FOSAD 2016

Privacy-preserving Information Sharing: Tools and Applications (Volume 1)

Emiliano De Cristofaro

University College London (UCL)

https://emilianodc.com

Prologue

Privacy-Enhancing Technologies (PETs):

- Increase privacy of users, groups, and/or organizations

PETs often respond to privacy threats

- Protect personally identifiable information
- Support anonymous communications
- Privacy-respecting data processing

Another angle: privacy as an enabler

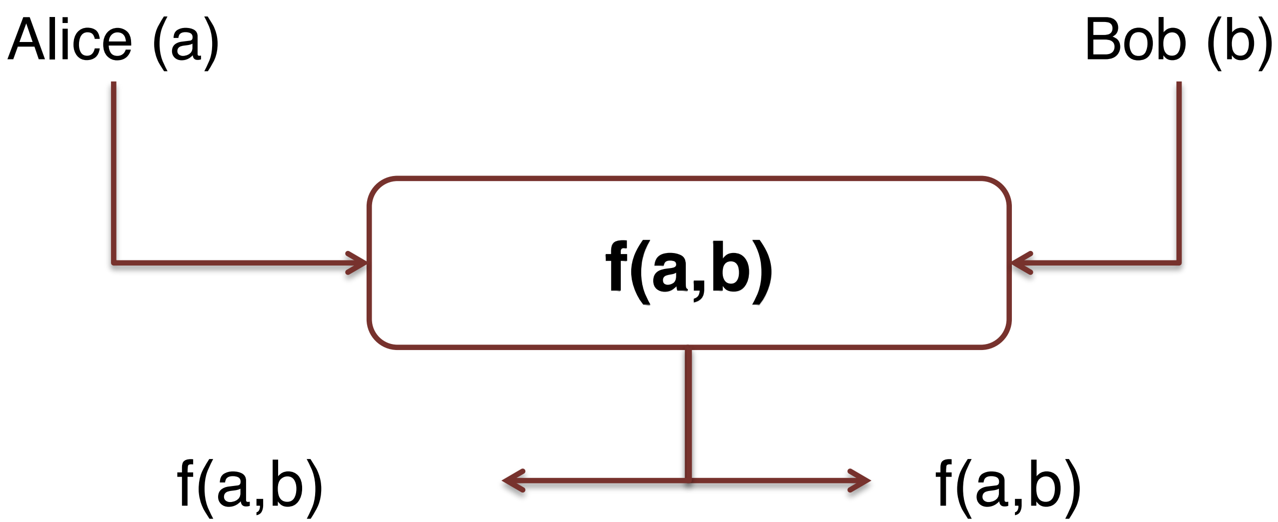
- Actively enabling scenarios otherwise impossible w/o clear privacy guarantees

Sharing Information w/ Privacy

Needed when parties with limited mutual trust willing or required to share information

- Only the required minimum amount of information should be disclosed in the process
- Relaxing the tension between the benefits of collaboration/ compliance and associated risks

Secure Computation (2PC)



Security in Secure Computation

Goldreich to the rescue!

Oded Goldreich. Foundations of cryptography: Basic Applications, Ch. 7.2. Cambridge Univ Press, 2004.

Computational indinguishability from an execution in the “ideal world”, involving a trusted third party (TTP)

Adversaries

Outside adversaries?

Not considered! Standard network security takes care of that

Honest but curious

Honest: follows protocol specifications, do not alter inputs

Curious: attempt to infer other party's input

Malicious

Arbitrary deviations from the protocol

Formalize/Prove Security (HbC)

The Ideal World/Real World Indistinguishability

Consider an ideal implementation where TTP receives inputs of both parties and outputs the result of the defined function

In the real implementation (without a TTP), each party **does not learn more information** than in the ideal one

→ Computational indistinguishability of views

With malicious adversaries, it is a bit more complicated ("simulation") > later

How to Implement 2PC?

1. Garbled Circuits

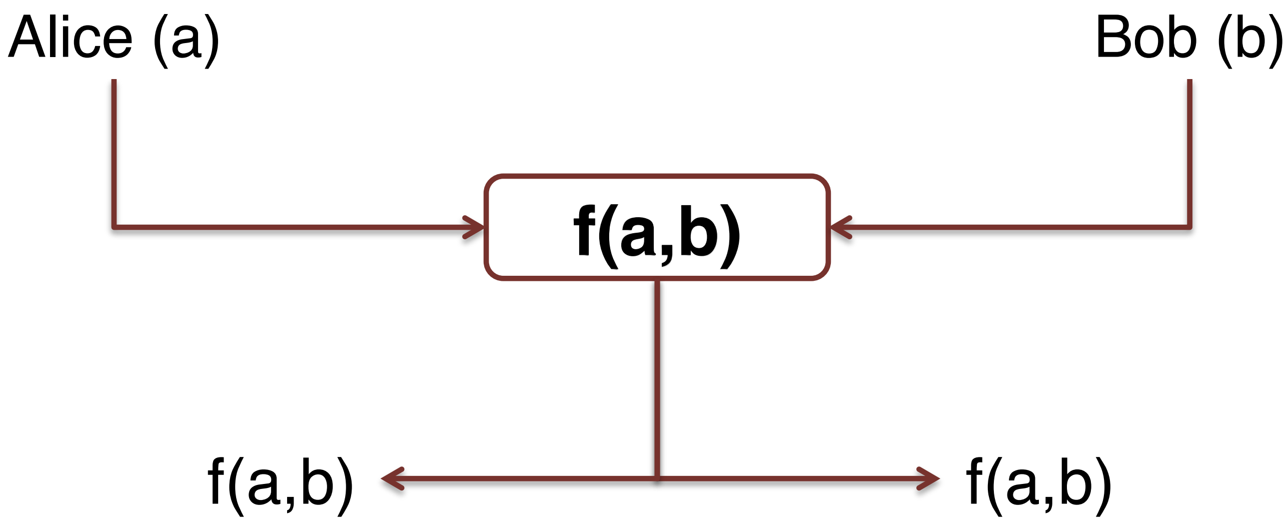
Sender prepares a “garbled” circuit and sends it to the receiver, who obliviously evaluates the circuit, learning the encodings corresponding to both his and the senders output

2. Special-Purpose Protocols

Implement one specific function (and only that)

Usually based on public-key crypto properties
[Have you ever heard of homomorphic encryption?]

Privacy-Preserving Information Sharing with 2PC?

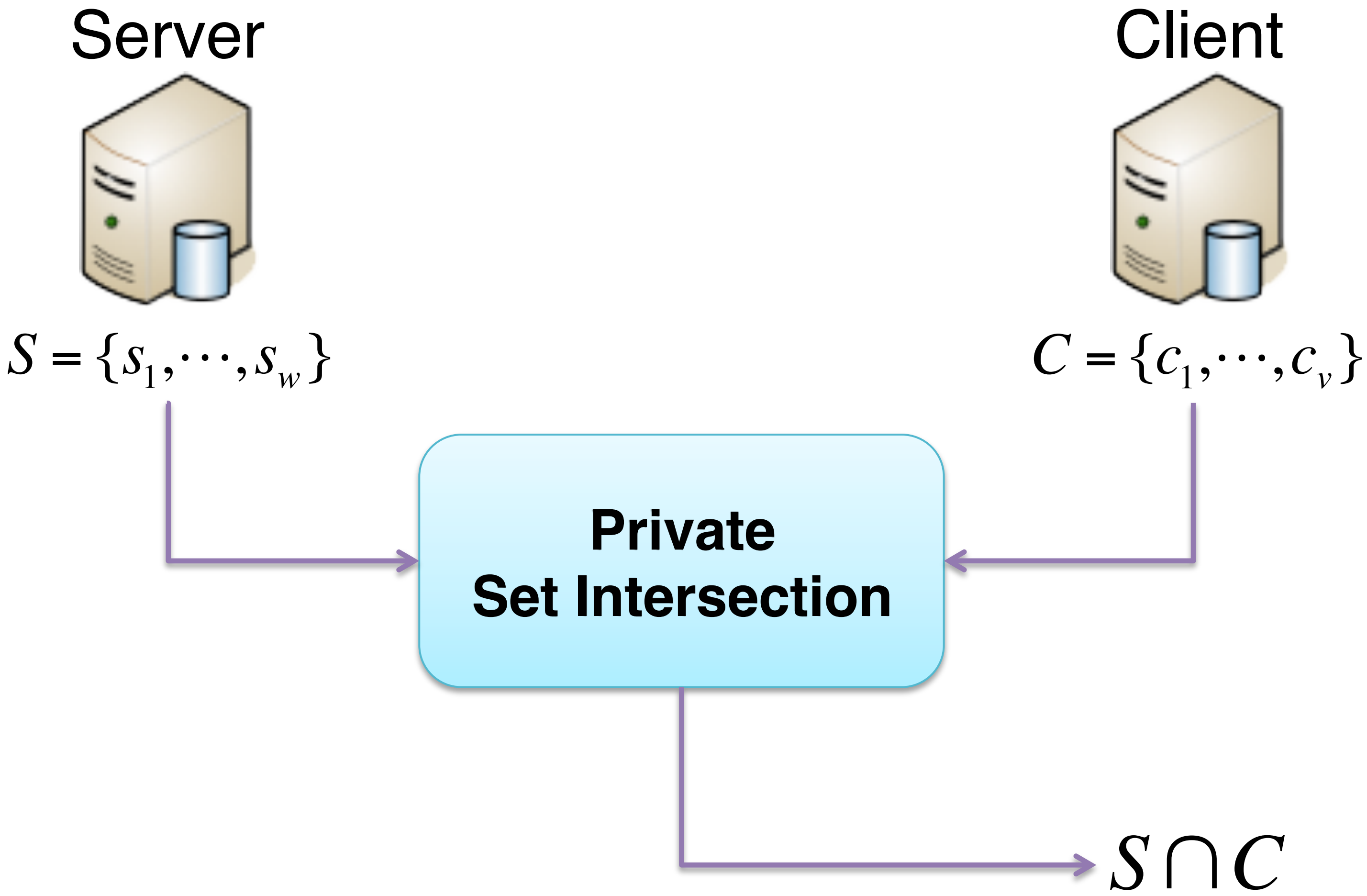


Map information sharing to f(·,·)?

Realize secure f(·,·) efficiently?

Quantify information disclosure from output of f(·,·)?

Private Set Intersection (PSI)



Private Set Intersection?

DHS (Terrorist Watch List) and **Airline** (Passenger List)

Find out whether any suspect is on a given flight

IRS (Tax Evaders) and **Swiss Bank** (Customers)

Discover if tax evaders have accounts at foreign banks

Hoag Hospital (Patients) and **SSA** (Social Security DB)

Patients with fake Social Security Number

Straightforward PSI

For each item s, the Server sends SHA-256(s)

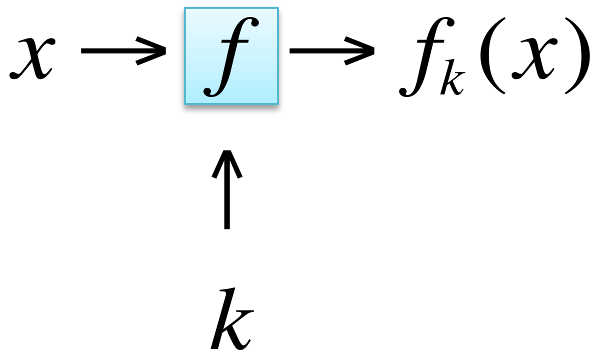
For each item c, the Client computes SHA-256(c)

Learn the intersection by matching SHA-256's outputs

What's the problem with this?

Background: Pseudorandom Functions

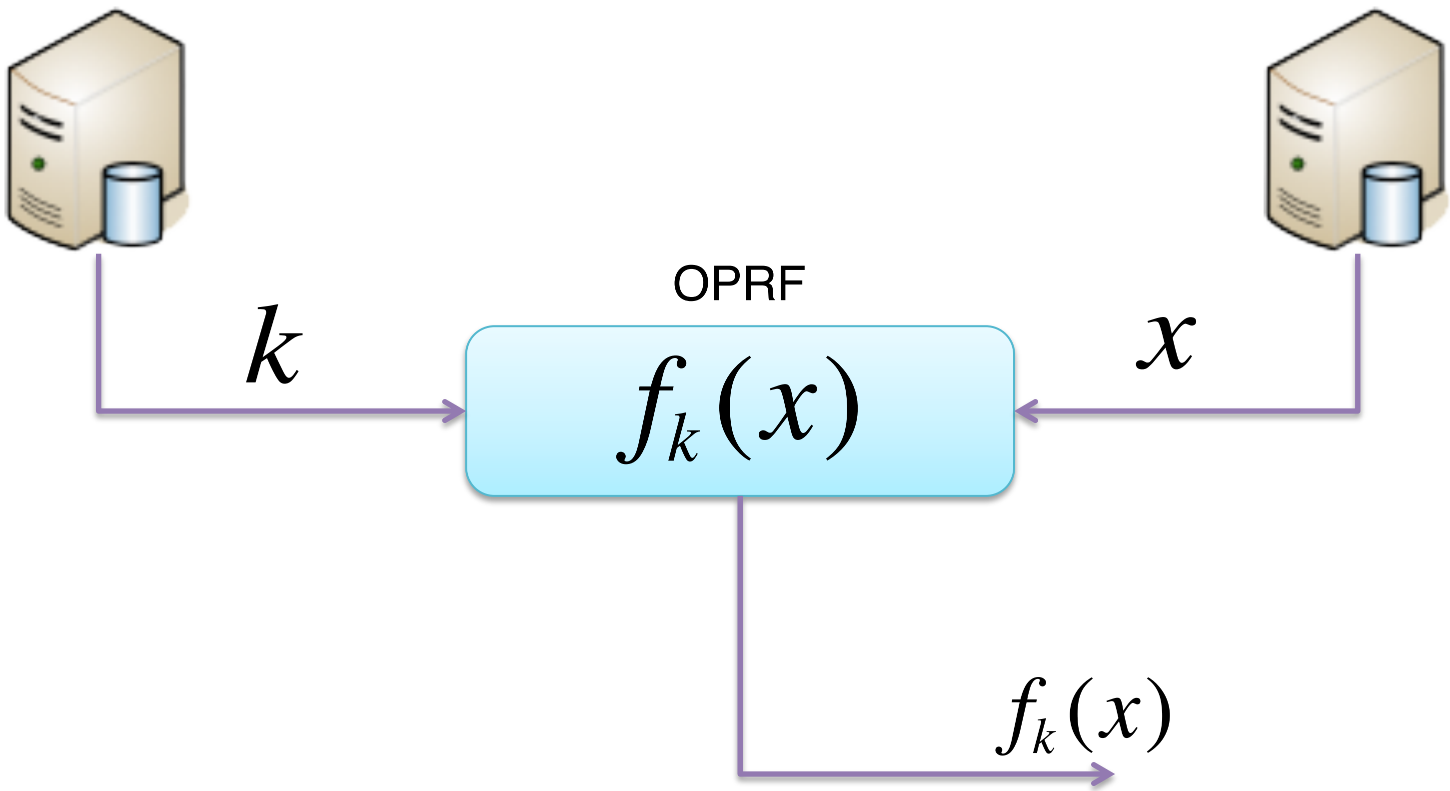
A **deterministic** function:



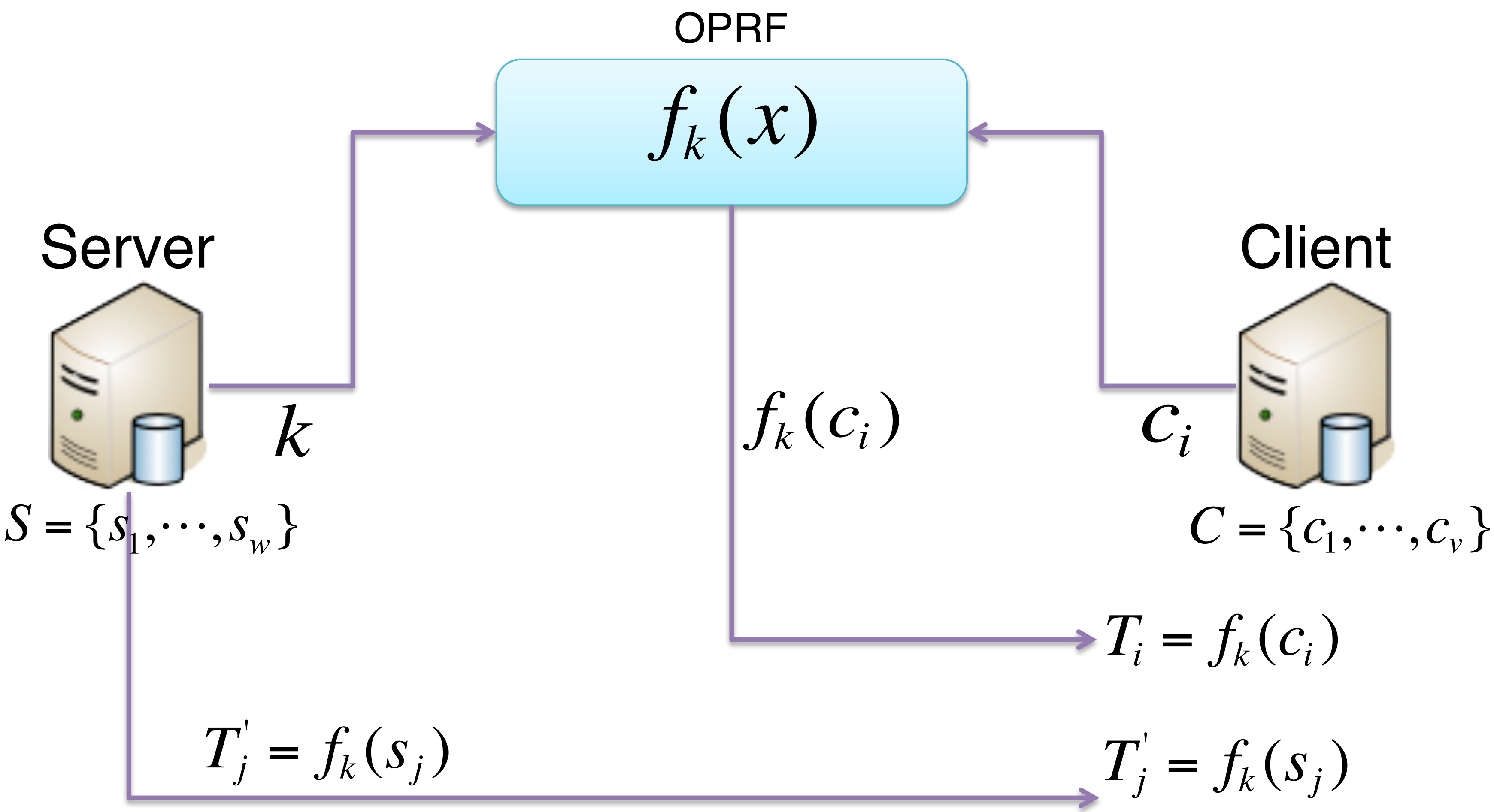
Efficient to compute

Outputs of the function “look” **random**

Oblivious PRF



OPRF-based PSI



Unless s_j is in the intersection T_j' looks random to the client

OPRF from Blind-RSA Signatures

RSA Signatures: $(N = p \cdot q, e), d \qquad e \cdot d \equiv 1 \bmod (p-1)(q-1)$

$$Sig_d(x) = H(x)^d \bmod N,$$

$$Ver(Sig(x), x) = 1 \Leftrightarrow Sig(x)^e = H(x) \bmod N$$

PRF: $f_d(x) = H(sig_d(x))$ (H one way function)

Server (d) | **Client (x)**

Client: $r \in Z_N$

Client → Server: $a = H(x) \cdot r^e$

Server → Client: $b = a^d$ $(= H(x)^d r^{\cancel{ed}})$

Client: $sig_d(x) = b / r$

Client: $f_d(x) = H(sig_d(x))$

PSI "Flavors"

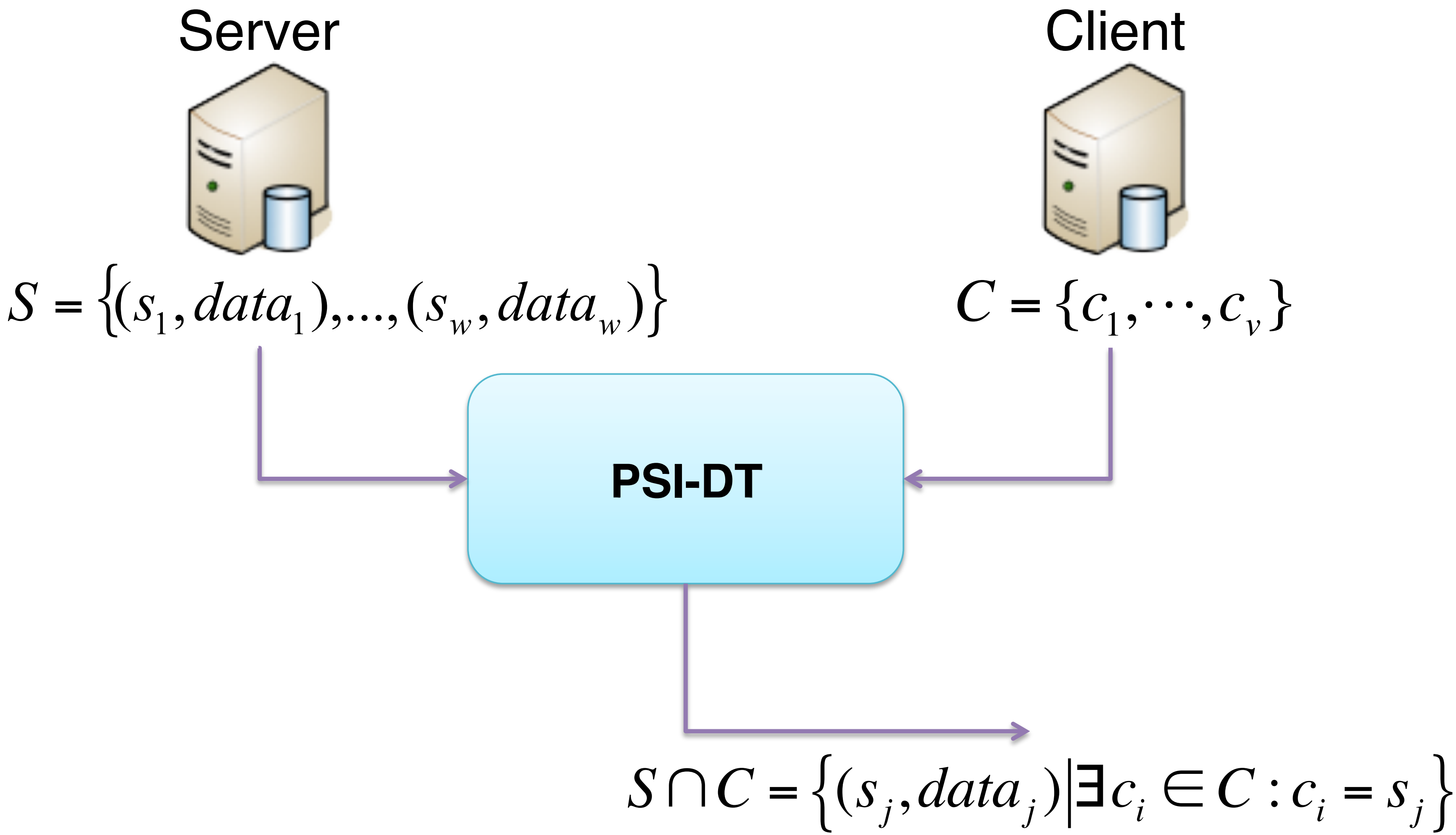
Honest-but-Curious (HbC) or Malicious Security?

HbC adversaries follow protocol specifications but try to violate privacy of other parties (passive)

Malicious adversaries can arbitrarily deviate (active)

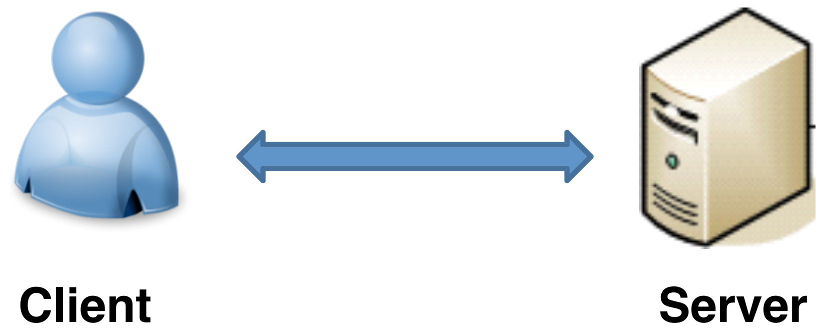
Cardinality only? Data Transfer?

PSI w/ Data Transfer (PSI-DT)



PSI w/ Data Transfer

SELECT * FROM DB WHERE $(attr_1^* = val_1^*$ OR $\cdots$ OR $attr_v^* = val_v^*)$



$\mathbf{attr_1}$	$\mathbf{attr_2}$	$\cdots$	$\mathbf{attr_m}$	
val_{11}	val_{12}	$\cdots$	val_{1m}	R_1
val_{21}	val_{22}	$\cdots$	val_{2m}	R_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
val_{w1}	val_{w2}	$\cdots$	val_{wm}	R_w

DB

See: De Cristofaro, Lu, Tsudik, Efficient Techniques for Privacy-preserving Sharing of Sensitive Information, TRUST 2011

How can we build PSI-DT?

A closer look at PSI

Server

$S = \{s_1, \cdots, s_w\}$

Client

$C = \{c_1, \cdots, c_v\}$

Private Set Intersection

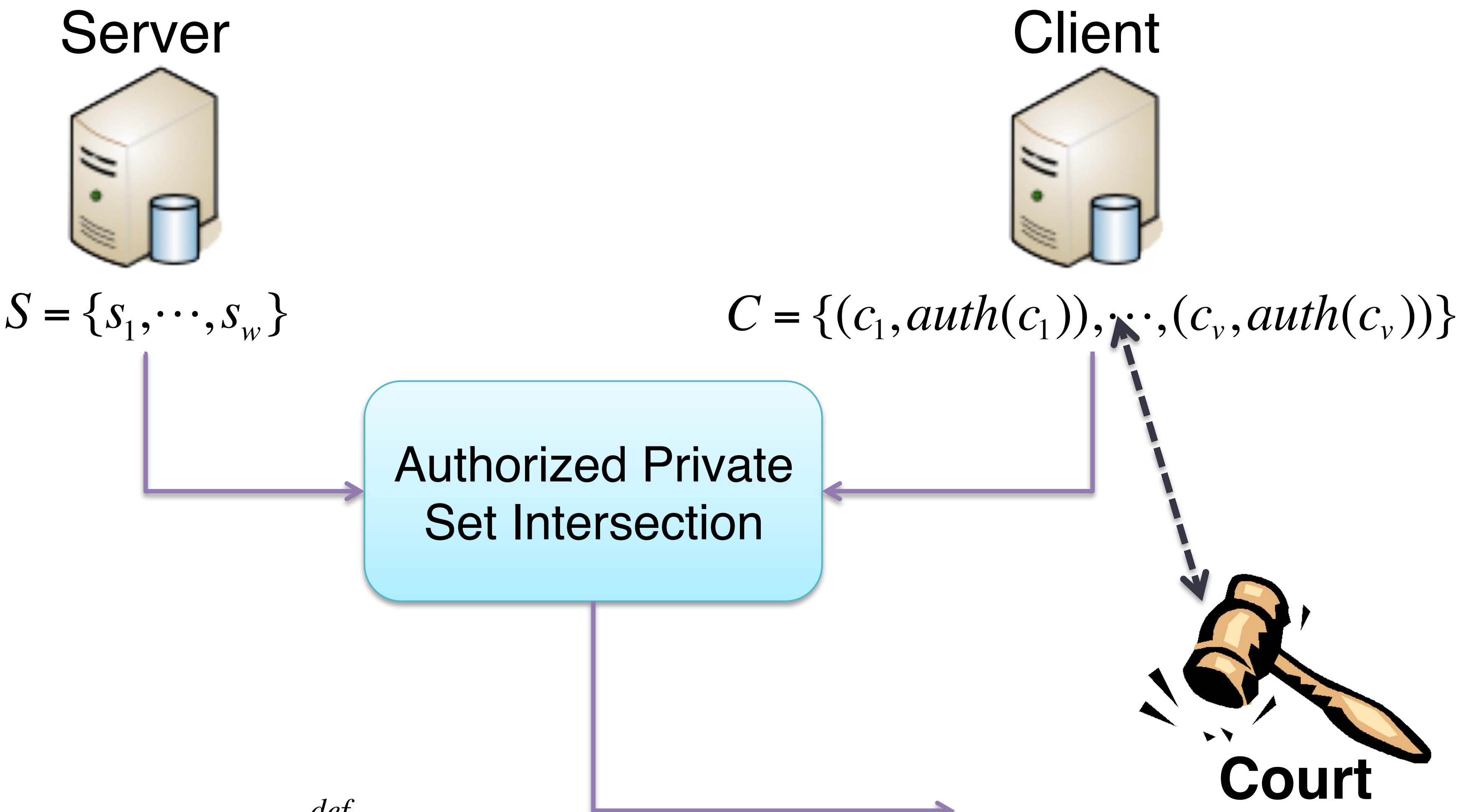
What if the client populates *C* with its best guesses for *S?*

Client needs to prove that inputs satisfy a policy or be authorized

Authorizations issued by appropriate authority

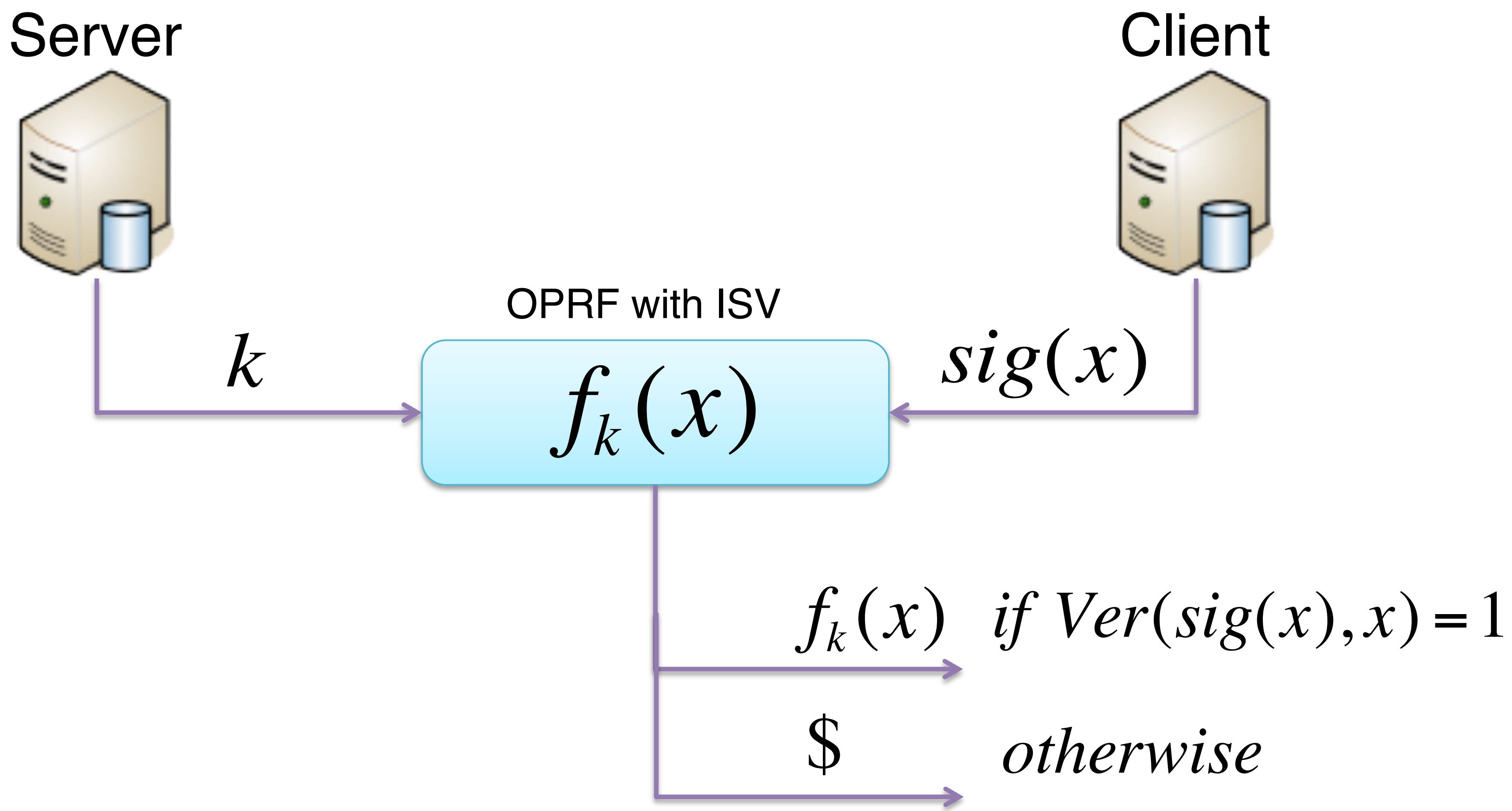
Authorizations need to be verified <u>implicitly</u>

Authorized Private Set Intersection (APSI)



$$S \cap C \overset{def}{=} \left\{ s_j \in S \middle| \exists c_i \in C : c_i = s_j \wedge auth(c_i) \text{ is valid} \right\}$$

OPRF w/ Implicit Signature Verification



A simple OPRF-like with ISV

Court issues authorizations: $Sig(x) = H(x)^d \bmod N$

OPRF: $f_k(x) = F(H(x)^{2k} \bmod N)$

Server (k)

Client ($H(x)^d$)

Client: $r \in Z_N$

Client → Server: $a = H(x)^d g^r$

Server → Client: $b = a^{2ek}; g^k$

(Implicit Verification) $(b = H(x)^{2e\not{d}k} g^{2rek})$

Client: $H(x)^{2k} = b/g^{2erk}$

Client: $f_k(x) = F(H(x)^{2k})$

OPRF with ISV – Malicious Security

OPRF: $f_k(x) = F(H(x)^{2k})$

Server (k)

Client ($H(x)^d$)

$r \in Z_N$

$a = H(x)^d g^r$ $\quad$ $\alpha = H(x)(g')^r$

$\pi = ZKPK\{r : a^{2e}/\alpha^2 = (g^e/g')^{2r}\}$

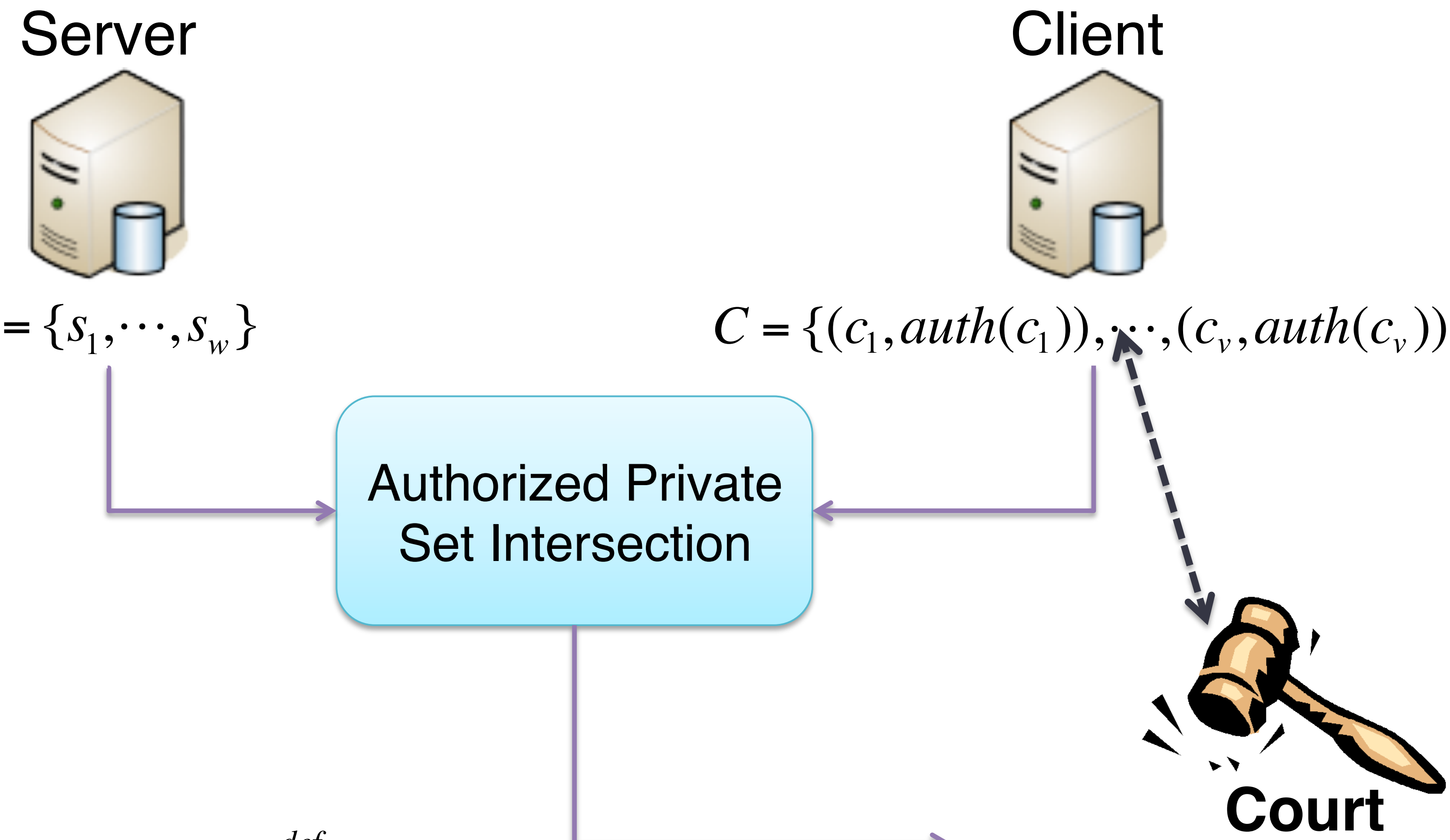
g^k $\quad$ $b = a^{2ek}$ $\quad$ $\pi' = ZKPK\{k : b = a^{2ek}\}$

$(b = H(x)^{2e\not{d}k} g^{2rek})$

$H(x)^{2k} = b/g^{2erk}$

$f_k(x) = F(H(x)^{2k})$

Authorized Private Set Intersection (APSI)



$S = \{s_1, \cdots, s_w\}$

$C = \{(c_1, auth(c_1)), \cdots, (c_v, auth(c_v))\}$

$$S \cap C \stackrel{def}{=} \left\{ s_j \in S \middle| \exists c_i \in C : c_i = s_j \wedge auth(c_i) \text{ is valid} \right\}$$

APSI: Preliminaries

Setup

Executed by the **Court**, on input sec. par. λ

(n,e,d) <- RSA.KeyGen(1^{λ}) on safe primes

Pick g, g' generators of QR_n

Select $H_1 : \{0,1\}^* \text{-->} Z_n$ (full-domain hash)

Select $H_2 : \{0,1\}^* \text{-->} \{0,1\}^{\lambda}$

Public parameters

n, e, g, g', $H_1()$, $H_2()$

Authorize

On item c_i, CA releases $\boldsymbol{\sigma_i = H(c_i)^d}$ mod n

Notation

Client has **v** items, $(c_1, \ldots, c_v)$ and c_i denotes i-th generic element

Server has **w** items, $(s_1, \ldots, s_w)$ and s_j denotes j-th generic element

$hs_j = H(s_j)$ $hc_i = H(c_i)$ $\sigma_i = (hc_i)^d$

APSI with linear complexity

SERVER
$(s_1, \ldots, s_w)$

If $hs_j = (\sigma_i)^e$ then $K_{S:j} = (hs_j)^{2Rs} = K_{c:i}$:

$K_{c:i} = M'_i \cdot Z^{-Rc:i} = M_i^{2eRs} \cdot g^{-Rc:i2eRs} =$
$= M_i^{2eRs} \cdot g^{-Rc:i2eRs} = \sigma_i^{2eRs} \cdot g^{2eRsRc:i} \cdot g^{-2eRsRc:i} =$
$= (hc_i)^{2Rs} = (hs_j)^{2Rs} = K_{S:j}$

LIENT
$\ldots_1, \sigma_1), \ldots, (c_v, \sigma_v))$

$'_i \leftarrow \{0,1\}$

$i \leftarrow Z_{N/2}$

$Rs \leftarrow Z_{N/2}$

$Z = g^{2eRs}$

$M_i = (-1)^{b_i} \cdot \sigma_i \cdot g^{Rc:i}$

$M'_i = (M_i)^{2eRs}$

$\{M_i, N_i\}$

$N_i = (-1)^{b'_i} \cdot hc_i \cdot g'^{Rc:i}$

$\leftarrow$

$K_{s:j} = (hs_j)^{2Rs}$

$ZKP_c = ZK\{ Rc:i \mid M_i^{2e}/N_i^2) = (g^e/g')^{2Rc:i} \}$

$T_{s:j} = H_2(K_{s:j}, hs_j, s_j)$

Client gets intersection C∩S:

c_i in C∩S if and only if

$T_{c:i}$ in $\{T_{c:1}, \ldots, T_{c:v}\} \cap \{T_{s:1}, \ldots, T_{s:w}\}$

$= M'_i \cdot Z^{-Rc:i}$

$T_{c:i} = H_2(K_{c:i}, hc_i, c_i)$

Common Input: n, e, g, g', $H_1()$, $H_2()$

Complexity

Input size:

Client's set contains **v** items

Server's set contains **w** items

Computational Complexity:

Client computes **O(v)** modular exponentiations

Server computes **O(w+v)** modular exponentiations

Exponentiations: 1024-bit mod 1024-bit

< 0.1ms on PC

~1ms on a smartphone

Communication Complexity:

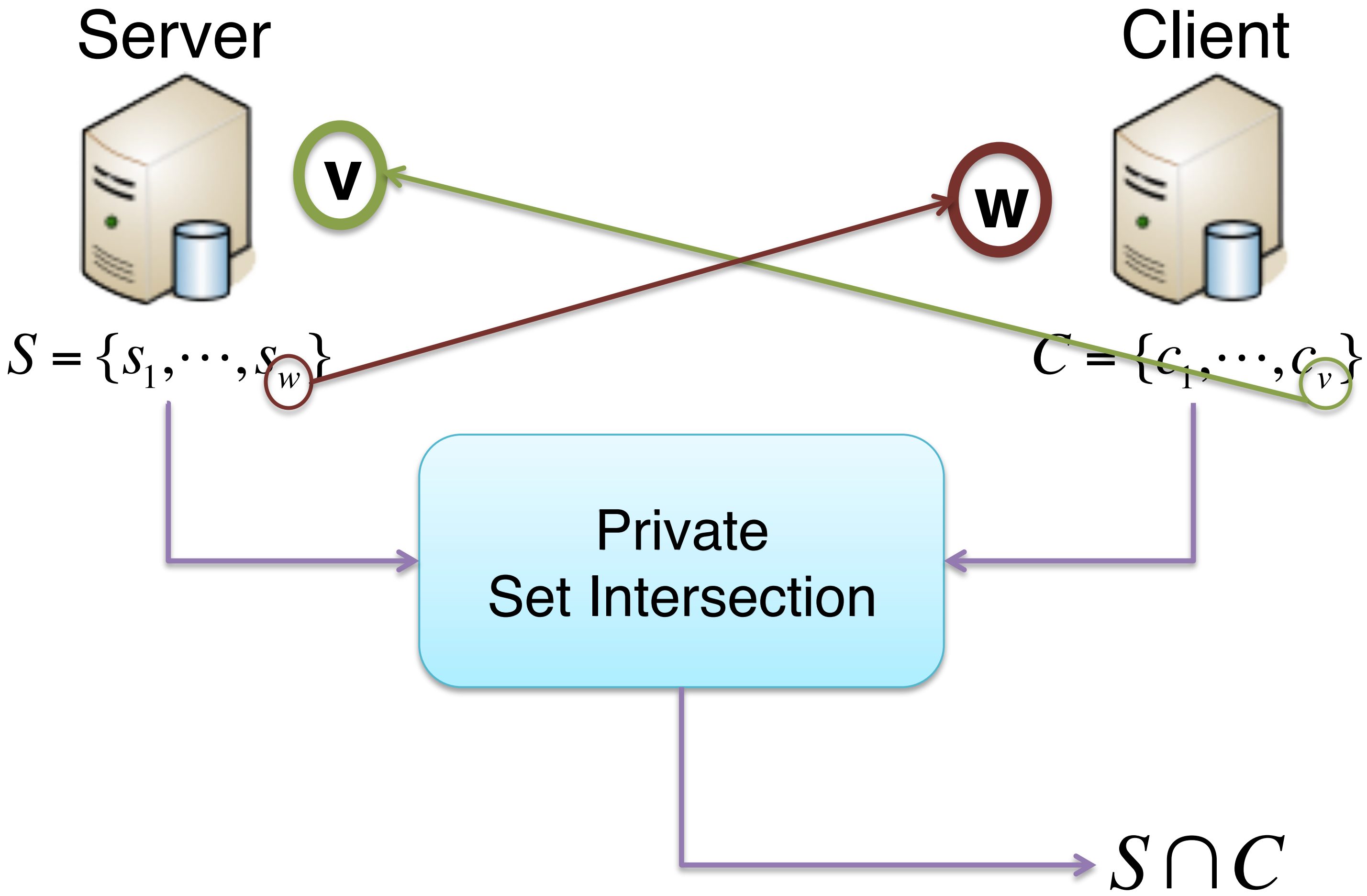
O(w+v)

Proofs in Malicious Model

Secure Computation of Authorized Set Intersection

- Use the Real World/Ideal World paradigm
- From a malicious client C*, construct an ideal world simulator SIM_C
  - SIM_C interacts with C* and **extracts C* inputs**
  - SIM_C interacts with the ideal-world server through a TTP to get the intersection
  - SIM_C plays (with C*) the role of the server on input the intersection
  - C*'s views when interacting with the simulator or in the real-world interaction are **indistinguishable** (show a reduction)
- From a malicious server S*, construct an ideal world simulator SIM_S
  - Similar idea but easier since the server has no output

Set Size in PSI



Why size matters?

DHS can't disclose the size of the **TWL**

TWL is dynamic: revealing its size leaks sensitive information

Fluctuations in set size may be even more sensitive

Ideally, the server's **workload** should be independent by client's input size

Feasibility of Size-Hiding

Run PSI with Random Padding?

Client chaffs up its set up to a fixed size

Upper bound would always be leaked

If client set is dynamic, the fixed size must reflect maximum possible set size: waste of computation and communication

Secure Two-Party Computation?

Input sizes are reciprocally known

Some feasibility results Lindell &Orlandi, Chase&Visconti, but require massive machinery (FHE, PCP)

SHI-PSI: The Building Blocks

RSA accumulator $g^{\prod_i x_i} \bmod N$

[Baric-Pfitzmann'97]

Unpredictable function $f_{p,q}(x,y) = x^{(1/y) \bmod \phi(N)} \bmod N$

Unpredictable if *p,q* are not known

Under the RSA assumption on safe moduli

Cannot invert in the exponent

SHI-PSI Intuition

The server selects $N=pq$

The client: (doesn't know p,q)

Compute a global witness for its set, X

An RSA accumulator on its (hashed) items

Hides client items (size too)

The server: (knows p,q)

Compute $f_{p,q}(X, s_j) = X^{1/H(s_j)}$

Apply a one-way function (a cryptographic hash)

The hash of an unpredictable function is a PRF (in ROM)

SH… col

Common Input: $N{=}pq, g, H(), F()$

Client
Input: $C = \{c_1, \ldots, c_i, \ldots c_v\}$

$$PCH \stackrel{def}{=} \prod_{i=1}^{v} hc_i$$

$$PCH_i \stackrel{def}{=} \prod_{l \neq i} hc_l \; (\forall i)$$

$$R_C \in_r \{1, \ldots, N^2\}$$

Server
Input: $S = \{s_1, \ldots, s_j, \ldots, s_w\}$
p, q

Client → Server:

$$X = \left(g^{PCH}\right)^{Rc} \bmod N$$

Server:

$$R_S \in_r \{0, \ldots, p'q'-1\}$$

$$\forall j : K_{s:j} = X^{R_S \cdot (1/hs_j)}$$

$$\forall j : T_{s:j} = F\left(K_{s:j}\right) \pmod N$$

Server → Client:

$$g^{R_S}, \{T_{s:1}, \ldots, T_{s:w}\}$$

Client:

$$\forall i : K_{c:i} = \left(g^{R_S}\right)^{R_C PCH_i}$$

$$\forall i : T_{c:i} = F\left(K_{c:i}\right) \pmod N$$

OUTPUT:

$$\{T_{c:1}, \ldots, T_{c:v}\} \cap \{T_{s:1}, \ldots, T_{s:w}\}$$

Correctness:

$$\forall c_i \in S \cap C, \exists j \; s.t. \; c_i = s_j \Rightarrow hc_i = hs_j$$

$$K_{c:i} = g^{R_S R_C \cdot PCH_i} = X^{R_S (1/hs_j)} = K_{s:j}$$

$$\Rightarrow T_{c:i} = T_{s:j}$$

-PSI: Complex

λ=length of H() outputs

v=|C| w=|S|

Client

Input: $C = \{c_1, \ldots, c_i, \ldots c_v\}$

$\forall i : PCH_i = \prod_{l \neq i} hc_l$

$PCH = \prod_{i=1}^{v} hc_i$

$R_C \in_r \{1, \ldots, N^2\}$

$X = \left(g^{PCH}\right)^{Rc} \bmod N$

v (λ)-bit exps

Server

Input: $S = \{s_1, \ldots, s_j, \ldots, s_w\}$

p,q

$R_S \in_r \{0, \ldots, p'q'-1\}$

$\forall j : K_{s:j} = X^{R_S \cdot (1/hs_j)}$

$\forall j : T_{s:j} = F\left(K_{s:j}\right)$

w |N|-bit exps

$g^{R_S}, \{T_{s:1}, \ldots, T_{s:w}\}$

1 |N|-bit exps

$\forall i : K_{c:i} = \left(g^{R_S}\right)^{R_C PCH_i}$

$\forall i : T_{c:i} = F\left(K_{c:i}\right)$

v*(v-1) (λ)-bit exps

Tree-based Optimization

O(vlog(v)) λ-bit exps

SHI-PSI: Security

Assumptions

Random Oracle Model (ROM)

Honest-but-Curious (HbC) adversaries

RSA assumption on safe moduli

Client Privacy: Indistinguishability

For every PPT S* that plays the role of the server, for every input set *S*, and for any client input set ($C^{(0)}$, $C^{(1)}$), two views of S* corresponding to client's inputs: $C^{(0)}$ and $C^{(1)}$ are computationally indistinguishable. **(Not even if $|C^{(0)}| \neq |C^{(1)}|$)**.

Server Privacy: Comparison to Ideal Model

Let $View_{Client}(C,S)$, be a random variable representing Client's view during execution of SHI-PSI with inputs *(C,S)*. There exists a PPT algorithm C* s.t.:

$$\left\{C^*(C, S \cap C)\right\}_{(C,S)} \equiv \left\{View_{Client}(C,S)\right\}_{(C,S)}$$

Special-purpose PSI

[**D**T10]: scales efficiently to very large sets

First protocol with linear complexities and fast crypto

[**D**KT10]: extends to arbitrarily malicious adversaries

Works also for Authorized Private Set Intersection

[**D**JLLT11]: PSI-based database querying

Won IARPA APP challenge, basis for IARPA SPAR

[**D**T12]: optimized toolkit for PSI

Privately intersect sets – 2,000 items/sec

[A**D**T11]: size-hiding PSI

Other Building Blocks

[**D**GT12]: Private Set Intersection Cardinality-only

[B**D**G12]: Private Sample Set Similarity

[**D**FT13]: Private and Size-Hiding Substring/Pattern Matching

[**D**JL11]: Private Database Querying