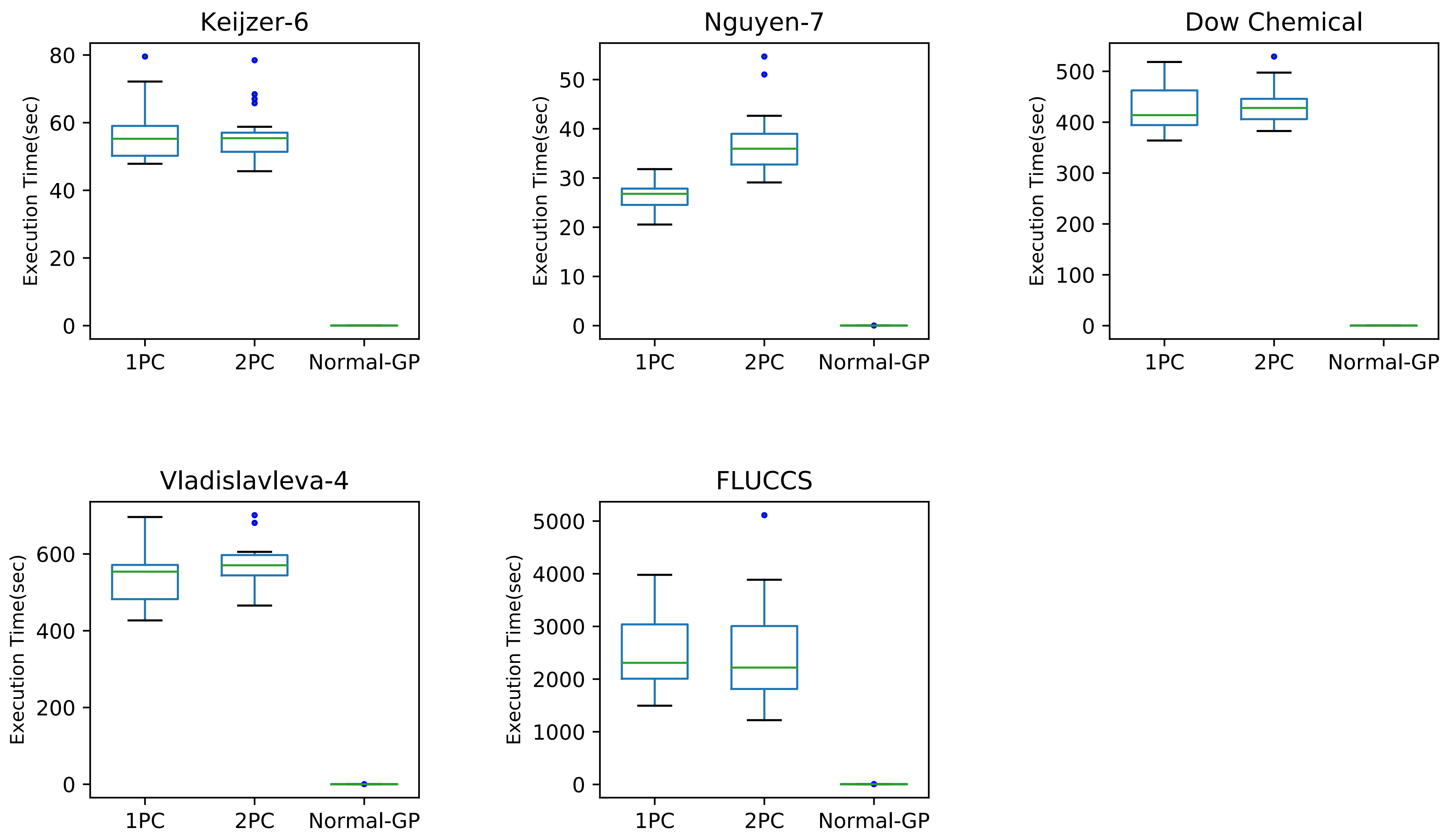


Figure 5: Boxplots of execution time by each subject.

Concluding Thoughts

- If you can do computation without revealing inputs, what will be possible?