PRIVACY-PRESERVING SHARING OF SENSITIVE INFORMATION

(OR "PRIVATE SET INTERSECTION AND FRIENDS")

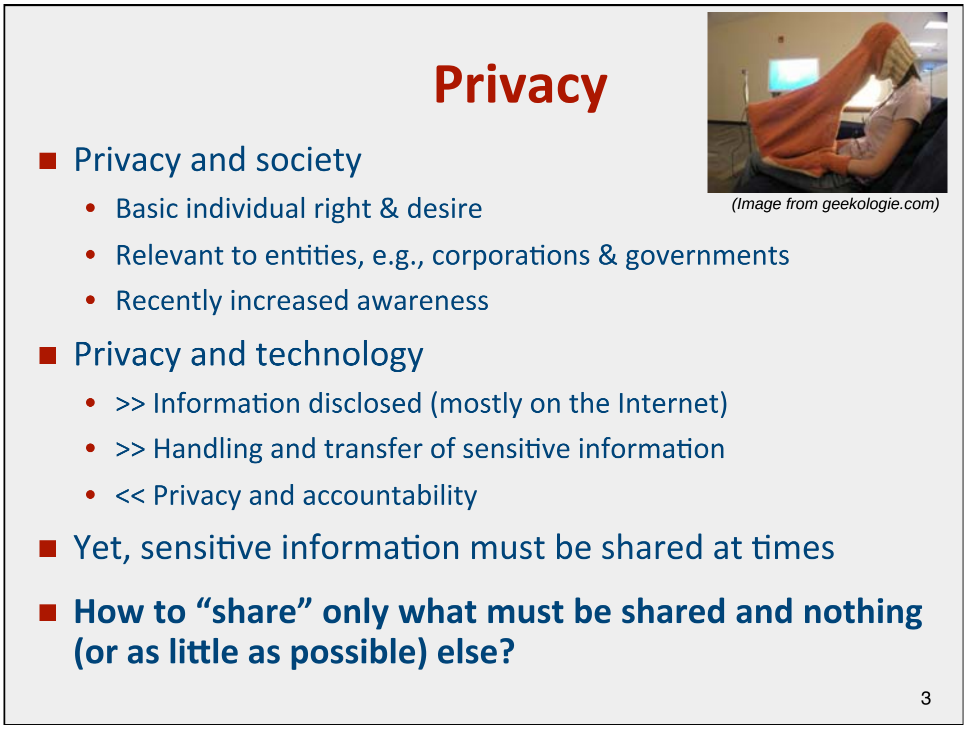
GENE TSUDIK

Computer Science Department, UC Irvine

http://sprout.ics.uci.edu

Outline

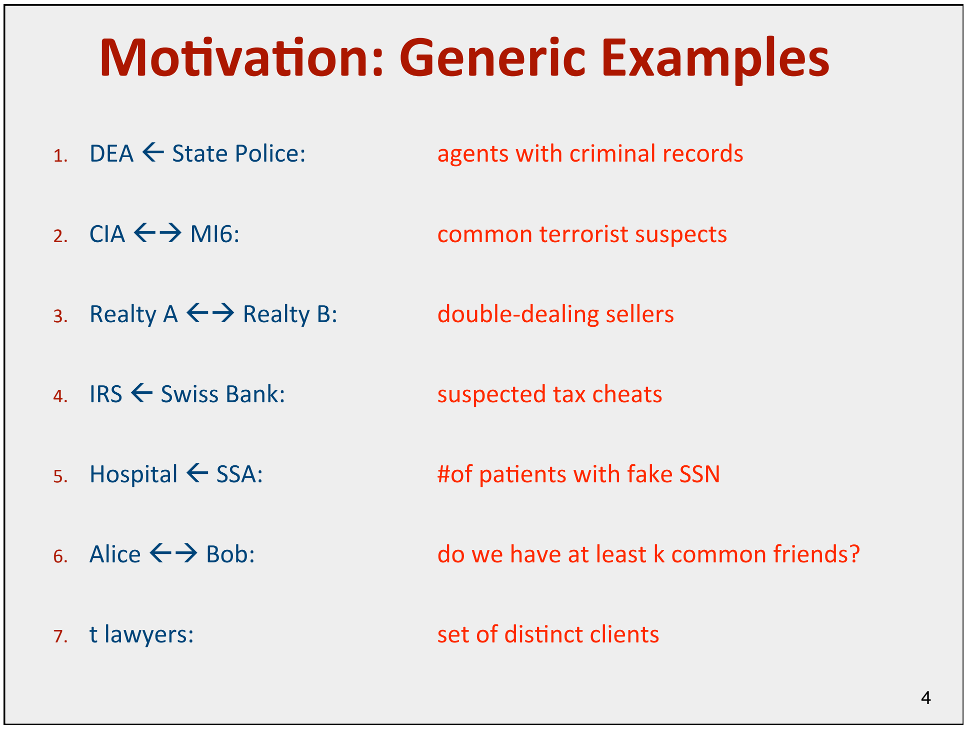
- Motivation
- PPIT: Privacy-Preserving Information Transfer
- (A)PSI: (Authorized) Private Set Intersection
- SHI-PSI: Size-Hiding Private Set Intersection
- System Issues
- On-going & Future Work



Privacy

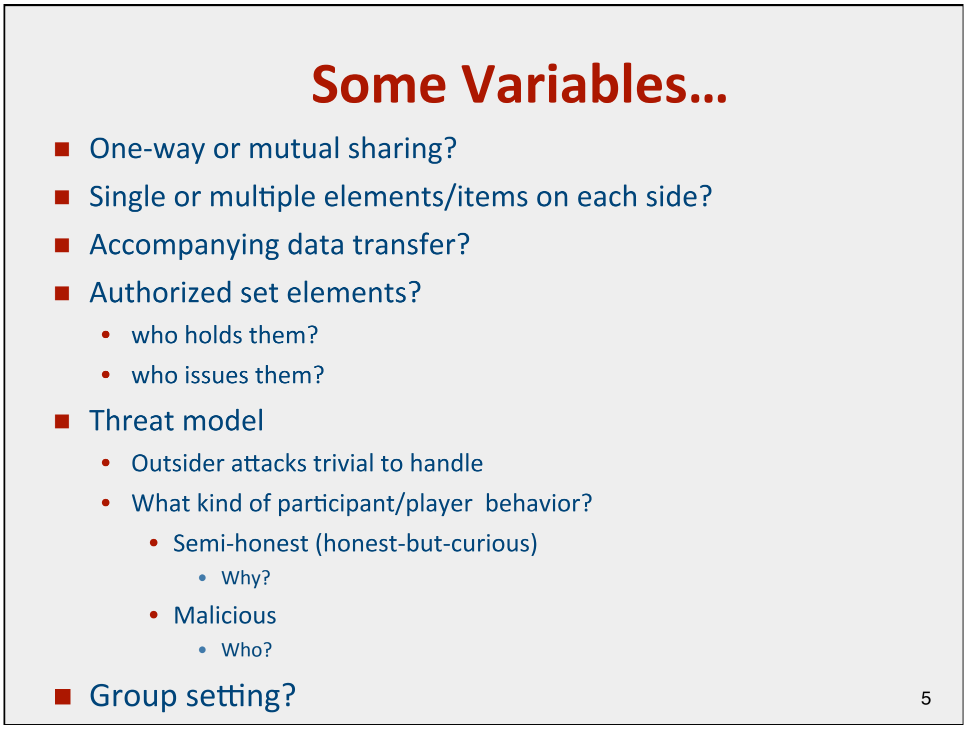
(Image from geekologie.com)

- Privacy and society
  - Basic individual right & desire
  - Relevant to entities, e.g., corporations & governments
  - Recently increased awareness
- Privacy and technology
  - >> Information disclosed (mostly on the Internet)
  - >> Handling and transfer of sensitive information
  - << Privacy and accountability
- Yet, sensitive information must be shared at times
- **How to “share” only what must be shared and nothing (or as little as possible) else?**



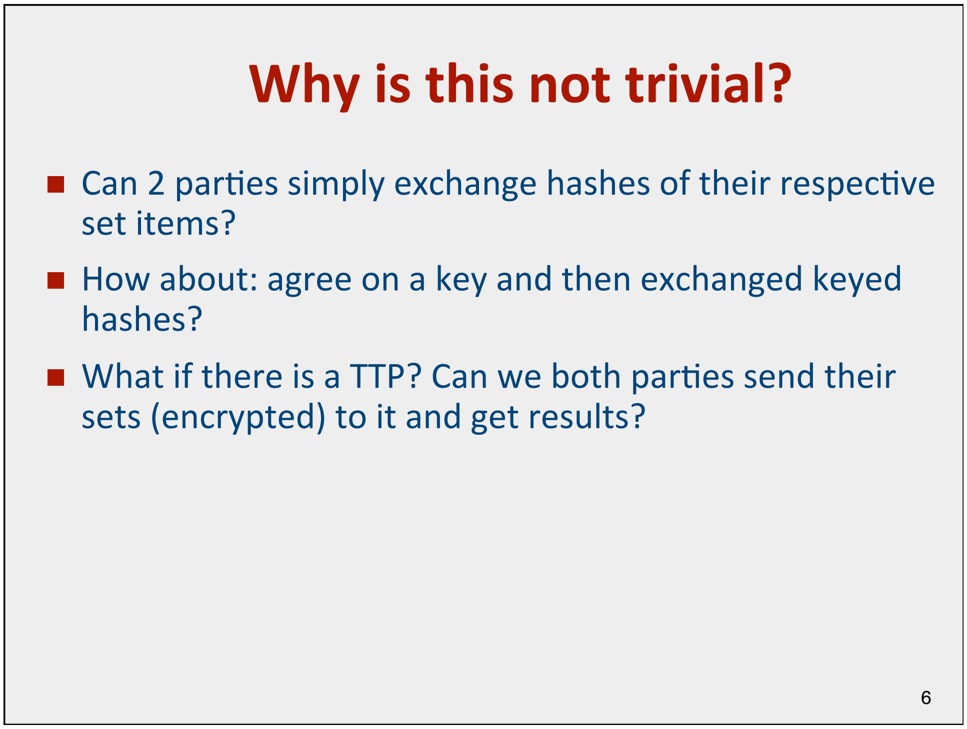
Motivation: Generic Examples

1.	DEA ← State Police:	agents with criminal records
2.	CIA ←→ MI6:	common terrorist suspects
3.	Realty A ←→ Realty B:	double-dealing sellers
4.	IRS ← Swiss Bank:	suspected tax cheats
5.	Hospital ← SSA:	#of patients with fake SSN
6.	Alice ←→ Bob:	do we have at least k common friends?
7.	t lawyers:	set of distinct clients



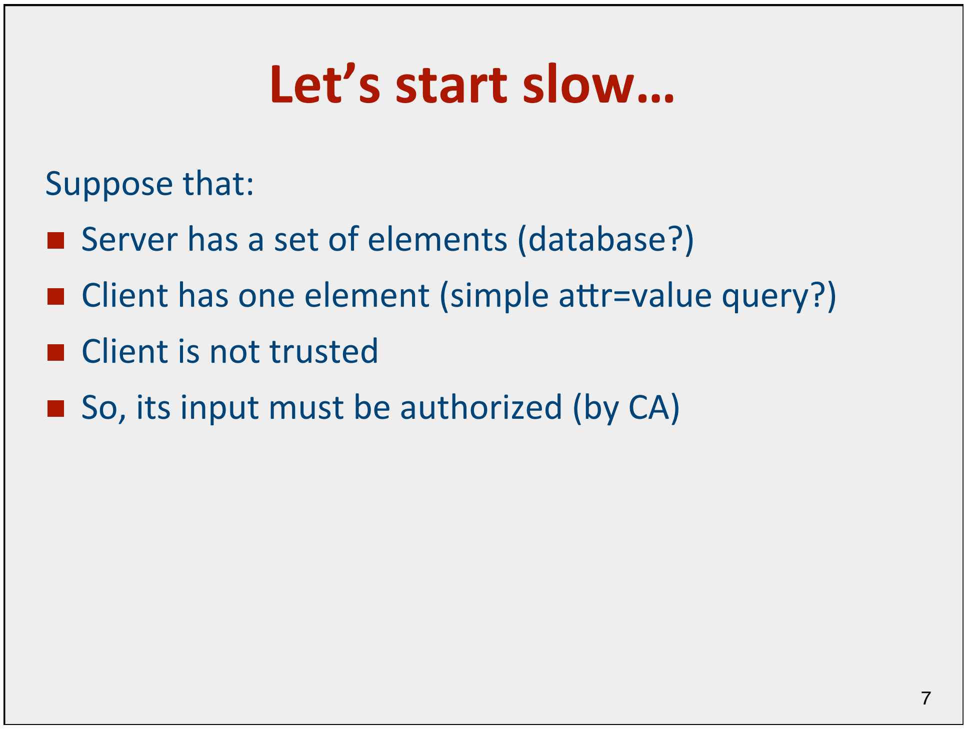
Some Variables…

- One-way or mutual sharing?
- Single or multiple elements/items on each side?
- Accompanying data transfer?
- Authorized set elements?
  - who holds them?
  - who issues them?
- Threat model
  - Outsider attacks trivial to handle
  - What kind of participant/player behavior?
    - Semi-honest (honest-but-curious)
      - Why?
    - Malicious
      - Who?
- Group setting?



Why is this not trivial?

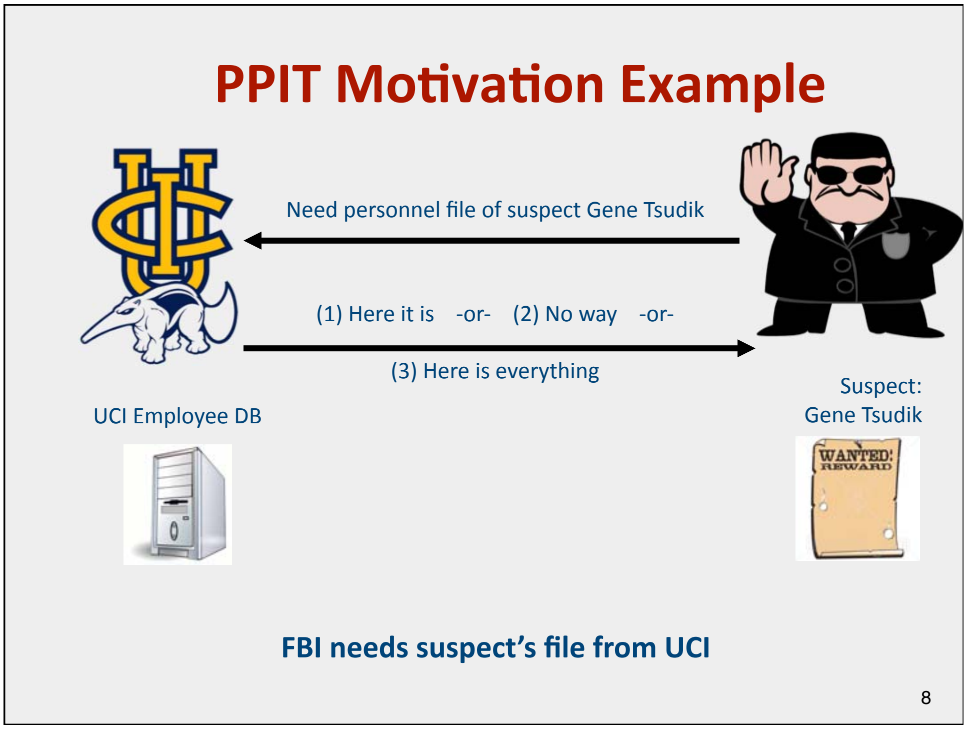
- Can 2 parties simply exchange hashes of their respective set items?
- How about: agree on a key and then exchanged keyed hashes?
- What if there is a TTP? Can we both parties send their sets (encrypted) to it and get results?



Let’s start slow…

Suppose that:

- Server has a set of elements (database?)
- Client has one element (simple attr=value query?)
- Client is not trusted
- So, its input must be authorized (by CA)



PPIT Motivation Example

Need personnel file of suspect Gene Tsudik

(1) Here it is -or- (2) No way -or-

(3) Here is everything

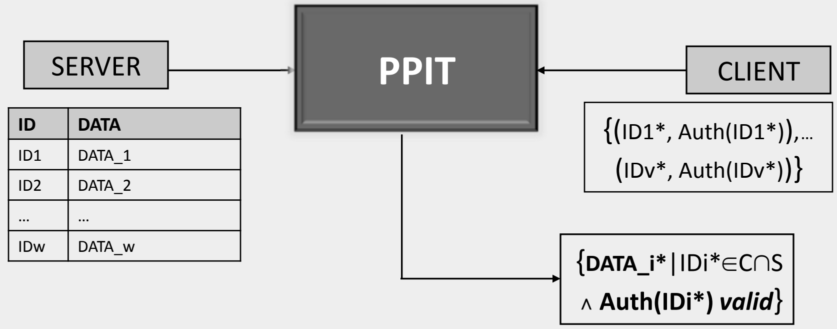
UCI Employee DB

Suspect:
Gene Tsudik

FBI needs suspect’s file from UCI

PPIT Requirements

- Policy-based
  - Client must have valid **authorization** by recognized authority, e.g., a CA
- Privacy-preserving
  - Minimize disclosure of sensitive information
  - Authorization itself represents sensitive information



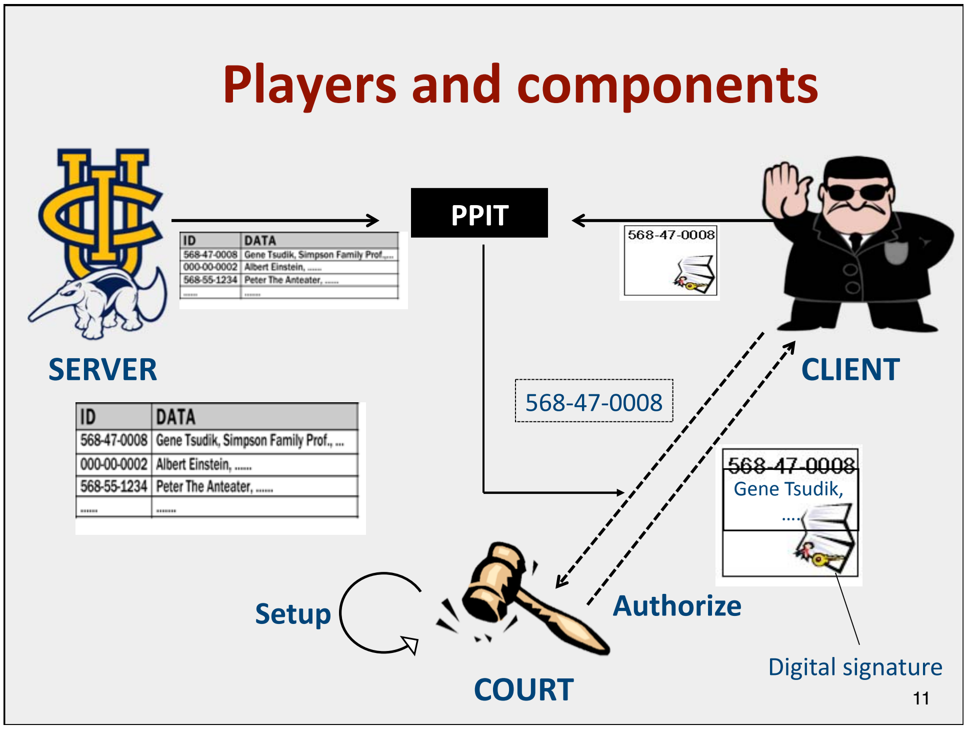
ID	DATA
ID1	DATA_1
ID2	DATA_2
…	…
IDw	DATA_w

{(ID1*, Auth(ID1*)),… (IDv*, Auth(IDv*))}

{**DATA_i***|IDi*∈C∩S ∧ **Auth(IDi*)** ***valid***}

Related Work

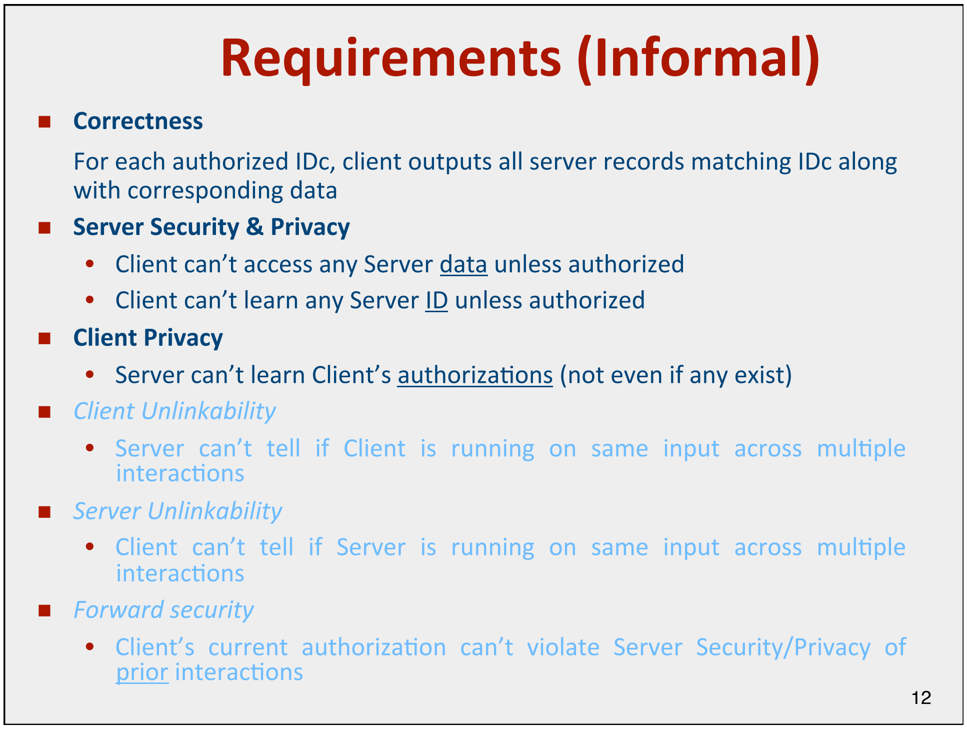
- Secure Computation
  - Inefficient
- Private Information Retrieval (PIR)
  - No Server Privacy, No authorization
- Oblivious Signature Based Envelope (OSBEs)
  - Single Message
- Public Key Encryption with Keyword Search (PEKS)
  - Not scalable



Players and components

ID	DATA
568-47-0008	Gene Tsudik, Simpson Family Prof., …
000-00-0002	Albert Einstein, ……
568-55-1234	Peter The Anteater, ……
……	……

11



Requirements (Informal)

- **Correctness**

  For each authorized IDc, client outputs all server records matching IDc along with corresponding data
- **Server Security & Privacy**
  - Client can't access any Server data unless authorized
  - Client can't learn any Server ID unless authorized
- **Client Privacy**
  - Server can't learn Client's authorizations (not even if any exist)
- *Client Unlinkability*
  - Server can't tell if Client is running on same input across multiple interactions
- *Server Unlinkability*
  - Client can't tell if Server is running on same input across multiple interactions
- *Forward security*
  - Client's current authorization can't violate Server Security/Privacy of prior interactions

12

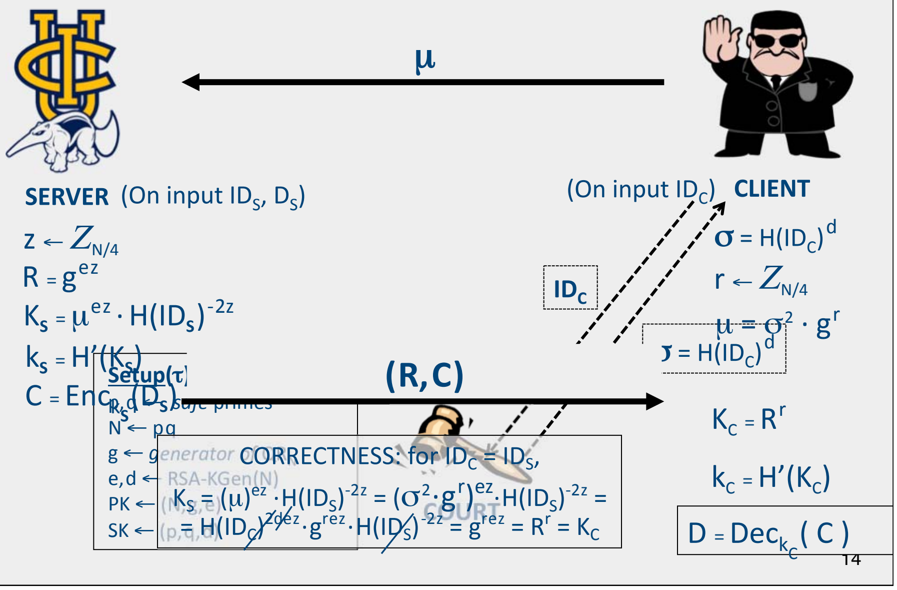
RSA-PPIT

- **Based on plain RSA**
- Setup

Court
- *p,q* primes
- *N=pq*
- e, such that gcd(e,φ(N))=1
- *d,* such that ed=1 *mod* φ(N)
- (*N*, *d*) = signing/secret key
- (*N*, *e*) = verification/public key
- H(), H’(), H’’() – full-domain cryptographic hash functions

- Sign(M, *d*) = σ = H(M)^{d} *mod* N
- Verify(M, σ, *e*) = 1 *iff* σ^{e} = H(M)

RSA-PPIT



SERVER (On input ID_S, D_S)

$z \leftarrow Z_{N/4}$

$R = g^{ez}$

$K_s = \mu^{ez} \cdot H(ID_s)^{-2z}$

$k_s = H'(K_s)$

$C = Enc_{k_s}(D_s)$

μ

(On input ID_C) **CLIENT**

$\sigma = H(ID_C)^d$

$r \leftarrow Z_{N/4}$

$\mu = \sigma^2 \cdot g^r$

ID_C

$\sigma = H(ID_C)^d$

(R,C)

$K_C = R^r$

$k_C = H'(K_C)$

$D = Dec_{k_C}(C)$

Setup(τ)

p,q ← safe primes

N ← pq

g ← *generator of QR*

e,d ← RSA-KGen(N)

PK ← (N,g,e)

SK ← (p,q,d)

COURT

CORRECTNESS: for $ID_C = ID_S$,

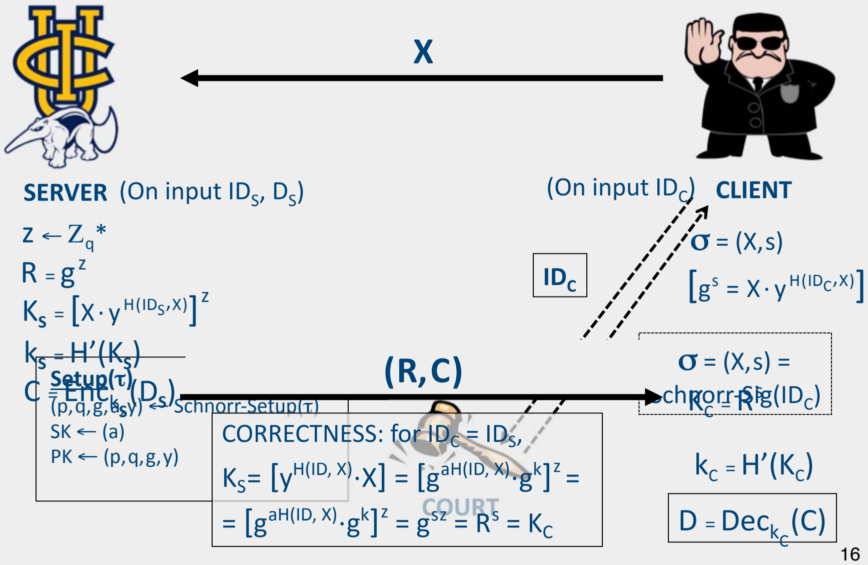
$K_S = (\mu)^{ez} \cdot H(ID_S)^{-2z} = (\sigma^2 \cdot g^r)^{ez} \cdot H(ID_S)^{-2z} =$

$= \cancel{H(ID_C)^{2dez}} \cdot g^{rez} \cdot \cancel{H(ID_S)^{-2z}} = g^{rez} = R^r = K_C$

Schnorr-PPIT

- Setup:
  - *p* – large prime, *q* – small prime factor of *p-1*
  - $g \in Z_p^*$ of order q
  - Secret key: $a \in Z_p^*$
  - Public key: $y=g^a$ *mod* p
- Sign(M, a) = **[X,s]** = [g^k *mod* p , k+a·H(M,X) *mod* q] for $k \in Z_q^*$
- Verify(M, y, [X, s]) = 1 *iff* $g^s = X \cdot y^{H(M,X)}$ *mod* p
- Similar to RSA-PPIT, but:
  - Client sends *half* of the signature → **X**
  - Client introduces no randomness

Schnorr-PPIT



SERVER (On input ID_S, D_S)

$z \leftarrow Z_q^*$

$R = g^z$

$K_s = [X \cdot y^{H(ID_S,X)}]^z$

$k_s = H'(K_s)$

$C = Enc_{k_s}(D_s)$

Setup(τ)

$(p,q,g,a,y) \leftarrow$ Schnorr-Setup(τ)

SK ← (a)

PK ← (p,q,g,y)

(On input ID_C) **CLIENT**

$\sigma = (X,s)$

$[g^s = X \cdot y^{H(ID_C,X)}]$

ID_C

$\sigma = (X,s) =$ Schnorr-Sig(ID_C)

$K_C = R^s$

$k_C = H'(K_C)$

$D = Dec_{k_C}(C)$

CORRECTNESS: for $ID_C = ID_S$,

$K_S = [y^{H(ID, X)} \cdot X] = [g^{aH(ID, X)} \cdot g^k]^z =$

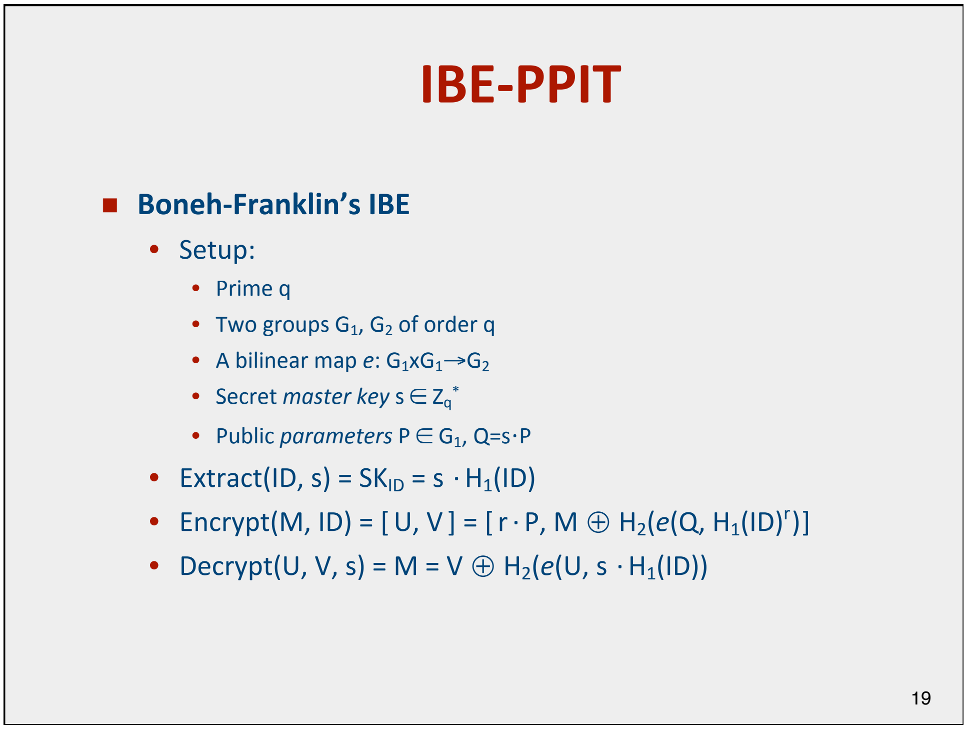
$= [g^{aH(ID, X)} \cdot g^k]^z = g^{sz} = R^s = K_C$

Optional security features

	RSA	Schnorr
Client Unlinkability	Yes	No
Forward Security	Yes	No

Identity-based Encryption

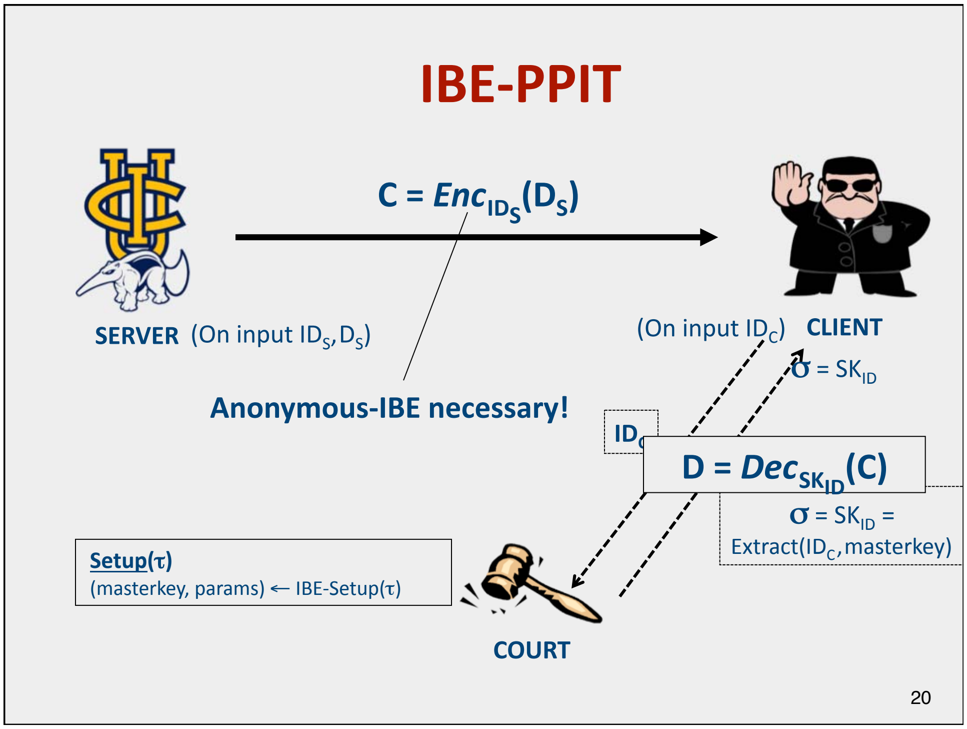
- A public-key cryptosystem where any string (*ID*) is a valid public (encryption) key
  - Involves a fully-trusted Private Key Generator (PKG)
- Algorithms:
  - **Setup**
    - Generates public parameters and secret master key *s*
  - **Encrypt(*M, ID*) → *C***
    - Encrypts M using ID as public (encryption) key
  - **Extract-Key(*ID, s*) → SK_{ID}**
    - Run by the PKG: outputs secret decryption key corresponding to *ID*
  - **Decrypt(*C*, SK_{ID}) → *M***
    - Decrypts ciphertext encrypted under *ID*



IBE-PPIT

- **Boneh-Franklin's IBE**
  - Setup:
    - Prime q
    - Two groups G_1, G_2 of order q
    - A bilinear map e: $G_1 x G_1 \rightarrow G_2$
    - Secret *master key* $s \in Z_q^*$
    - Public *parameters* $P \in G_1$, $Q = s \cdot P$
  - Extract(ID, s) = $SK_{ID} = s \cdot H_1(ID)$
  - Encrypt(M, ID) = $[\,U, V\,] = [\,r \cdot P, M \oplus H_2(e(Q, H_1(ID)^r)\,]$
  - Decrypt(U, V, s) = $M = V \oplus H_2(e(U, s \cdot H_1(ID))$

19



IBE-PPIT

C = $Enc_{ID_S}(D_S)$

SERVER (On input ID_S, D_S)

(On input ID_C) **CLIENT**

$\sigma = SK_{ID}$

Anonymous-IBE necessary!

ID_C

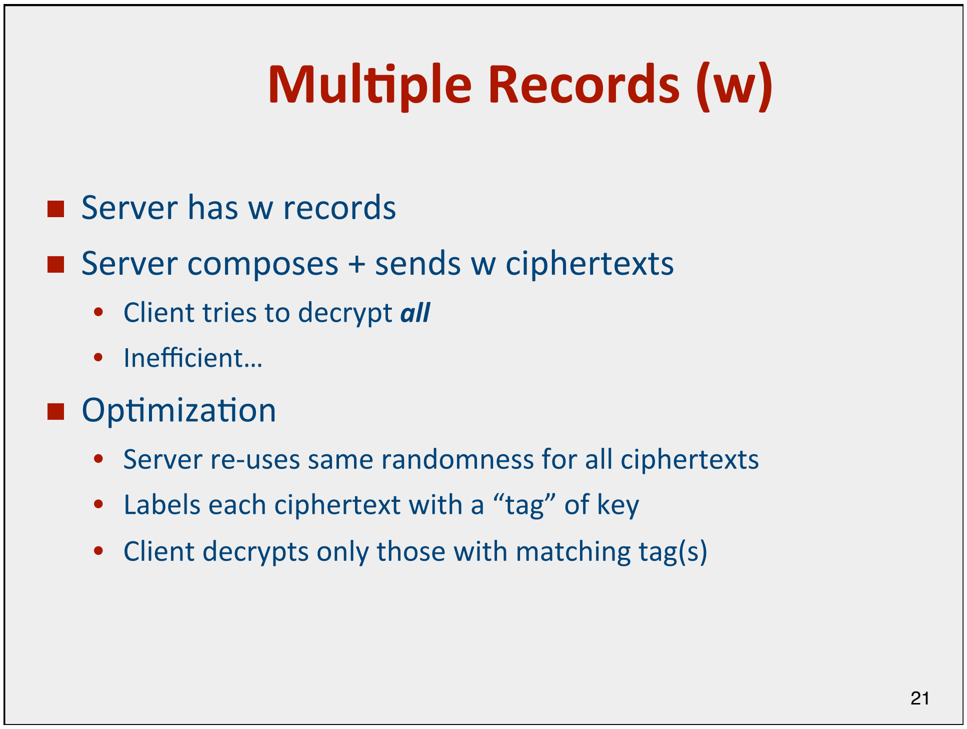
D = $Dec_{SK_{ID}}(C)$

$\sigma = SK_{ID} =$ Extract(ID_C, masterkey)

Setup(τ)
(masterkey, params) ← IBE-Setup(τ)

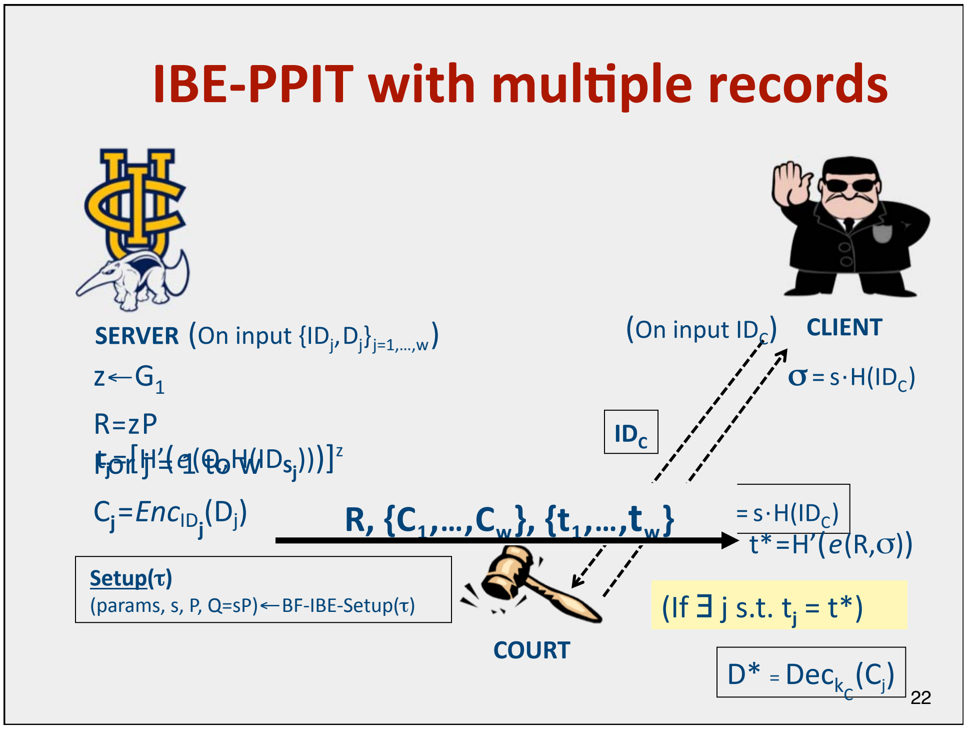
COURT

20



Multiple Records (w)

- Server has w records
- Server composes + sends w ciphertexts
  - Client tries to decrypt ***all***
  - Inefficient…
- Optimization
  - Server re-uses same randomness for all ciphertexts
  - Labels each ciphertext with a "tag" of key
  - Client decrypts only those with matching tag(s)



IBE-PPIT with multiple records

SERVER (On input $\{ID_j, D_j\}_{j=1,\ldots,w}$)

$z \leftarrow G_1$

$R = zP$

For j = 1 to w

$t_j = [H'(e(Q, H(IDs_j)))]^z$

$C_j = Enc_{ID_j}(D_j)$

Setup(τ)
(params, s, P, Q=sP)←BF-IBE-Setup(τ)

(On input ID_C) **CLIENT**

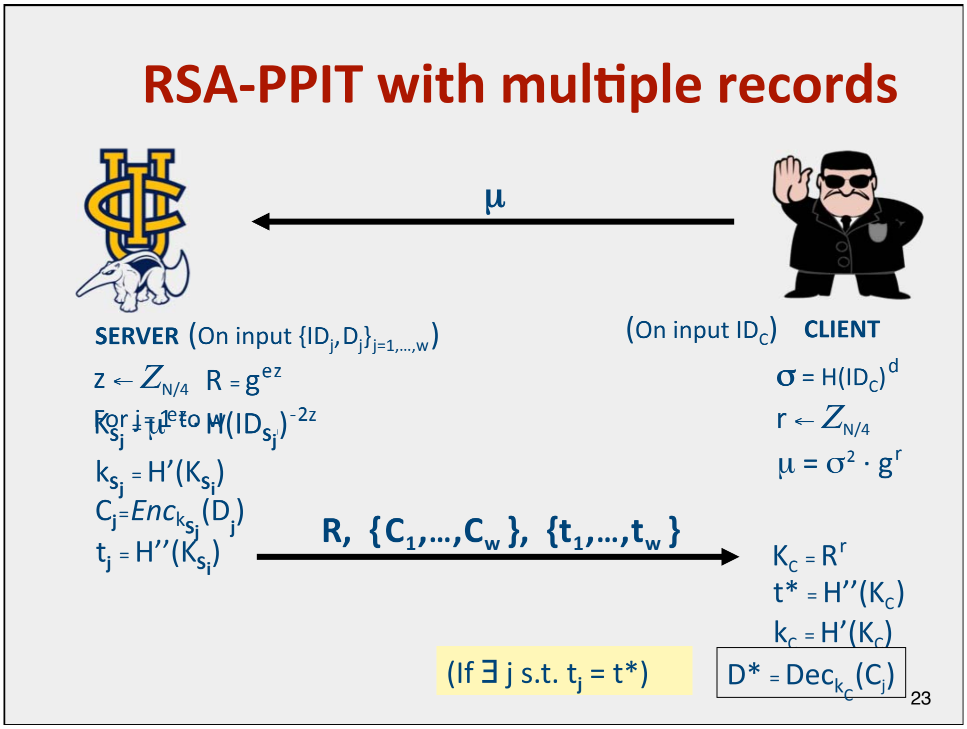
$\sigma = s \cdot H(ID_C)$

$= s \cdot H(ID_C)$

$t^* = H'(e(R, \sigma))$

(If ∃ j s.t. $t_j = t^*$)

$D^* = Dec_{k_C}(C_j)$



RSA-PPIT with multiple records

μ

SERVER (On input $\{ID_j, D_j\}_{j=1,\ldots,w}$)

$z \leftarrow Z_{N/4}$ $R = g^{ez}$

For j = 1 to w

$K_{s_j} = \mu^{ez} \cdot H(ID_{s_j})^{-2z}$

$k_{s_j} = H'(K_{s_j})$

$C_j = Enc_{k_{s_j}}(D_j)$

$t_j = H''(K_{s_j})$

(On input ID_C) **CLIENT**

$\sigma = H(ID_C)^d$

$r \leftarrow Z_{N/4}$

$\mu = \sigma^2 \cdot g^r$

R, $\{C_1,\ldots,C_w\}$, $\{t_1,\ldots,t_w\}$

$K_C = R^r$

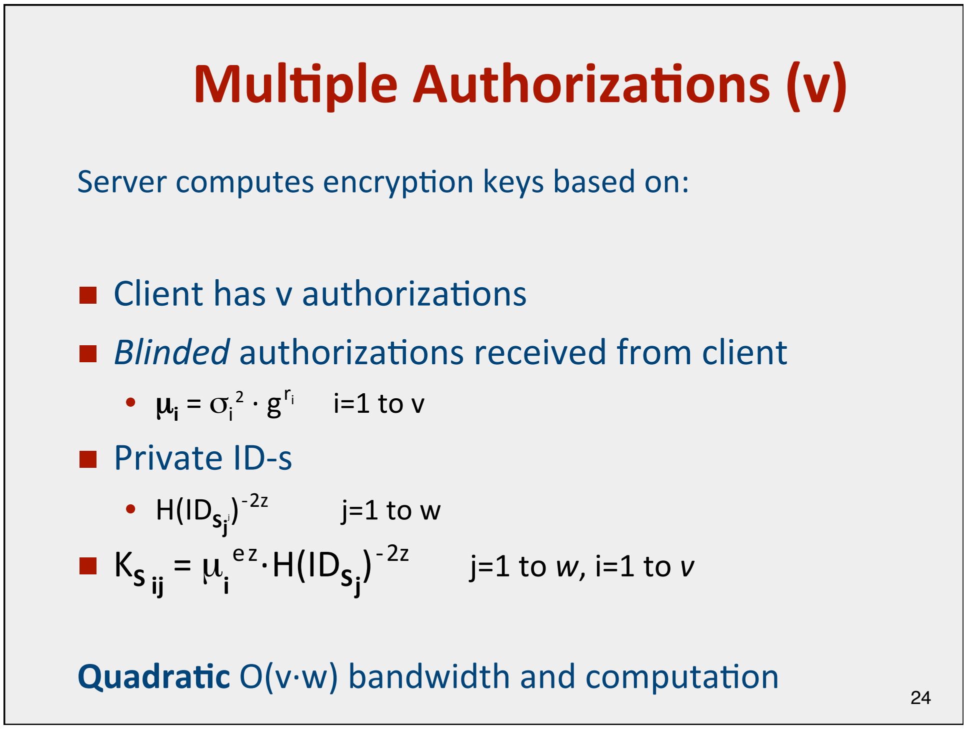
$t^* = H''(K_C)$

$k_C = H'(K_C)$

(If $\exists$ j s.t. $t_j = t^*$)

$\boxed{D^* = Dec_{k_C}(C_j)}$

23



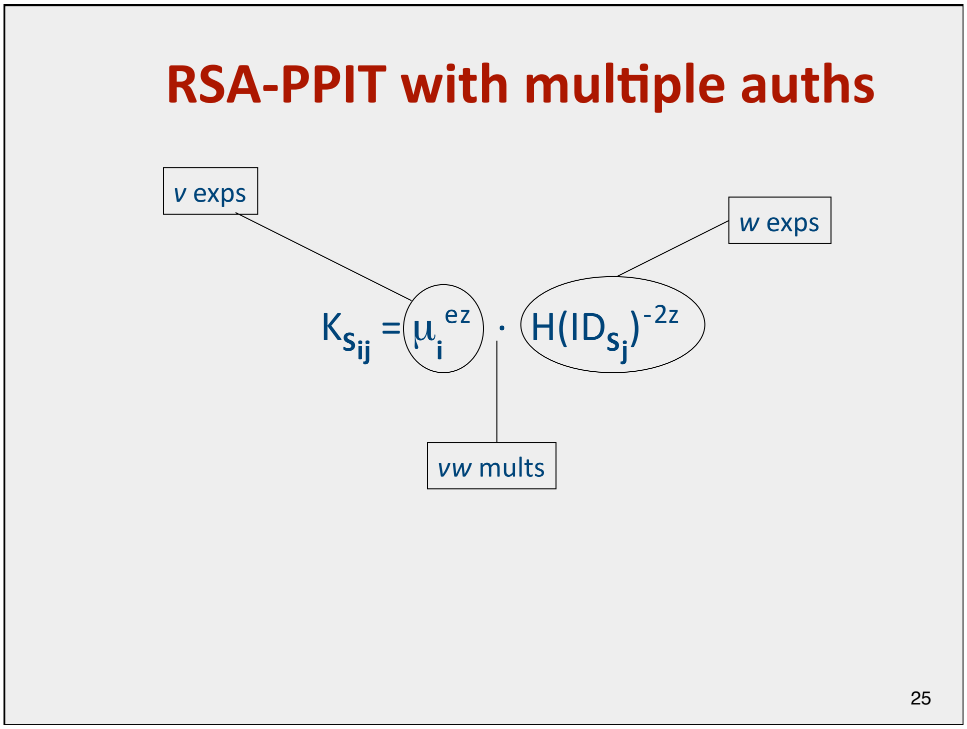
Multiple Authorizations (v)

Server computes encryption keys based on:

- Client has v authorizations
- *Blinded* authorizations received from client
  - $\mu_i = \sigma_i^2 \cdot g^{r_i}$ i=1 to v
- Private ID-s
  - $H(ID_{s_j})^{-2z}$ j=1 to w
- $K_{s\,ij} = \mu_i^{ez} \cdot H(ID_{s_j})^{-2z}$ j=1 to *w*, i=1 to *v*

Quadratic O(v·w) bandwidth and computation

24



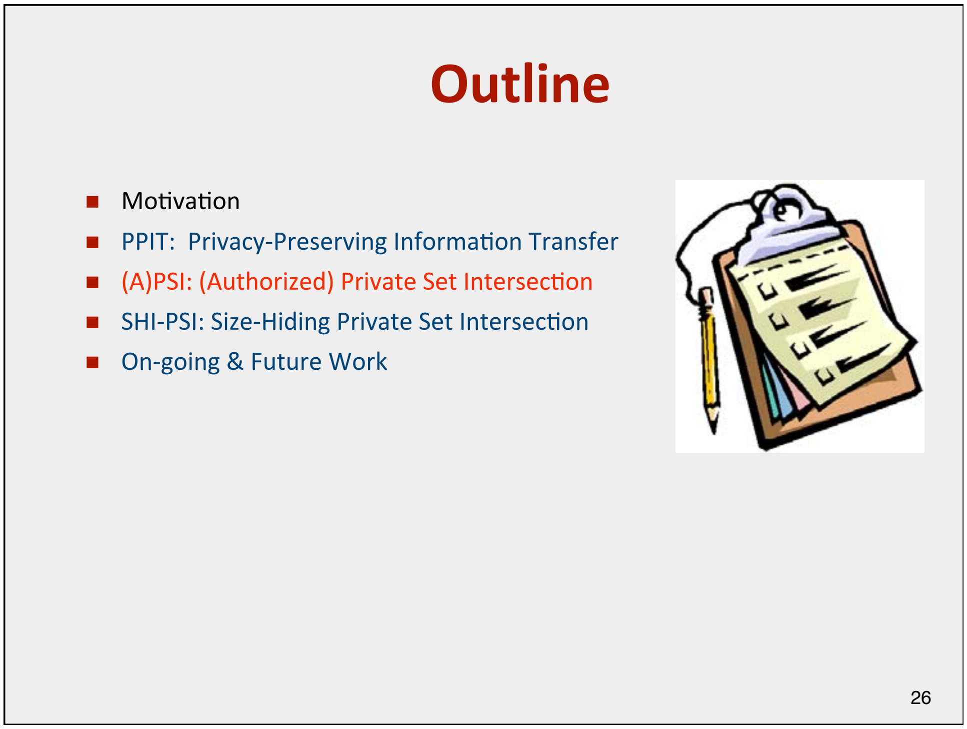
RSA-PPIT with multiple auths

v exps

w exps

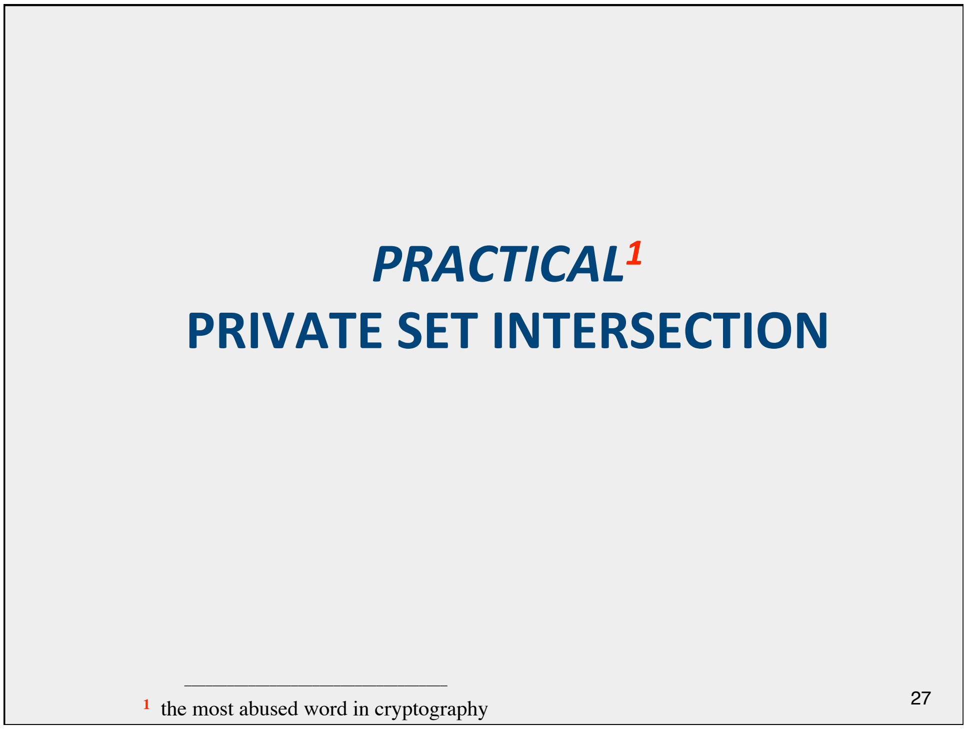
$$K_{s_{ij}} = \mu_i^{ez} \cdot H(ID_{s_j})^{-2z}$$

vw mults



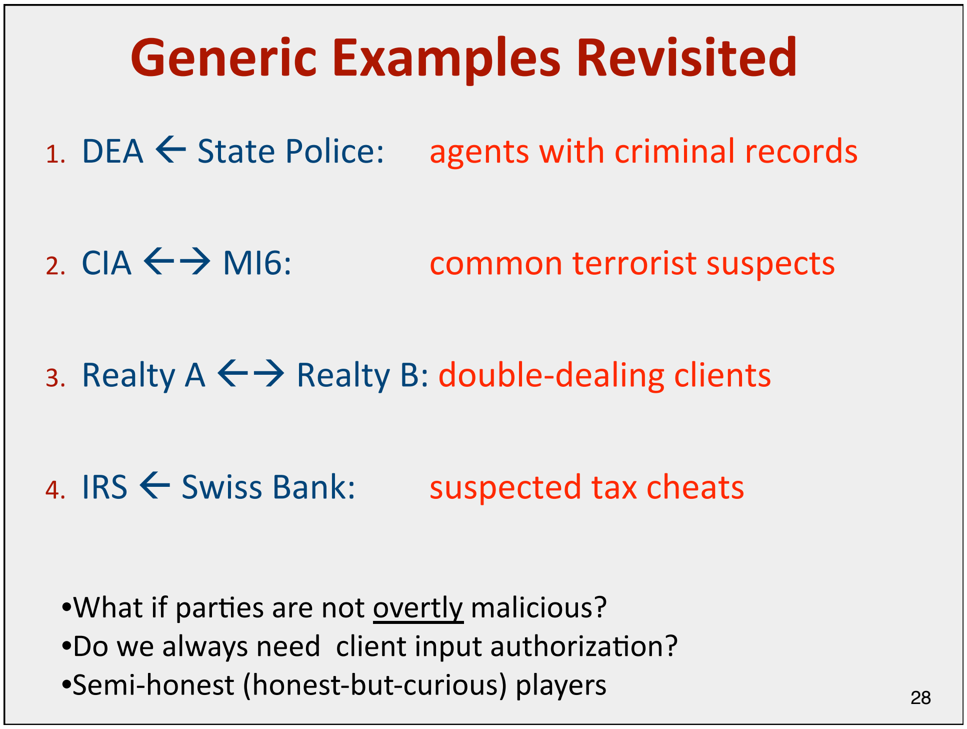
Outline

- Motivation
- PPIT: Privacy-Preserving Information Transfer
- (A)PSI: (Authorized) Private Set Intersection
- SHI-PSI: Size-Hiding Private Set Intersection
- On-going & Future Work



PRACTICAL[1] PRIVATE SET INTERSECTION

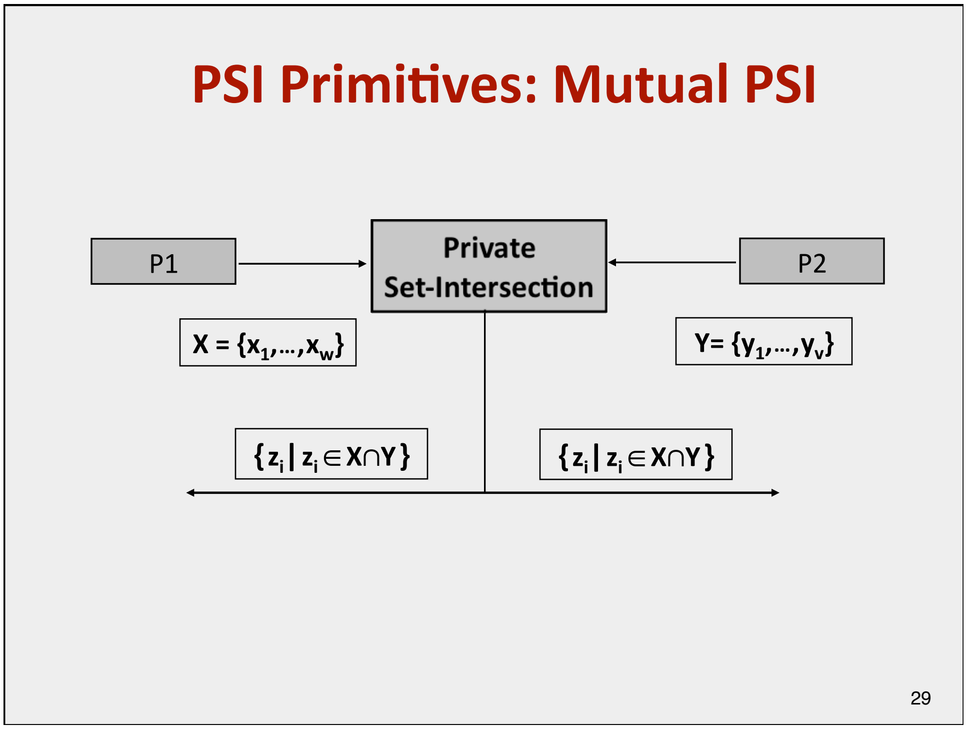
[1] the most abused word in cryptography



Generic Examples Revisited

1. DEA ← State Police: agents with criminal records
2. CIA ←→ MI6: common terrorist suspects
3. Realty A ←→ Realty B: double-dealing clients
4. IRS ← Swiss Bank: suspected tax cheats

- What if parties are not <u>overtly</u> malicious?
- Do we always need client input authorization?
- Semi-honest (honest-but-curious) players



PSI Primitives: Mutual PSI

P1

Private Set-Intersection

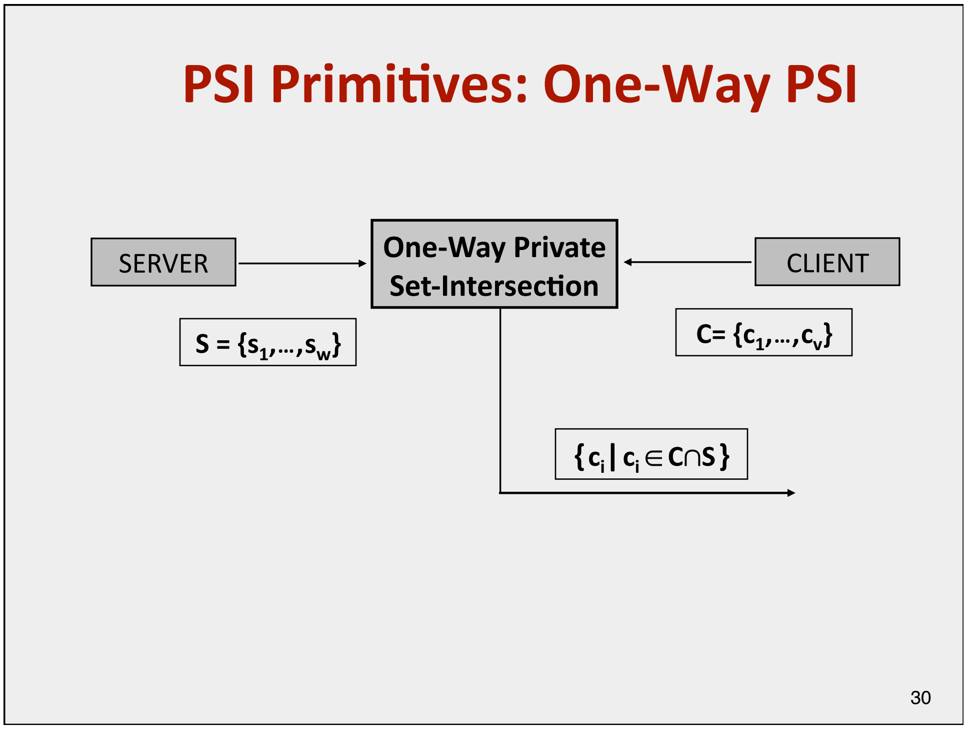
P2

$X = \{x_1,\ldots,x_w\}$

$Y= \{y_1,\ldots,y_v\}$

$\{z_i \mid z_i \in X \cap Y\}$

$\{z_i \mid z_i \in X \cap Y\}$



PSI Primitives: One-Way PSI

SERVER

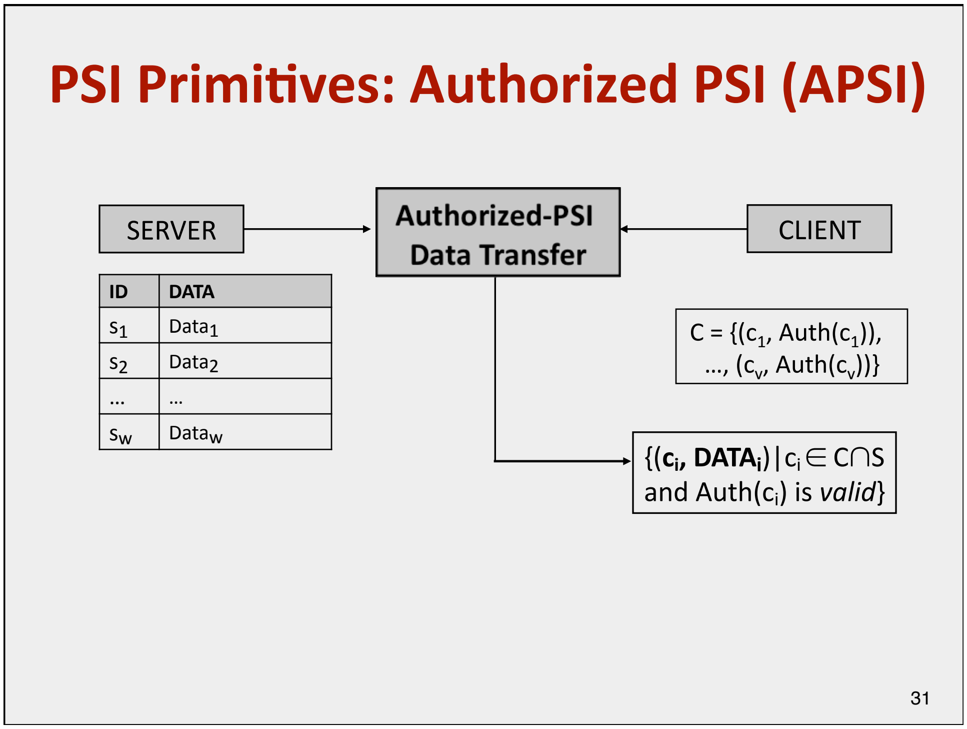
One-Way Private Set-Intersection

CLIENT

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

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

$\{c_i \mid c_i \in C \cap S\}$



PSI Primitives: Authorized PSI (APSI)

SERVER

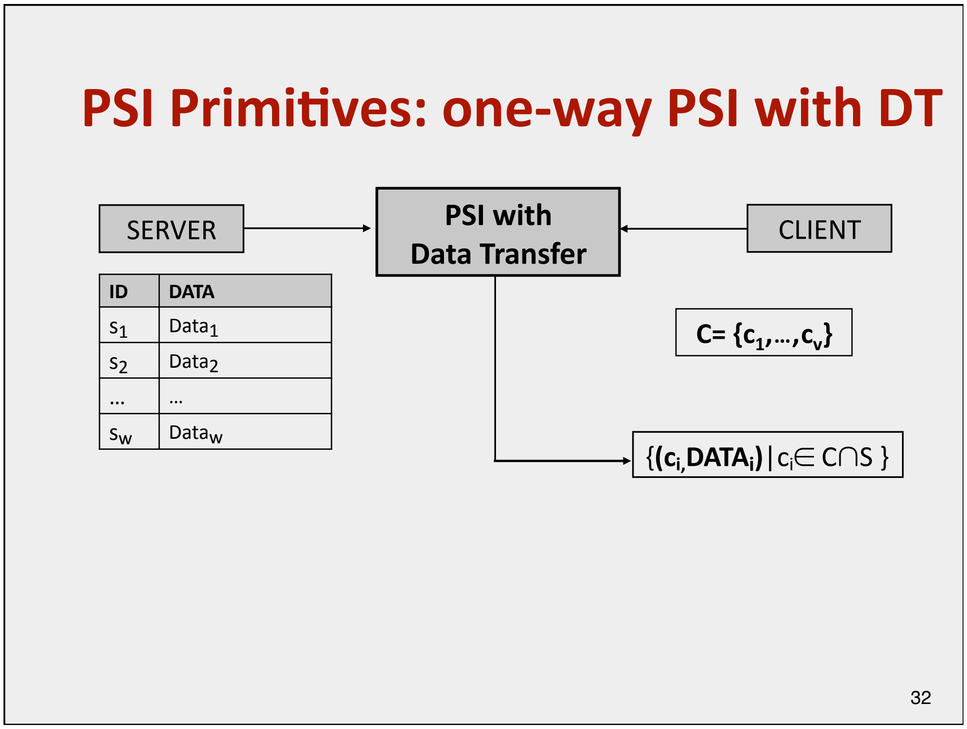
Authorized-PSI Data Transfer

CLIENT

ID	DATA
s_1	Data_1
s_2	Data_2
…	…
s_w	Data_w

$C = \{(c_1, \text{Auth}(c_1)), \dots, (c_v, \text{Auth}(c_v))\}$

$\{(\mathbf{c_i}, \mathbf{DATA_i}) \mid c_i \in C \cap S$ and $\text{Auth}(c_i)$ is *valid*$\}$



PSI Primitives: one-way PSI with DT

SERVER

PSI with Data Transfer

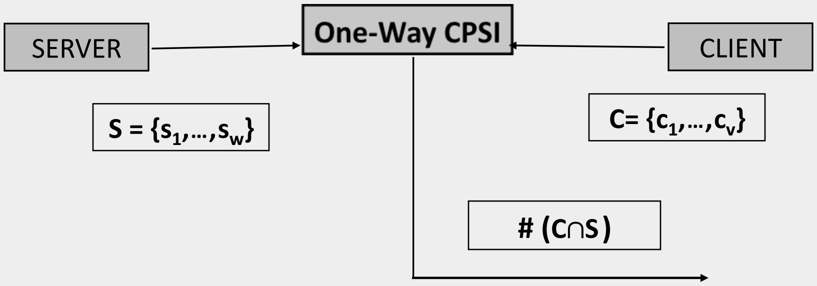
CLIENT

ID	DATA
s_1	Data_1
s_2	Data_2
…	…
s_w	Data_w

$\mathbf{C = \{c_1, \dots, c_v\}}$

$\{(\mathbf{c_i}, \mathbf{DATA_i}) \mid c_i \in C \cap S \}$

PSI Primitives: One-Way Cardinality-only PCSI



SERVER

One-Way CPSI

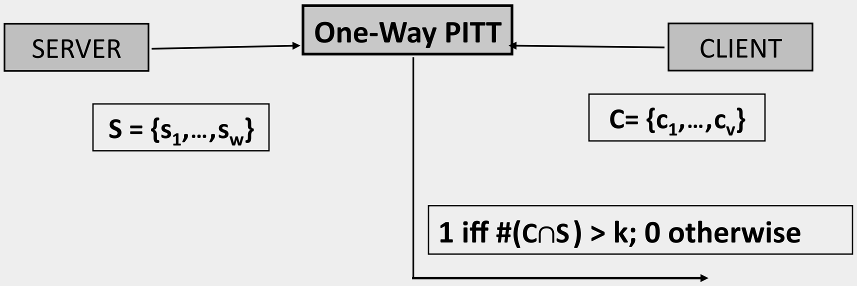
CLIENT

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

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

$\#\,(C \cap S)$

PSI Primitives: One-Way Private Intersection Threshold Testing



SERVER

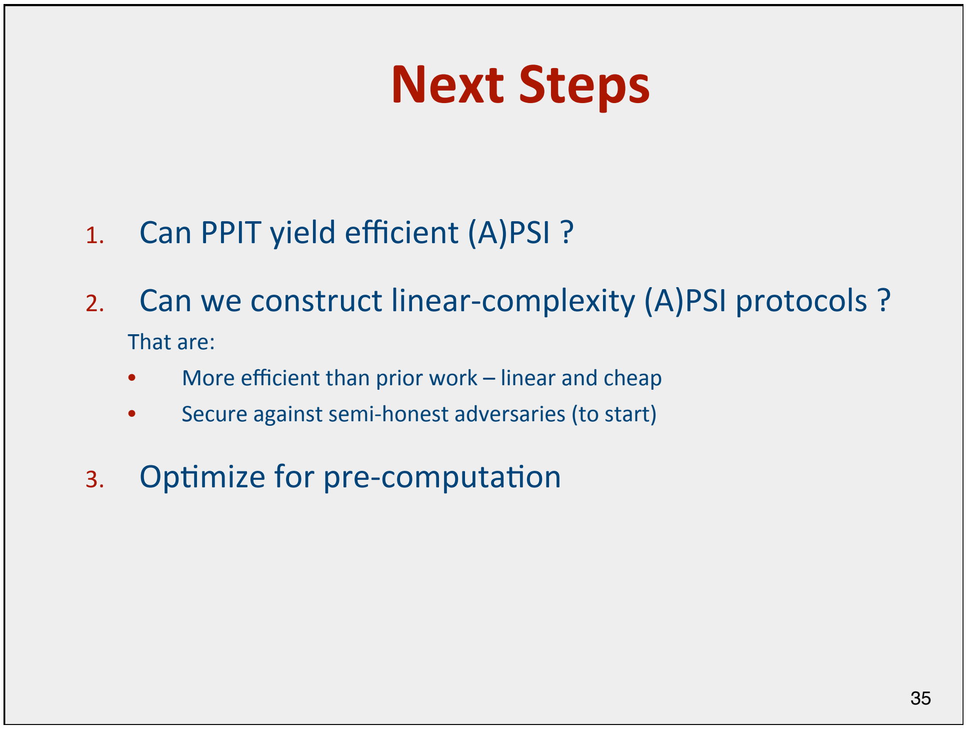
One-Way PITT

CLIENT

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

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

1 iff $\#(C \cap S) > k$; 0 otherwise



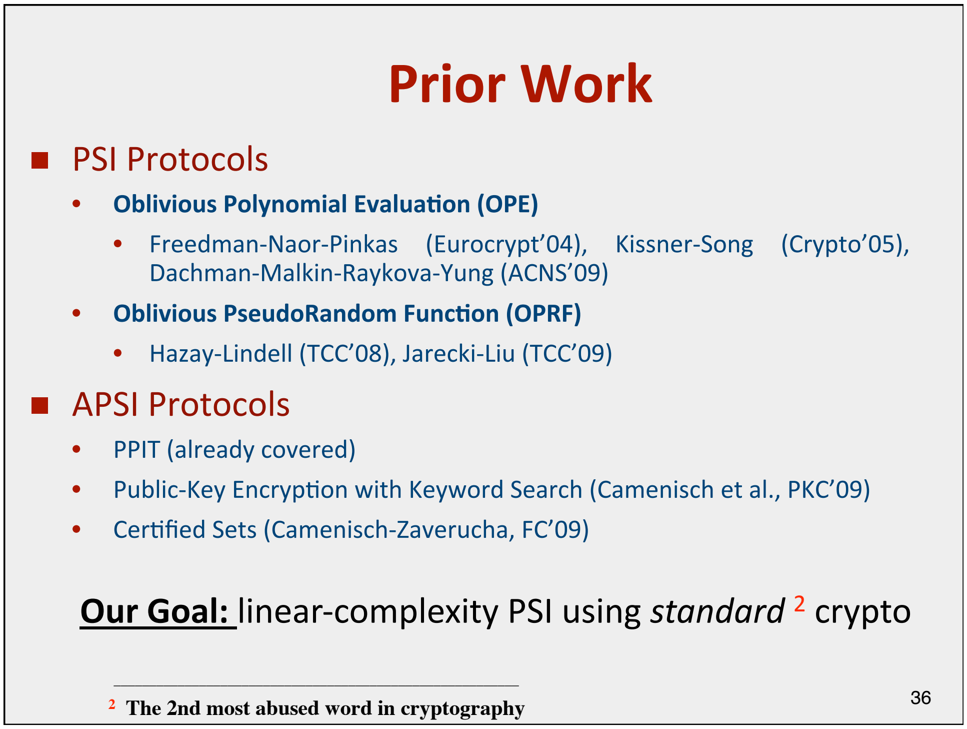
Next Steps

1. Can PPIT yield efficient (A)PSI ?
2. Can we construct linear-complexity (A)PSI protocols ?

   That are:

   - More efficient than prior work – linear and cheap
   - Secure against semi-honest adversaries (to start)
3. Optimize for pre-computation

35



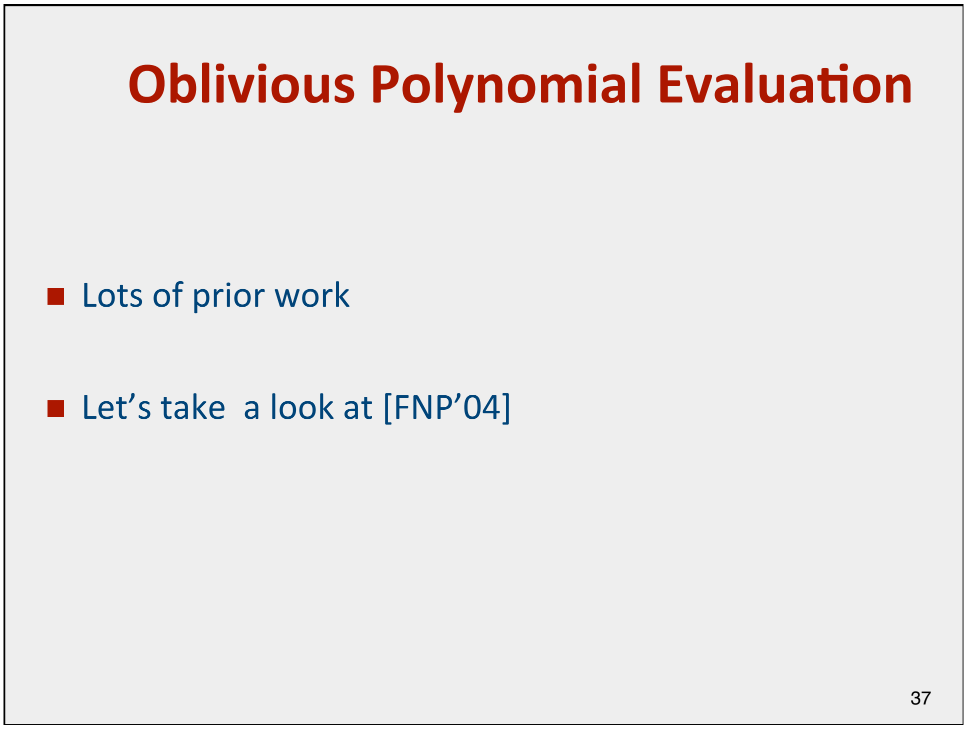
Prior Work

- PSI Protocols
  - **Oblivious Polynomial Evaluation (OPE)**
    - Freedman-Naor-Pinkas (Eurocrypt'04), Kissner-Song (Crypto'05), Dachman-Malkin-Raykova-Yung (ACNS'09)
  - **Oblivious PseudoRandom Function (OPRF)**
    - Hazay-Lindell (TCC'08), Jarecki-Liu (TCC'09)
- APSI Protocols
  - PPIT (already covered)
  - Public-Key Encryption with Keyword Search (Camenisch et al., PKC'09)
  - Certified Sets (Camenisch-Zaverucha, FC'09)

Our Goal: linear-complexity PSI using *standard* [2] crypto

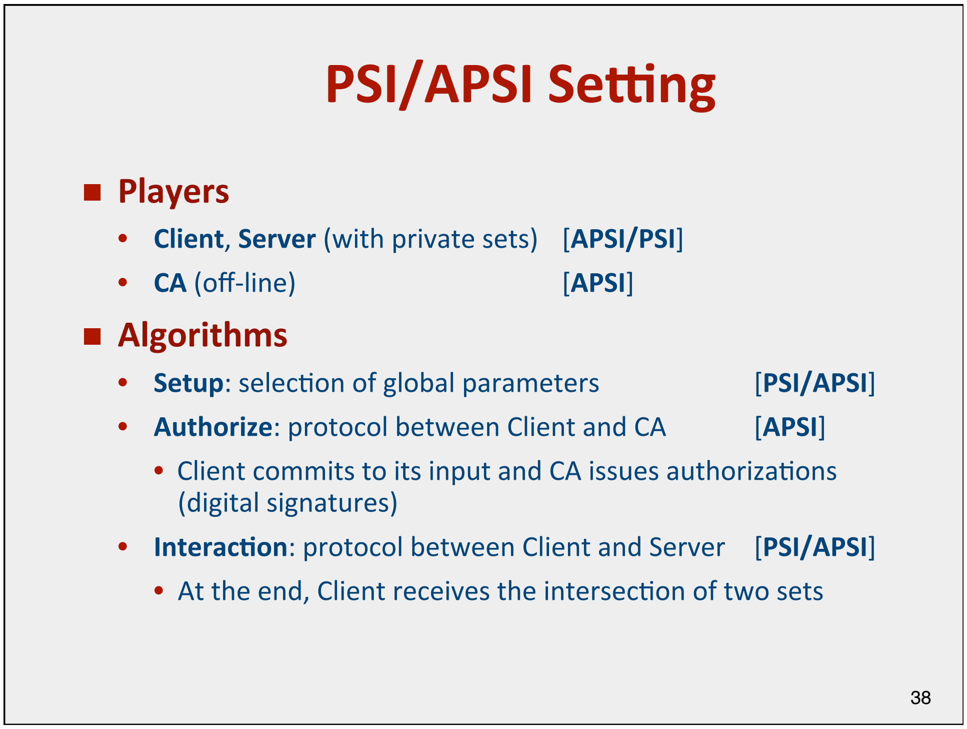
[2] **The 2nd most abused word in cryptography**

36



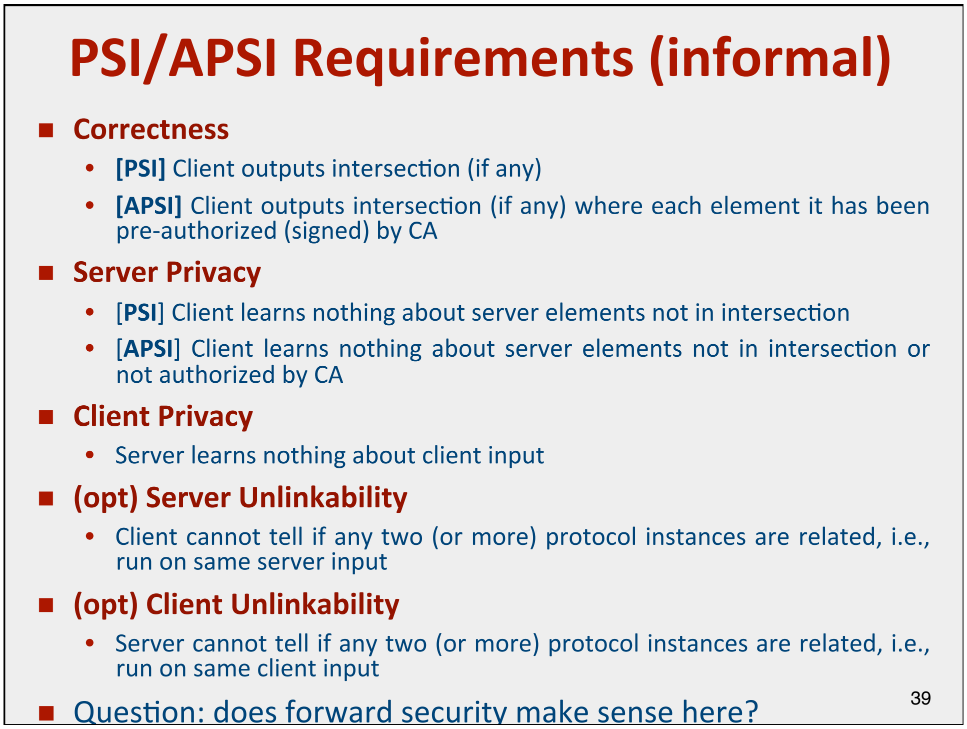
Oblivious Polynomial Evaluation

- Lots of prior work
- Let's take a look at [FNP'04]



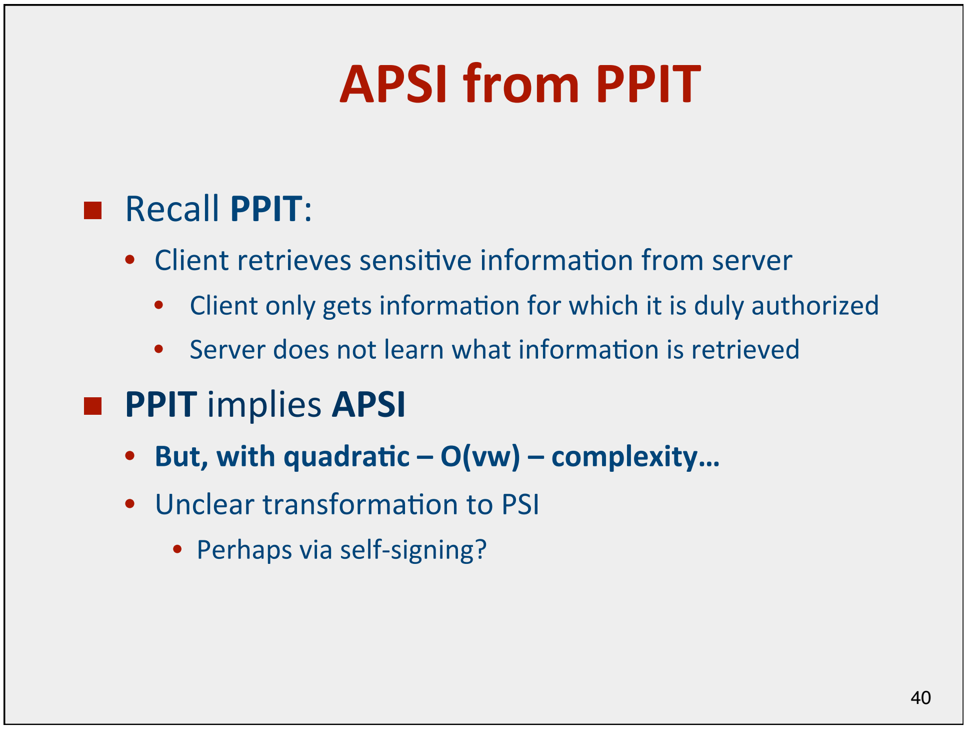
PSI/APSI Setting

- **Players**
  - **Client**, **Server** (with private sets)   [**APSI/PSI**]
  - **CA** (off-line)   [**APSI**]
- **Algorithms**
  - **Setup**: selection of global parameters   [**PSI/APSI**]
  - **Authorize**: protocol between Client and CA   [**APSI**]
    - Client commits to its input and CA issues authorizations (digital signatures)
  - **Interaction**: protocol between Client and Server   [**PSI/APSI**]
    - At the end, Client receives the intersection of two sets



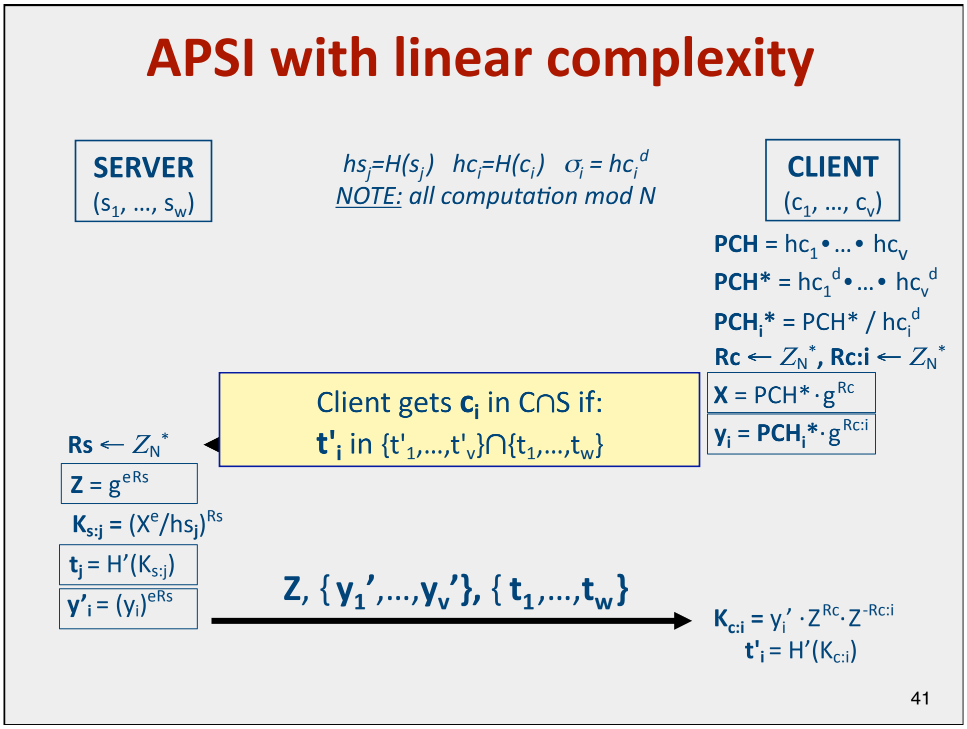
PSI/APSI Requirements (informal)

- **Correctness**
  - **[PSI]** Client outputs intersection (if any)
  - **[APSI]** Client outputs intersection (if any) where each element it has been pre-authorized (signed) by CA
- **Server Privacy**
  - [**PSI**] Client learns nothing about server elements not in intersection
  - [**APSI**] Client learns nothing about server elements not in intersection or not authorized by CA
- **Client Privacy**
  - Server learns nothing about client input
- **(opt) Server Unlinkability**
  - Client cannot tell if any two (or more) protocol instances are related, i.e., run on same server input
- **(opt) Client Unlinkability**
  - Server cannot tell if any two (or more) protocol instances are related, i.e., run on same client input
- Question: does forward security make sense here?



APSI from PPIT

- Recall **PPIT**:
  - Client retrieves sensitive information from server
    - Client only gets information for which it is duly authorized
    - Server does not learn what information is retrieved
- **PPIT** implies **APSI**
  - **But, with quadratic – O(vw) – complexity…**
  - Unclear transformation to PSI
    - Perhaps via self-signing?



APSI with linear complexity

SERVER
$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$ $\sigma_i = hc_i^d$
NOTE: all computation mod N

CLIENT
$(c_1, \ldots, c_v)$

PCH = $hc_1 \bullet \ldots \bullet hc_v$

PCH* = $hc_1^d \bullet \ldots \bullet hc_v^d$

$\mathbf{PCH_i^*} = PCH^* / hc_i^d$

Rc $\leftarrow Z_N^*$, **Rc:i** $\leftarrow Z_N^*$

$\mathbf{X} = PCH^* \cdot g^{Rc}$

$\mathbf{y_i = PCH_i^*} \cdot g^{Rc:i}$

Client gets $\mathbf{c_i}$ in C∩S if:
$\mathbf{t'_i}$ in $\{t'_1,\ldots,t'_v\} \cap \{t_1,\ldots,t_w\}$

Rs $\leftarrow Z_N^*$

$\mathbf{Z} = g^{eRs}$

$\mathbf{K_{s:j}} = (X^e/hs_j)^{Rs}$

$\mathbf{t_j} = H'(K_{s:j})$

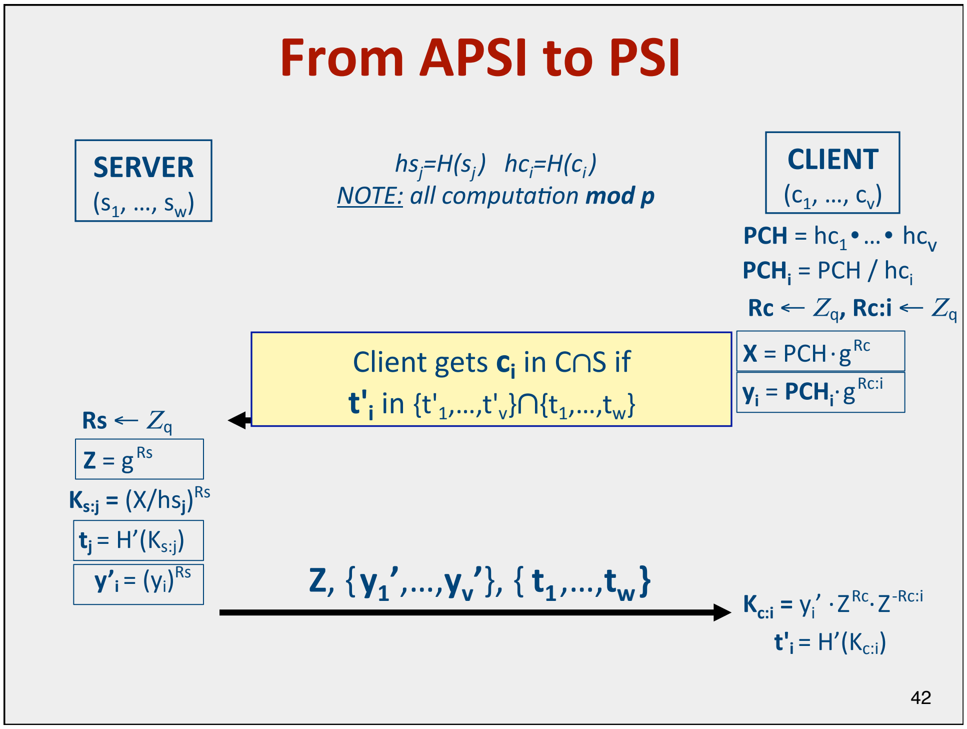
$\mathbf{y'_i} = (y_i)^{eRs}$

$\mathbf{Z, \{y_1',\ldots,y_v'\}, \{t_1,\ldots,t_w\}}$

$\mathbf{K_{c:i}} = y_i' \cdot Z^{Rc} \cdot Z^{-Rc:i}$

$\mathbf{t'_i} = H'(K_{c:i})$

41



From APSI to PSI

SERVER
$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$
*NOTE: all computation **mod p***

CLIENT
$(c_1, \ldots, c_v)$

PCH = $hc_1 \bullet \ldots \bullet hc_v$

$\mathbf{PCH_i} = PCH / hc_i$

Rc $\leftarrow Z_q$, **Rc:i** $\leftarrow Z_q$

$\mathbf{X} = PCH \cdot g^{Rc}$

$\mathbf{y_i = PCH_i} \cdot g^{Rc:i}$

Client gets $\mathbf{c_i}$ in C∩S if
$\mathbf{t'_i}$ in $\{t'_1,\ldots,t'_v\} \cap \{t_1,\ldots,t_w\}$

Rs $\leftarrow Z_q$

$\mathbf{Z} = g^{Rs}$

$\mathbf{K_{s:j}} = (X/hs_j)^{Rs}$

$\mathbf{t_j} = H'(K_{s:j})$

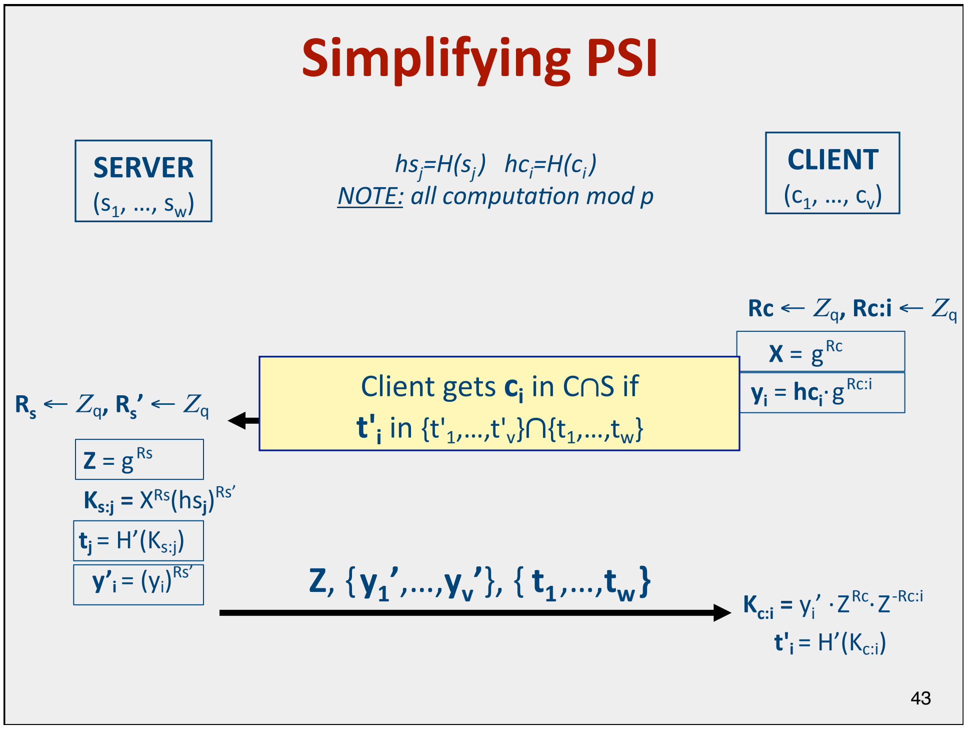
$\mathbf{y'_i} = (y_i)^{Rs}$

$\mathbf{Z, \{y_1',\ldots,y_v'\}, \{t_1,\ldots,t_w\}}$

$\mathbf{K_{c:i}} = y_i' \cdot Z^{Rc} \cdot Z^{-Rc:i}$

$\mathbf{t'_i} = H'(K_{c:i})$

42



Simplifying PSI

SERVER $(s_1, \dots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$

NOTE: all computation mod p

CLIENT $(c_1, \dots, c_v)$

$Rc \leftarrow Z_q$, $Rc{:}i \leftarrow Z_q$

$X = g^{Rc}$

$y_i = hc_i \cdot g^{Rc:i}$

Client gets c_i in C∩S if t'_i in $\{t'_1,\dots,t'_v\} \cap \{t_1,\dots,t_w\}$

$R_s \leftarrow Z_q$, $R_s' \leftarrow Z_q$

$Z = g^{Rs}$

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

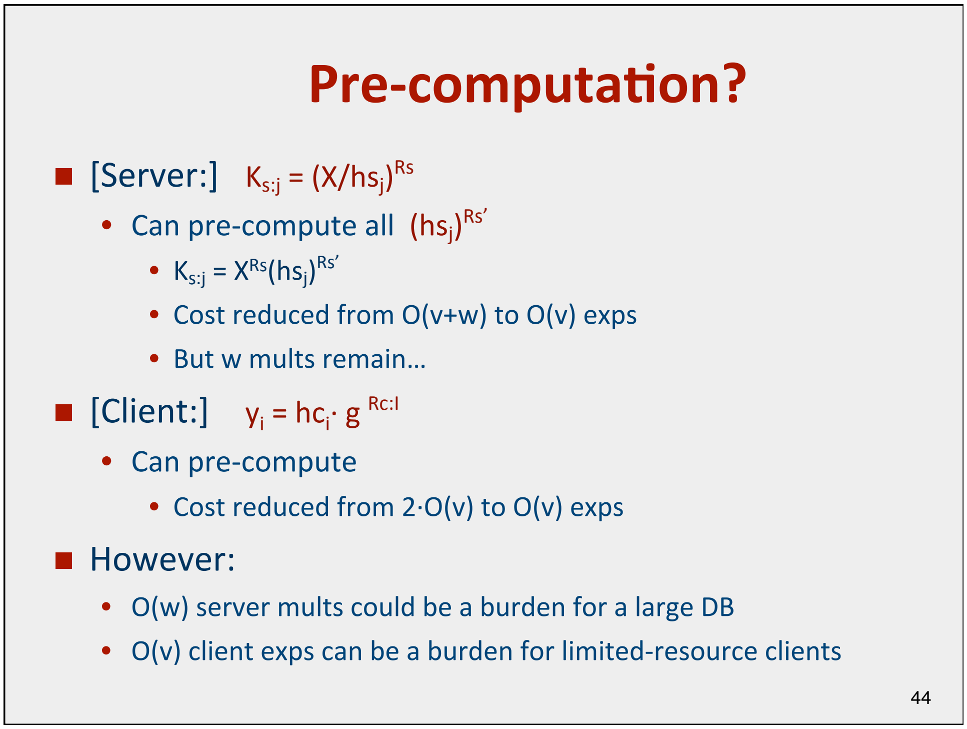
$t_j = H'(K_{s:j})$

$y'_i = (y_i)^{Rs'}$

$Z, \{y_1',\dots,y_v'\}, \{t_1,\dots,t_w\}$

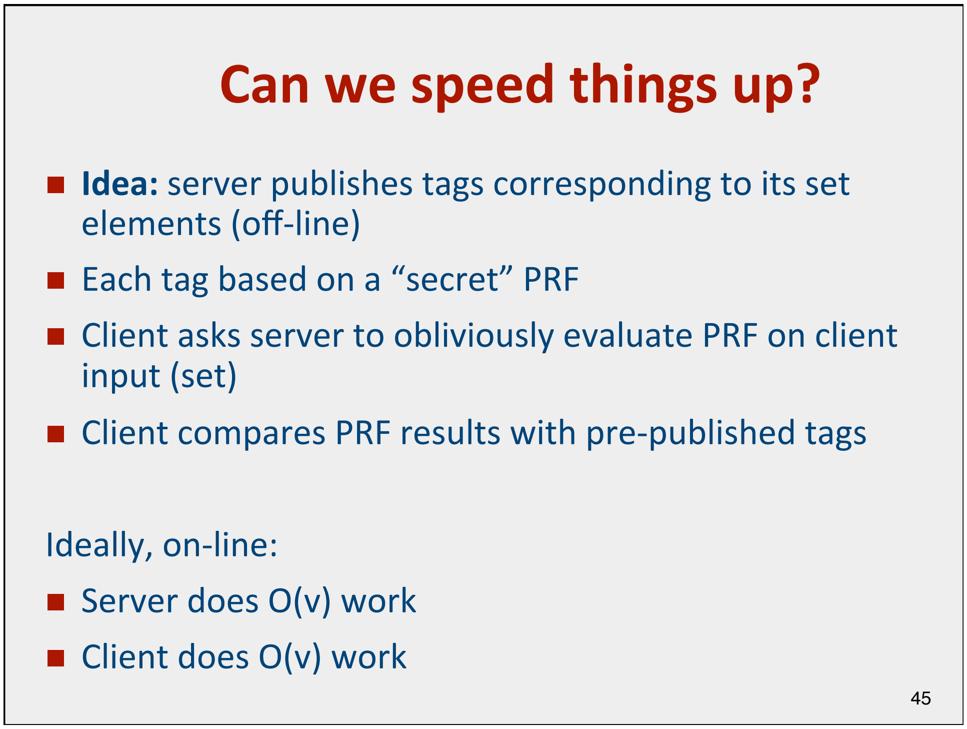
$K_{c:i} = y_i' \cdot Z^{Rc} \cdot Z^{-Rc:i}$

$t'_i = H'(K_{c:i})$



Pre-computation?

- [Server:] $K_{s:j} = (X/hs_j)^{Rs}$
  - Can pre-compute all $(hs_j)^{Rs'}$
    - $K_{s:j} = X^{Rs}(hs_j)^{Rs'}$
    - Cost reduced from O(v+w) to O(v) exps
    - But w mults remain…
- [Client:] $y_i = hc_i \cdot g^{Rc:I}$
  - Can pre-compute
    - Cost reduced from 2·O(v) to O(v) exps
- However:
  - O(w) server mults could be a burden for a large DB
  - O(v) client exps can be a burden for limited-resource clients

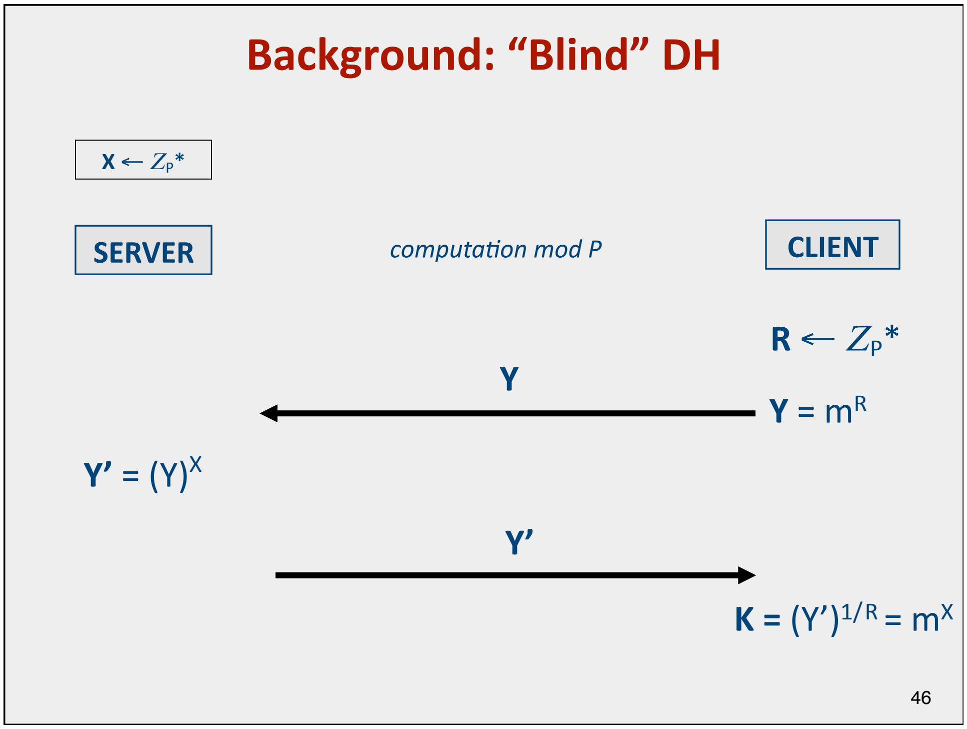


Can we speed things up?

- **Idea:** server publishes tags corresponding to its set elements (off-line)
- Each tag based on a “secret” PRF
- Client asks server to obliviously evaluate PRF on client input (set)
- Client compares PRF results with pre-published tags

Ideally, on-line:

- Server does O(v) work
- Client does O(v) work



Background: “Blind” DH

$\mathbf{X} \leftarrow \mathbb{Z}_P^*$

SERVER

computation mod P

CLIENT

$\mathbf{R} \leftarrow \mathbb{Z}_P^*$

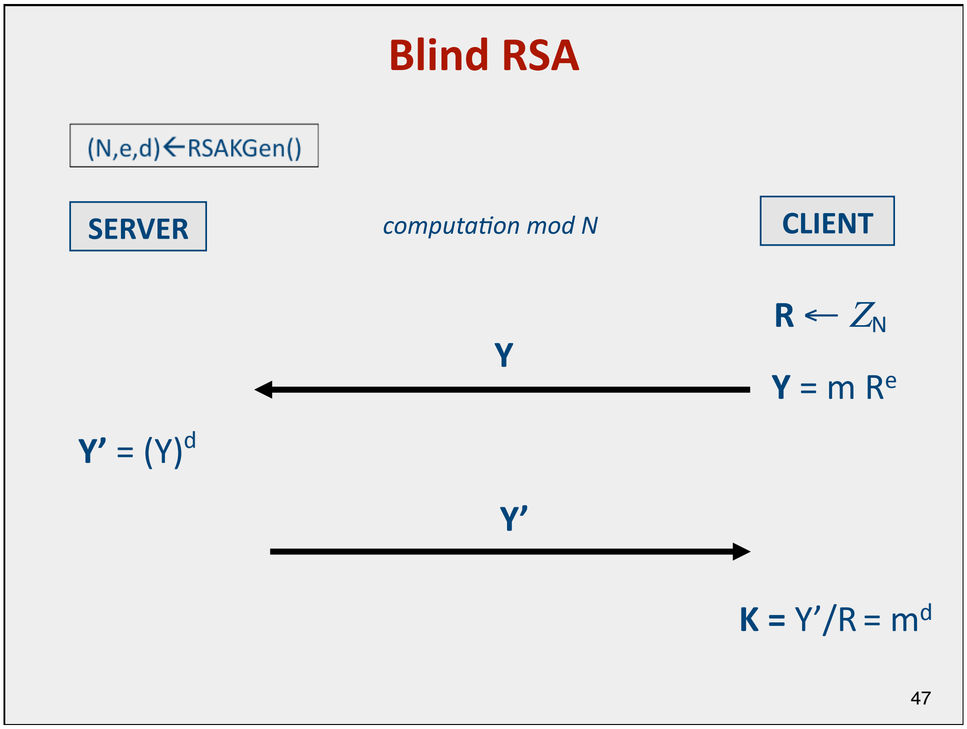
$\mathbf{Y} = m^R$

$\xleftarrow{\mathbf{Y}}$

$\mathbf{Y'} = (Y)^X$

$\xrightarrow{\mathbf{Y'}}$

$\mathbf{K} = (Y')^{1/R} = m^X$



Blind RSA

(N,e,d)←RSAKGen()

SERVER

computation mod N

CLIENT

$\mathbf{R} \leftarrow Z_N$

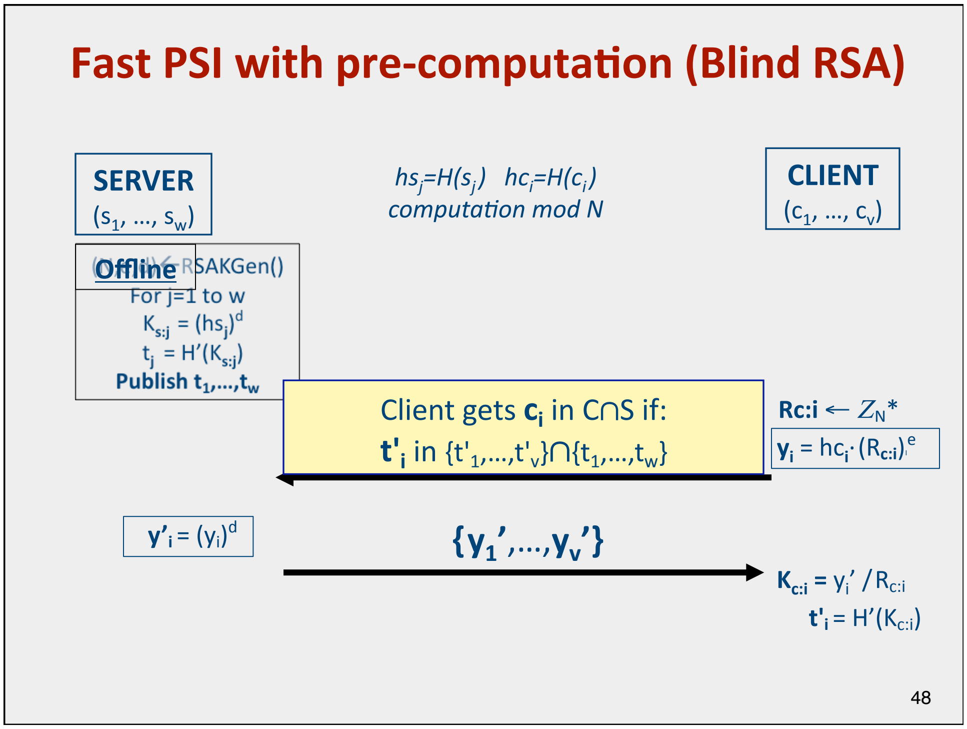
Y

$\mathbf{Y} = m\,R^e$

$\mathbf{Y'} = (Y)^d$

Y’

$\mathbf{K} = Y'/R = m^d$



Fast PSI with pre-computation (Blind RSA)

SERVER

$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$

computation mod N

CLIENT

$(c_1, \ldots, c_v)$

Offline

(N,e,d)←RSAKGen()

For j=1 to w

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

$t_j = H'(K_{s:j})$

Publish $t_1,\ldots,t_w$

$\mathbf{Rc{:}i} \leftarrow Z_N{}^*$

$\mathbf{y_i} = hc_i \cdot (R_{c:i})^e$

Client gets $\mathbf{c_i}$ in C∩S if:

$\mathbf{t'_i}$ in $\{t'_1,\ldots,t'_v\} \cap \{t_1,\ldots,t_w\}$

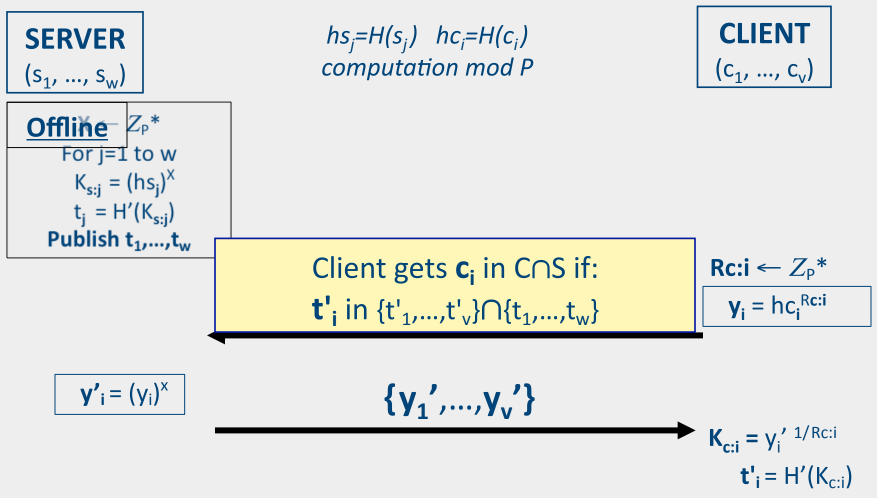
$\mathbf{y'_i} = (y_i)^d$

$\mathbf{\{y_1',\ldots,y_v'\}}$

$\mathbf{K_{c:i}} = y_i' / R_{c:i}$

$\mathbf{t'_i} = H'(K_{c:i})$

Fast PSI with pre-computation (Blind DH)



SERVER
$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$
computation mod P

CLIENT
$(c_1, \ldots, c_v)$

Offline
$X \leftarrow Z_P^*$
For j=1 to w
$K_{s:j} = (hs_j)^X$
$t_j = H'(K_{s:j})$
Publish $t_1,\ldots,t_w$

Client gets $\mathbf{c_i}$ in C∩S if:
$\mathbf{t'_i}$ in $\{t'_1,\ldots,t'_v\} \cap \{t_1,\ldots,t_w\}$

Rc:i $\leftarrow Z_P^*$
$y_i = hc_i^{Rc:i}$

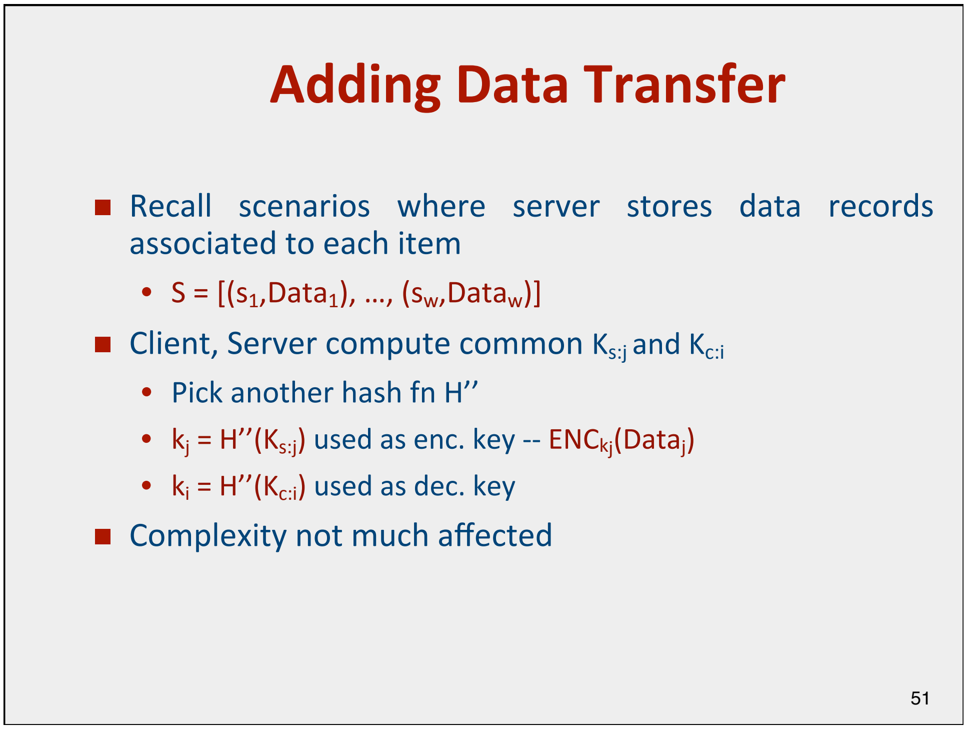
$y'_i = (y_i)^X$

$\{y_1',\ldots,y_v'\}$

$K_{c:i} = y_i'^{\,1/Rc:i}$
$t'_i = H'(K_{c:i})$

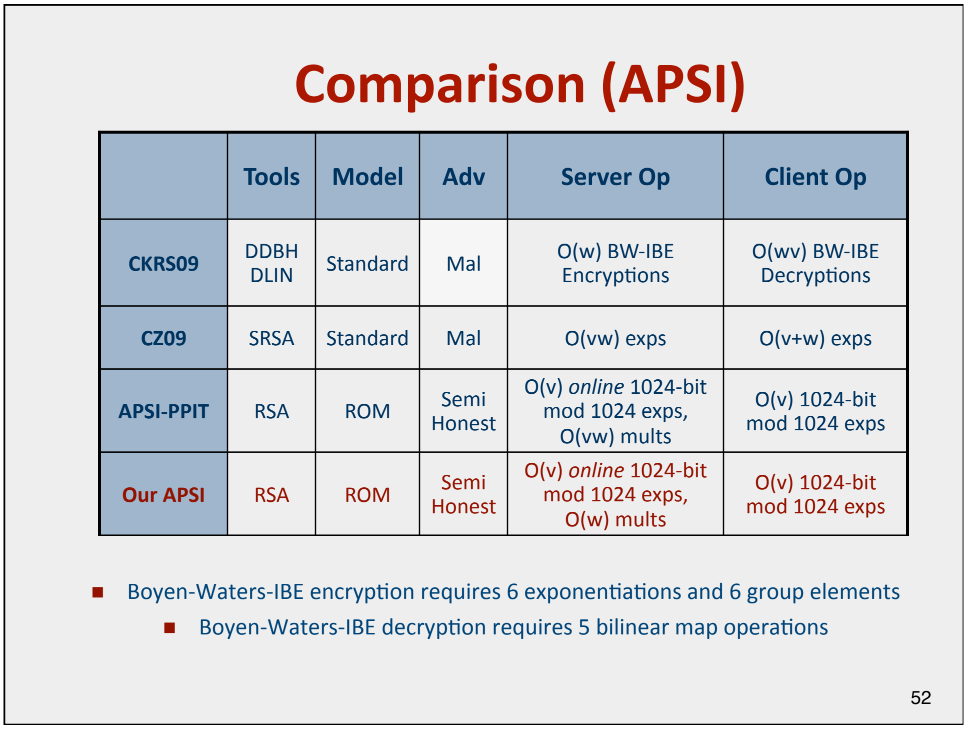
Caveats

- **Server unlinkability** hard to attain
  - Changes in **S** must be propagated
  - Hiding nature of changes not easy
- No easy conversion to APSI



Adding Data Transfer

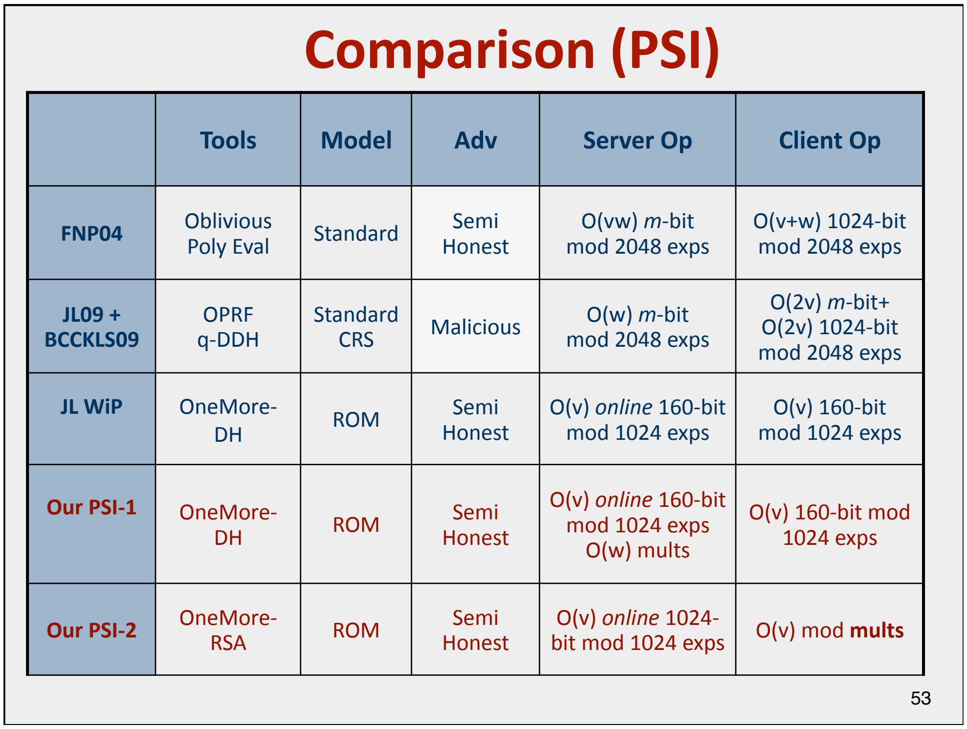
- Recall scenarios where server stores data records associated to each item
  - S = [(s_1,Data_1), …, (s_w,Data_w)]
- Client, Server compute common $K_{s:j}$ and $K_{c:i}$
  - Pick another hash fn H’’
  - k_j = H’’($K_{s:j}$) used as enc. key -- ENC_{kj}(Data_j)
  - k_i = H’’($K_{c:i}$) used as dec. key
- Complexity not much affected



Comparison (APSI)

	Tools	Model	Adv	Server Op	Client Op
CKRS09	DDBH DLIN	Standard	Mal	O(w) BW-IBE Encryptions	O(wv) BW-IBE Decryptions
CZ09	SRSA	Standard	Mal	O(vw) exps	O(v+w) exps
APSI-PPIT	RSA	ROM	Semi Honest	O(v) *online* 1024-bit mod 1024 exps, O(vw) mults	O(v) 1024-bit mod 1024 exps
Our APSI	RSA	ROM	Semi Honest	O(v) *online* 1024-bit mod 1024 exps, O(w) mults	O(v) 1024-bit mod 1024 exps

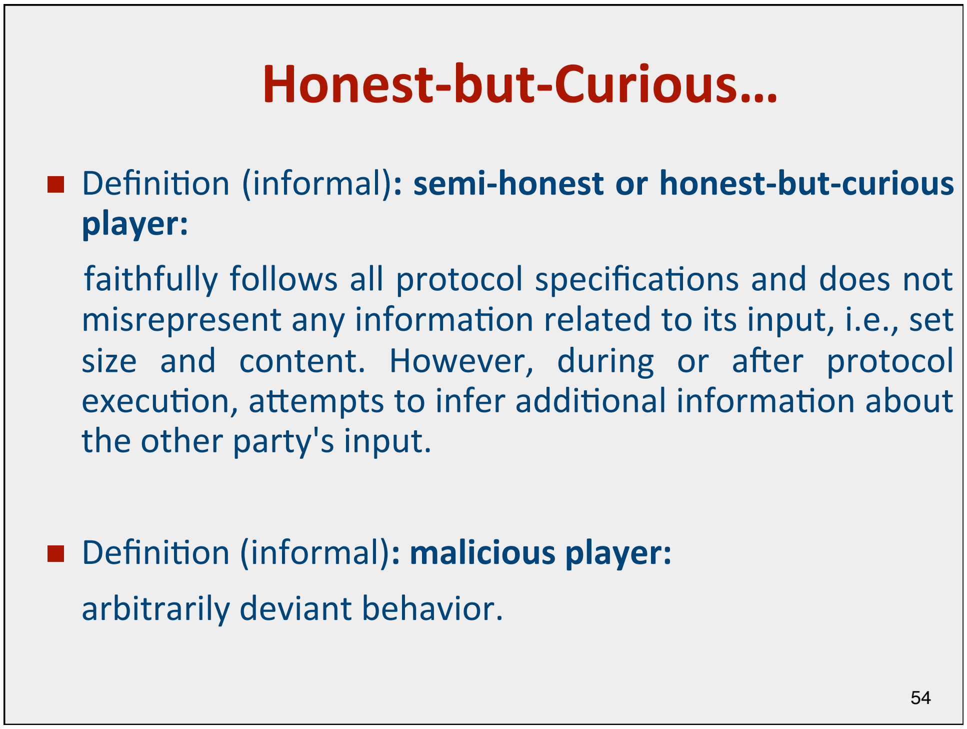
- Boyen-Waters-IBE encryption requires 6 exponentiations and 6 group elements
  - Boyen-Waters-IBE decryption requires 5 bilinear map operations



Comparison (PSI)

	Tools	Model	Adv	Server Op	Client Op
FNP04	Oblivious Poly Eval	Standard	Semi Honest	O(vw) *m*-bit mod 2048 exps	O(v+w) 1024-bit mod 2048 exps
JL09 + BCCKLS09	OPRF q-DDH	Standard CRS	Malicious	O(w) *m*-bit mod 2048 exps	O(2v) *m*-bit+ O(2v) 1024-bit mod 2048 exps
JL WiP	OneMore-DH	ROM	Semi Honest	O(v) *online* 160-bit mod 1024 exps	O(v) 160-bit mod 1024 exps
Our PSI-1	OneMore-DH	ROM	Semi Honest	O(v) *online* 160-bit mod 1024 exps O(w) mults	O(v) 160-bit mod 1024 exps
Our PSI-2	OneMore-RSA	ROM	Semi Honest	O(v) *online* 1024-bit mod 1024 exps	O(v) mod **mults**

53



Honest-but-Curious…

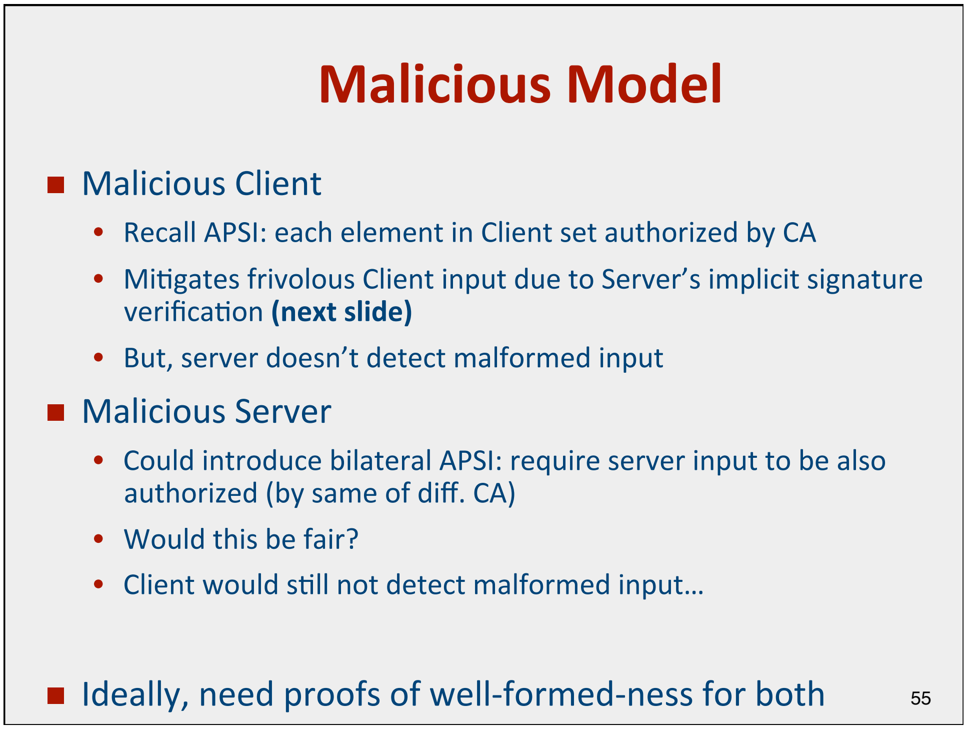
- Definition (informal)**: semi-honest or honest-but-curious player:**

  faithfully follows all protocol specifications and does not misrepresent any information related to its input, i.e., set size and content. However, during or after protocol execution, attempts to infer additional information about the other party's input.

- Definition (informal)**: malicious player:**

  arbitrarily deviant behavior.

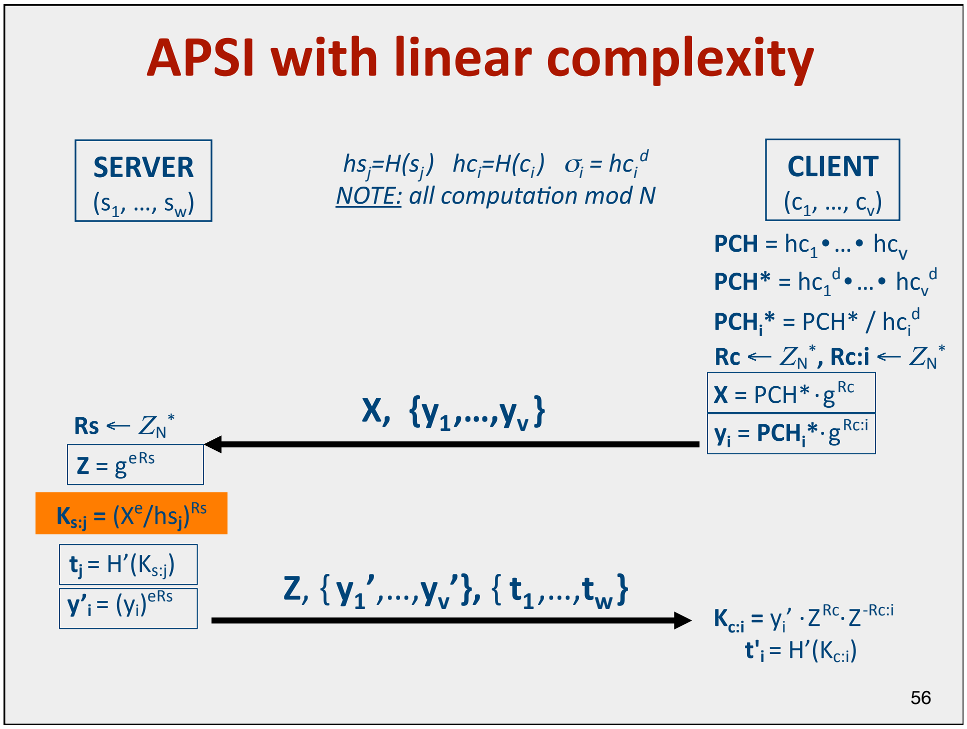
54



Malicious Model

- Malicious Client
  - Recall APSI: each element in Client set authorized by CA
  - Mitigates frivolous Client input due to Server's implicit signature verification **(next slide)**
  - But, server doesn't detect malformed input
- Malicious Server
  - Could introduce bilateral APSI: require server input to be also authorized (by same of diff. CA)
  - Would this be fair?
  - Client would still not detect malformed input…
- Ideally, need proofs of well-formed-ness for both

55



APSI with linear complexity

SERVER $(s_1, \ldots, s_w)$

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

NOTE: all computation mod N

CLIENT $(c_1, \ldots, c_v)$

PCH $= hc_1 \bullet \ldots \bullet hc_v$

PCH* $= hc_1^d \bullet \ldots \bullet hc_v^d$

PCH$_i$* $= PCH^* / hc_i^d$

Rc $\leftarrow Z_N^*$, **Rc:i** $\leftarrow Z_N^*$

$X = PCH^* \cdot g^{Rc}$

$y_i = PCH_i^* \cdot g^{Rc:i}$

$X, \{y_1, \ldots, y_v\}$

Rs $\leftarrow Z_N^*$

$Z = g^{eRs}$

$K_{s:j} = (X^e / hs_j)^{Rs}$

$t_j = H'(K_{s:j})$

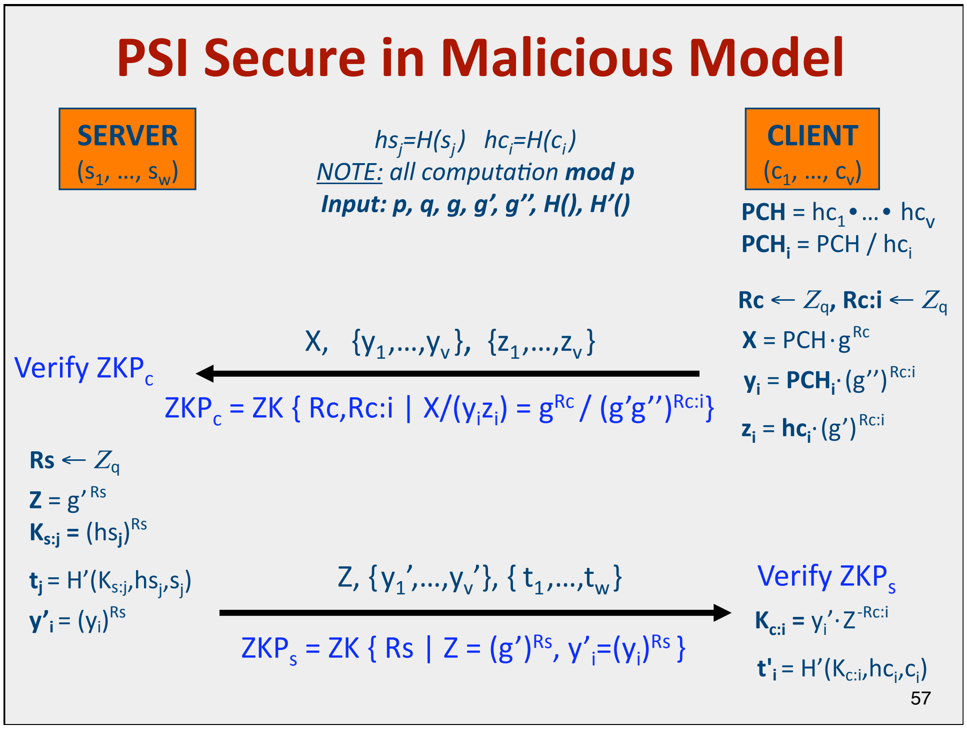
$y'_i = (y_i)^{eRs}$

$Z, \{y_1', \ldots, y_v'\}, \{t_1, \ldots, t_w\}$

$K_{c:i} = y_i' \cdot Z^{Rc} \cdot Z^{-Rc:i}$

$t'_i = H'(K_{c:i})$

56



PSI Secure in Malicious Model

SERVER
$(s_1, \dots, s_w)$
CLIENT
$(c_1, \dots, c_v)$
X, $\{y_1,\dots,y_v\}$, $\{z_1,\dots,z_v\}$
Z, $\{y_1',\dots,y_v'\}$, $\{t_1,\dots,t_w\}$

$hs_j=H(s_j)$ $hc_i=H(c_i)$

*NOTE: all computation **mod p***

Input: p, q, g, g', g'', H(), H'()

$\mathbf{PCH} = hc_1 \bullet \dots \bullet hc_v$

$\mathbf{PCH_i} = PCH / hc_i$

$\mathbf{Rc} \leftarrow Z_q$, $\mathbf{Rc{:}i} \leftarrow Z_q$

$\mathbf{X} = PCH \cdot g^{Rc}$

$\mathbf{y_i} = \mathbf{PCH_i} \cdot (g'')^{Rc:i}$

$\mathbf{z_i} = \mathbf{hc_i} \cdot (g')^{Rc:i}$

Verify ZKP_c

$ZKP_c = ZK \{ Rc, Rc{:}i \mid X/(y_i z_i) = g^{Rc} / (g'g'')^{Rc:i} \}$

$\mathbf{Rs} \leftarrow Z_q$

$\mathbf{Z} = g'^{Rs}$

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

$\mathbf{t_j} = H'(K_{s:j}, hs_j, s_j)$

$\mathbf{y'_i} = (y_i)^{Rs}$

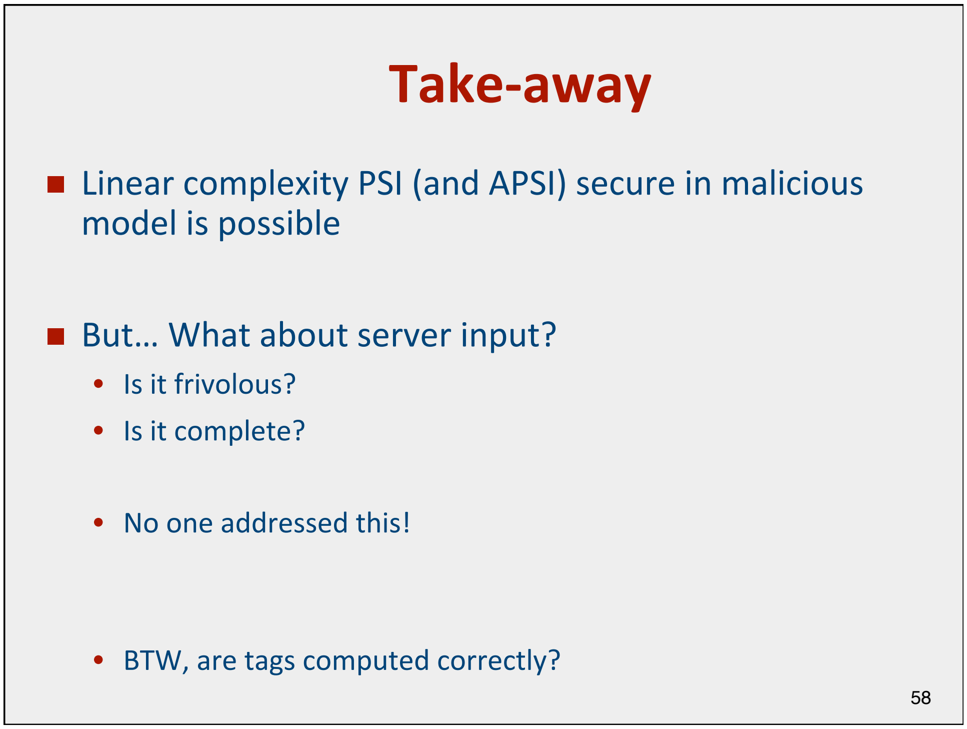
$ZKP_s = ZK \{ Rs \mid Z = (g')^{Rs}, y'_i=(y_i)^{Rs} \}$

Verify ZKP_s

$\mathbf{K_{c:i}} = y_i' \cdot Z^{-Rc:i}$

$\mathbf{t'_i} = H'(K_{c:i}, hc_i, c_i)$

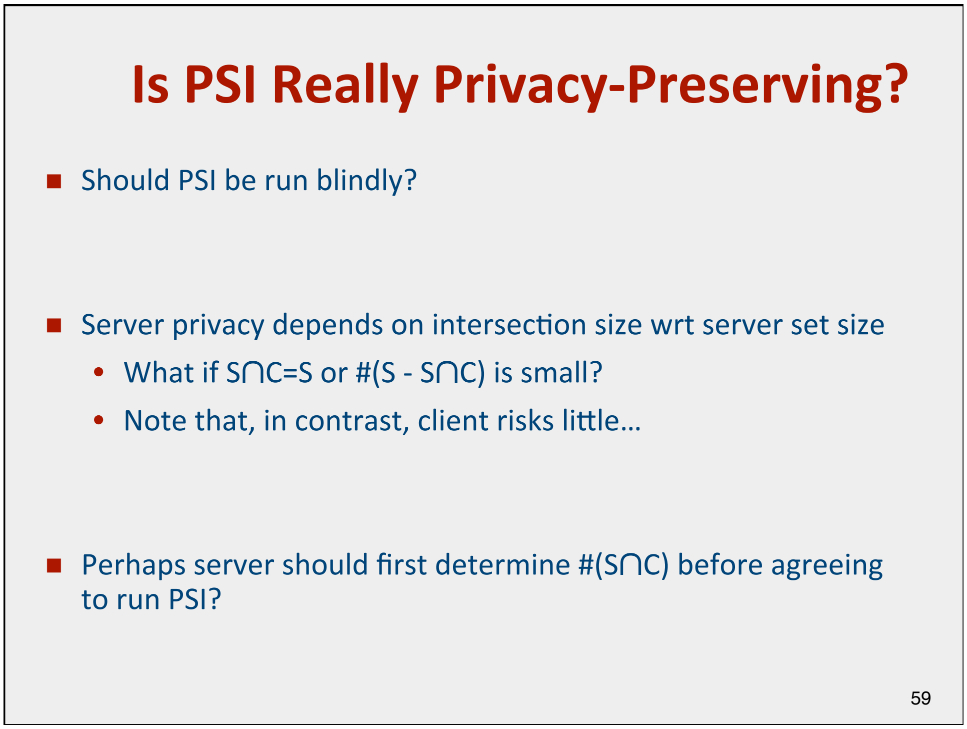
57



Take-away

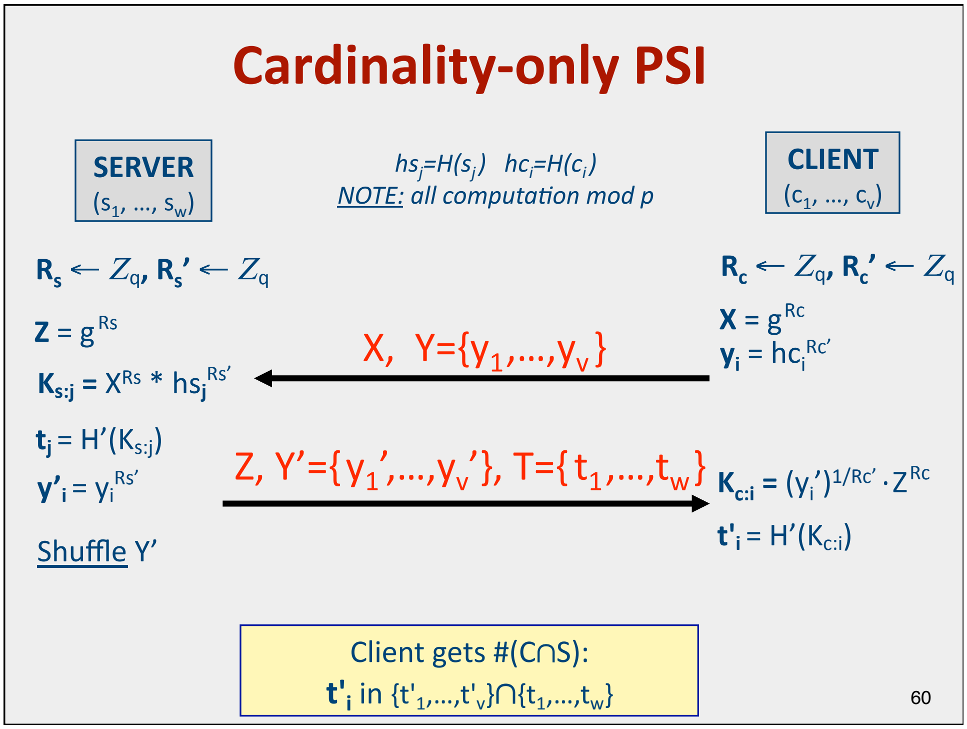
- Linear complexity PSI (and APSI) secure in malicious model is possible
- But… What about server input?
  - Is it frivolous?
  - Is it complete?
  - No one addressed this!
  - BTW, are tags computed correctly?

58



Is PSI Really Privacy-Preserving?

- Should PSI be run blindly?
- Server privacy depends on intersection size wrt server set size
  - What if S∩C=S or #(S - S∩C) is small?
  - Note that, in contrast, client risks little…
- Perhaps server should first determine #(S∩C) before agreeing to run PSI?



Cardinality-only PSI

SERVER
$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$
NOTE: all computation mod p

CLIENT
$(c_1, \ldots, c_v)$

Client: $R_c \leftarrow Z_q$, $R_c' \leftarrow Z_q$

$X = g^{Rc}$

$y_i = hc_i^{Rc'}$

Client → Server: X, $Y=\{y_1,\ldots,y_v\}$

Server: $R_s \leftarrow Z_q$, $R_s' \leftarrow Z_q$

$Z = g^{Rs}$

$K_{s:j} = X^{Rs} * hs_j^{Rs'}$

$t_j = H'(K_{s:j})$

$y'_i = y_i^{Rs'}$

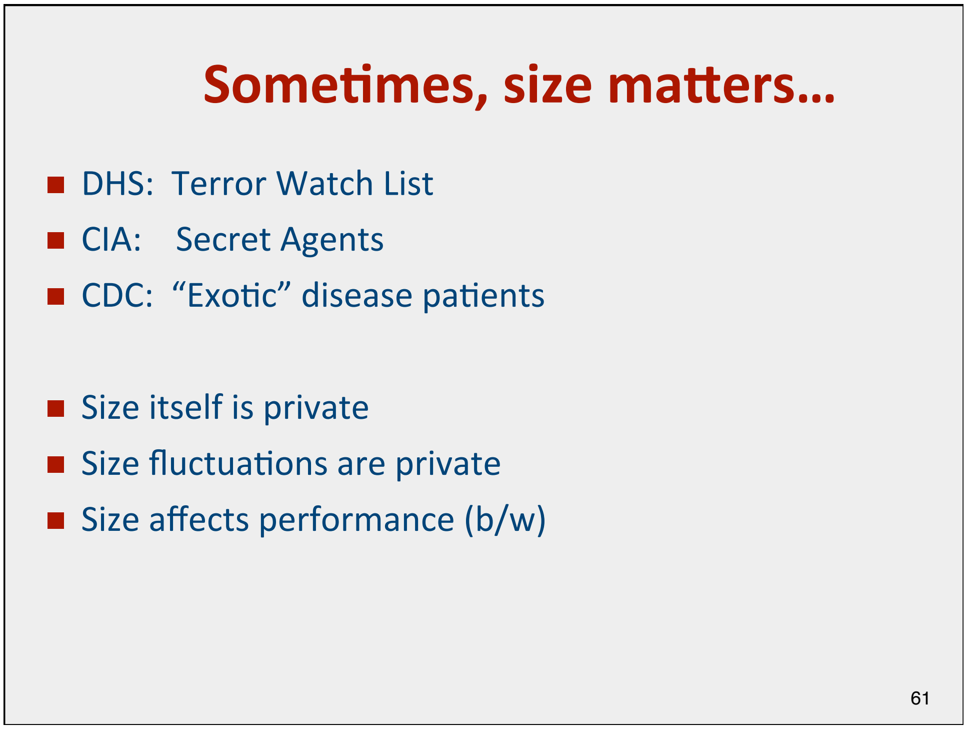
Shuffle Y'

Server → Client: Z, $Y'=\{y_1',\ldots,y_v'\}$, $T=\{t_1,\ldots,t_w\}$

Client: $K_{c:i} = (y_i')^{1/Rc'} \cdot Z^{Rc}$

$t'_i = H'(K_{c:i})$

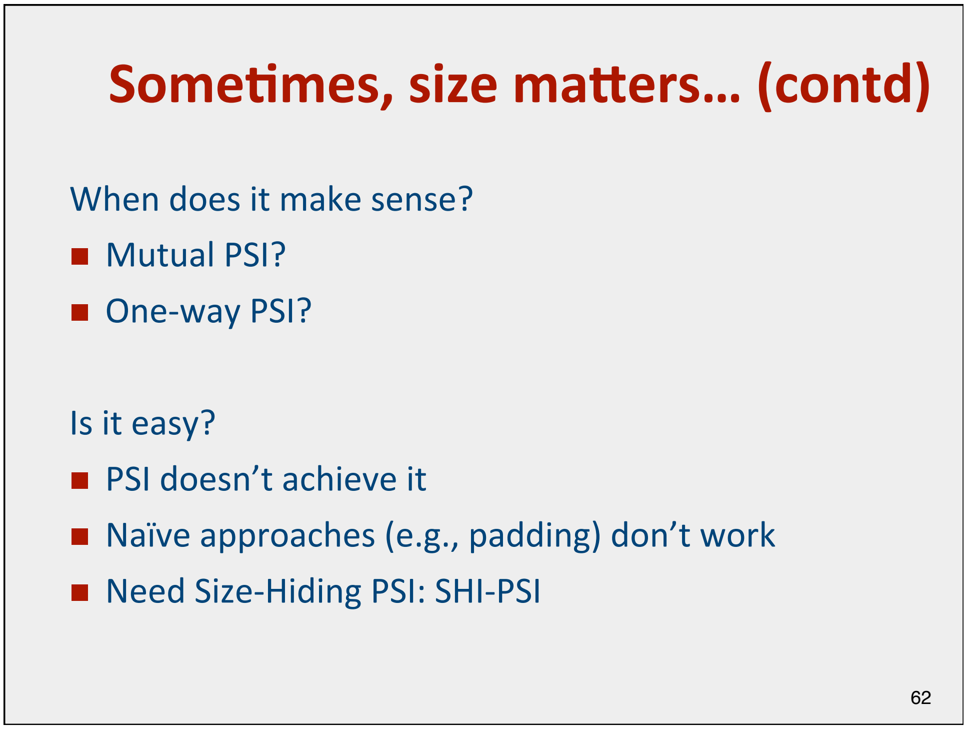
Client gets #(C∩S):
t'_i in $\{t'_1,\ldots,t'_v\} \cap \{t_1,\ldots,t_w\}$



Sometimes, size matters…

- DHS: Terror Watch List
- CIA: Secret Agents
- CDC: “Exotic” disease patients

- Size itself is private
- Size fluctuations are private
- Size affects performance (b/w)



Sometimes, size matters… (contd)

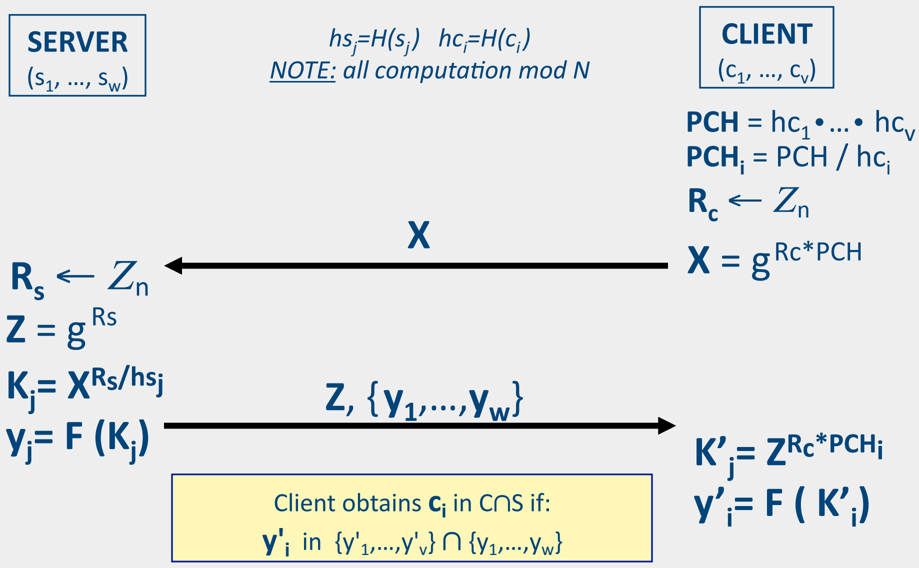
When does it make sense?

- Mutual PSI?
- One-way PSI?

Is it easy?

- PSI doesn’t achieve it
- Naïve approaches (e.g., padding) don’t work
- Need Size-Hiding PSI: SHI-PSI

SHI-PSI (fully unlinkable)



SERVER

$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$

<u>NOTE:</u> all computation mod N

CLIENT

$(c_1, \ldots, c_v)$

$\mathbf{PCH} = hc_1 \bullet \ldots \bullet hc_v$

$\mathbf{PCH_i} = PCH / hc_i$

$\mathbf{R_c} \leftarrow Z_n$

$\mathbf{X} = g^{Rc*PCH}$

$\mathbf{X}$

$\mathbf{R_s} \leftarrow Z_n$

$\mathbf{Z} = g^{Rs}$

$\mathbf{K_j = X^{Rs/hs_j}}$

$\mathbf{y_j = F\,(K_j)}$

$\mathbf{Z}, \{\mathbf{y_1,\ldots,y_w}\}$

$\mathbf{K'_j = Z^{Rc*PCH_i}}$

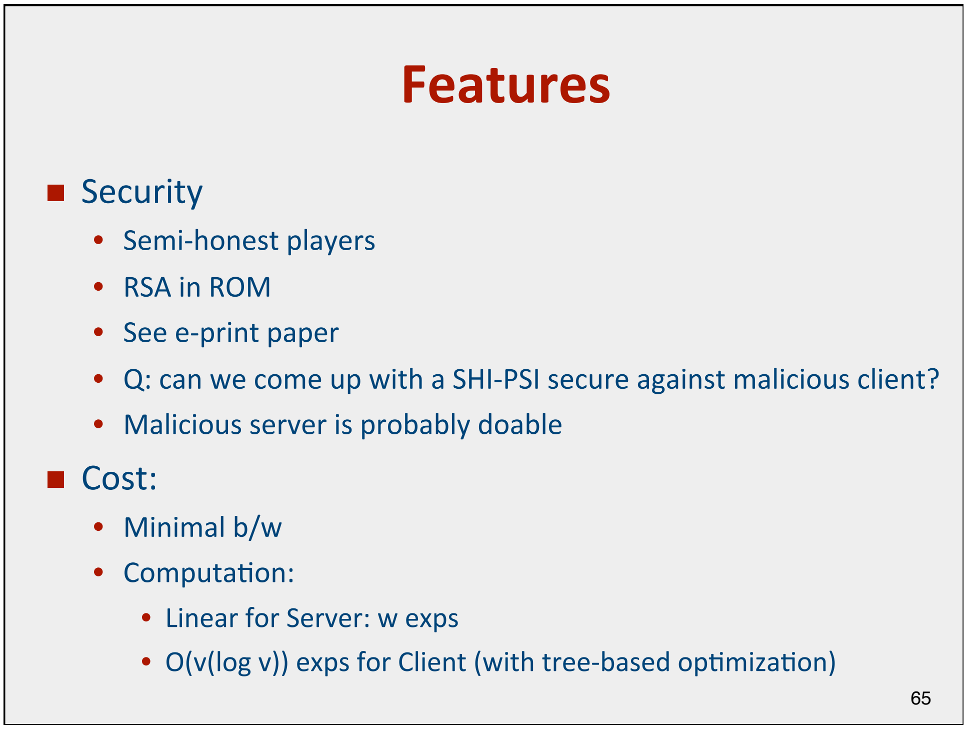
$\mathbf{y'_i = F\,(\,K'_i)}$

Client obtains $\mathbf{c_i}$ in C∩S if:

$\mathbf{y'_i}$ in $\{y'_1,\ldots,y'_v\} \cap \{y_1,\ldots,y_w\}$

Notes

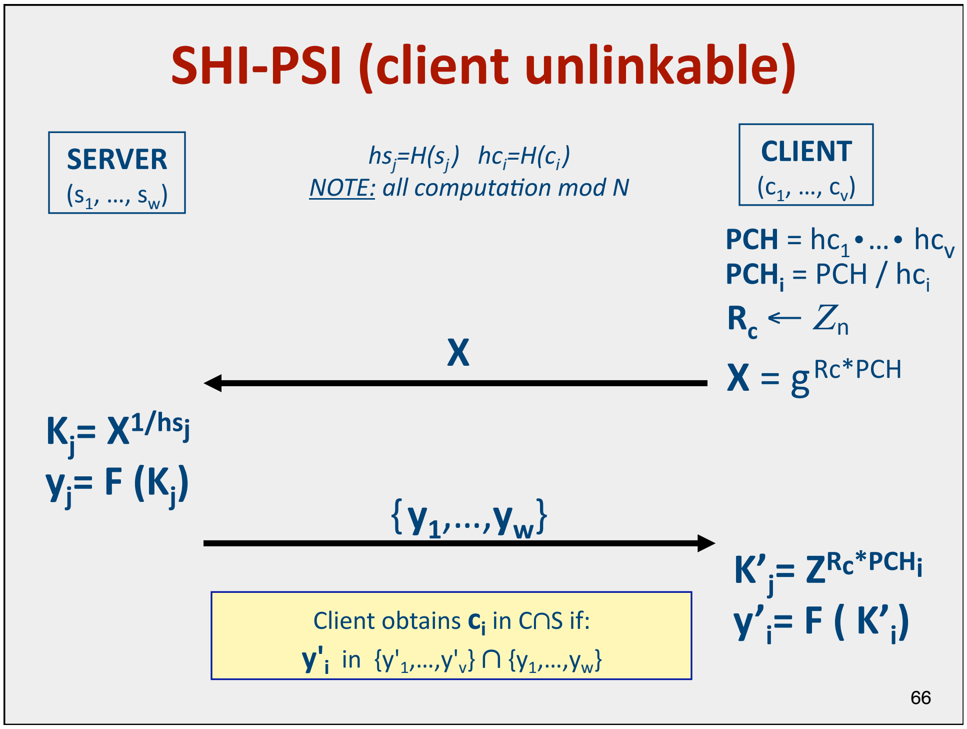
- Client (obviously) does not know phi(N)
- So, $\mathbf{X} = g^{Rc*PCH}$ is computed piece-meal (v exps)
- And, each $\mathbf{K'_j = Z^{Rc*PCH_i}}$ costs v exps
  - Resulting in … $O(v^2)$ exps total



Features

- Security
  - Semi-honest players
  - RSA in ROM
  - See e-print paper
  - Q: can we come up with a SHI-PSI secure against malicious client?
  - Malicious server is probably doable
- Cost:
  - Minimal b/w
  - Computation:
    - Linear for Server: w exps
    - O(v(log v)) exps for Client (with tree-based optimization)

65



SHI-PSI (client unlinkable)

SERVER
$(s_1, \ldots, s_w)$

$hs_j=H(s_j)$ $hc_i=H(c_i)$
NOTE: all computation mod N

CLIENT
$(c_1, \ldots, c_v)$

PCH = $hc_1 \bullet \ldots \bullet hc_v$
PCH$_i$ = PCH / hc_i
$\mathbf{R_c} \leftarrow Z_n$
$\mathbf{X} = g^{Rc*PCH}$

$\mathbf{K_j = X^{1/hs_j}}$
$\mathbf{y_j = F\,(K_j)}$

$\mathbf{K'_j = Z^{Rc*PCH_i}}$
$\mathbf{y'_i = F\,(\,K'_i)}$

Client obtains $\mathbf{c_i}$ in C∩S if:
$\mathbf{y'_i}$ in $\{y'_1,\ldots,y'_v\} \cap \{y_1,\ldots,y_w\}$

66

Real World

- How to support SQL-type queries, e.g.

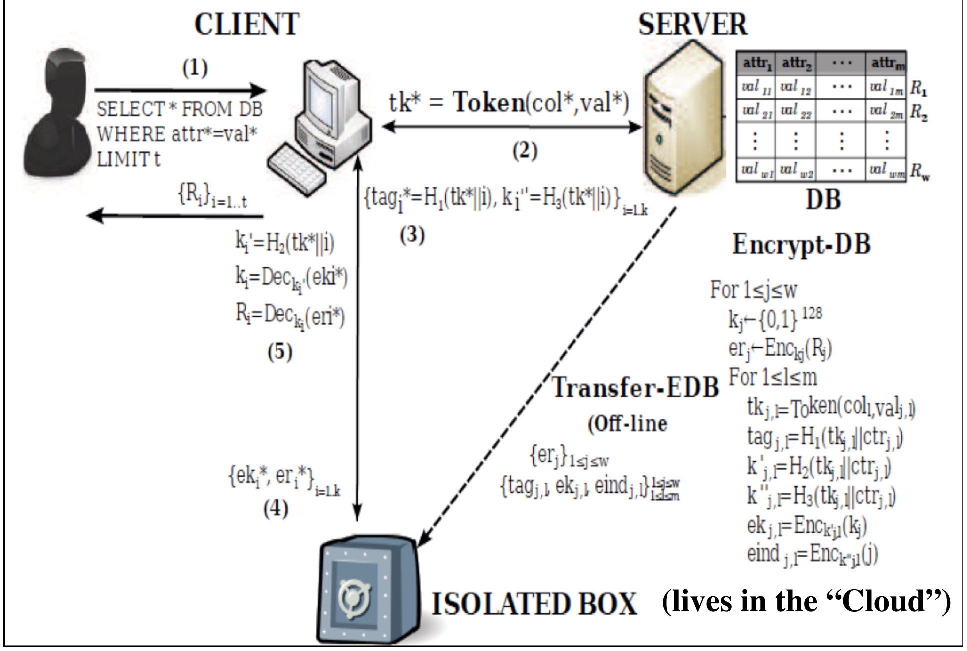
"SELECT * FROM DB WHERE attr*=value*"

Issues:

- Multiple record match (multi-sets): patterns exposed?
- Any attribute can be queried: how to encrypt?
- Server must transfer entire encrypted DB
  - Bandwidth to (and storage on) client
- Liability of possessing (even encrypted) data

67

PSST Architecture


CLIENT
SERVER
(1)
SELECT * FROM DB WHERE attr*=val* LIMIT t
$\{R_i\}_{i=1..t}$
$tk^* = \textbf{Token}(col^*, val^*)$
(2)
$\{tag_i^* = H_1(tk^*||i), k_i'' = H_3(tk^*||i)\}_{i=1.k}$
(3)
$k_i' = H_2(tk^*||i)$
$k_i = Dec_{k_i'}(eki^*)$
$R_i = Dec_{k_i}(eri^*)$
(5)
$\{ek_i^*, er_i^*\}_{i=1.k}$
(4)
DB
Encrypt-DB
Transfer-EDB
(Off-line
$\{er_j\}_{1 \le j \le w}$
$\{tag_{j,l}, ek_{j,l}, eind_{j,l}\}_{1 \le j \le w, 1 \le l \le m}$
ISOLATED BOX

attr₁	attr₂	⋯	attrₘ	
val_{11}	val_{12}	⋯	val_{1m}	R_1
val_{21}	val_{22}	⋯	val_{2m}	R_2
⋮	⋮	⋮	⋮	
val_{w1}	val_{w2}	⋯	val_{wm}	R_w

```
For $1 \le j \le w$
  $k_j \leftarrow \{0,1\}^{128}$
  $er_j \leftarrow Enc_{kj}(R_j)$
  For $1 \le l \le m$
    $tk_{j,l} = Token(col_l, val_{j,l})$
    $tag_{j,l} = H_1(tk_{j,l}||ctr_{j,l})$
    $k'_{j,l} = H_2(tk_{j,l}||ctr_{j,l})$
    $k''_{j,l} = H_3(tk_{j,l}||ctr_{j,l})$
    $ek_{j,l} = Enc_{k'_{jl}}(k_j)$
    $eind_{j,l} = Enc_{k''_{jl}}(j)$
```

(lives in the "Cloud")

Conclusions

- PPIT, Practical APSI + PSI and SHI-PSI
- Future + on-going work
  - Security in Malicious Model
  - Group PSI
  - Size-Hiding PSI
  - Cardinality-only PSI, server-APSI, subset testing
  - **Building real systems**
- References:
  - *"Policy-Based Privacy-Preserving Information Transfer"*, PETS'09.
  - *"Efficient Private Set Intersection"*, FC'10.
  - "*(If) Size Matters: Size-Hiding Private Set Intersection*", Crypto ePrint Archive: 2010/220, in submission.
  - *"Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model"*, Asiacrypt'10.
  - *"Privacy-Preserving Sharing of Sensitive Information is (Really) Practical"*, Crypto ePrint Archive: 2010/471, in submission.

Thank you!

WE'RE SUPPOSED TO KNOCK AND ANNOUNCE.
YEAH, SURE.
4Th AMENDMENT PRIVACY RIGHTS
...BUT IT'S OK IF YOU DON'T. -SUPREME COURT
POWELL 6/17/06

Image from truthdig.com