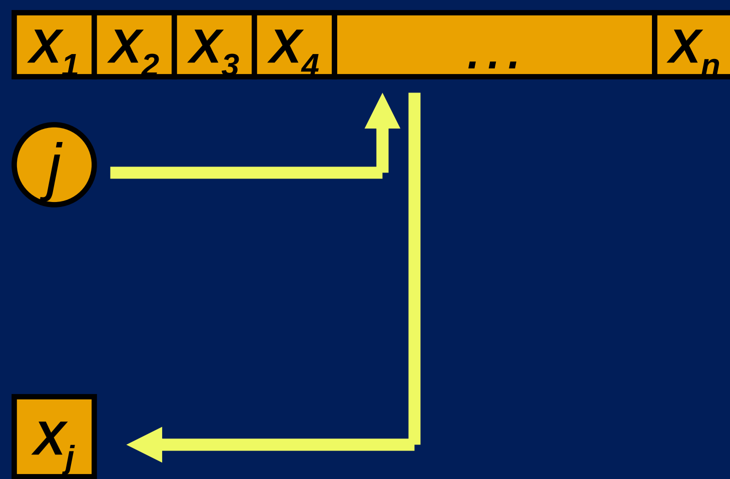
Keyword Search and Oblivious Pseudo-Random Functions

Mike Freedman
NYU

Yuval Ishai, Benny Pinkas, Omer Reingold

Background: Oblivious Transfer

- Oblivious Transfer (OT) [R], 1-out-of-N [EGL]:
  - Input:
    - Server: $x_1, x_2, \ldots, x_n$
    - Client: $1 \leq j \leq n$
  - Output:
    - Server: nothing
    - Client: x_j
  - Privacy:
    - Server learns nothing about j
    - Client learns nothing about x_i for $i \neq j$
  - Well-studied, good solutions: O(n) overhead



Background: Private Information Retrieval (PIR)

- Private Information Retrieval (PIR) [CGKS,KO]
  - Client hides which element retrieved
  - Client can learn more than a single x_j
  - $o(N)$ communication, $O(N)$ computation

- Symmetric Private Information Retrieval (SPIR) [GIKM,NP]
  - PIR in which client learns *only* x_j
  - Hence, privacy for both client and server
  - “OT with sublinear communication”

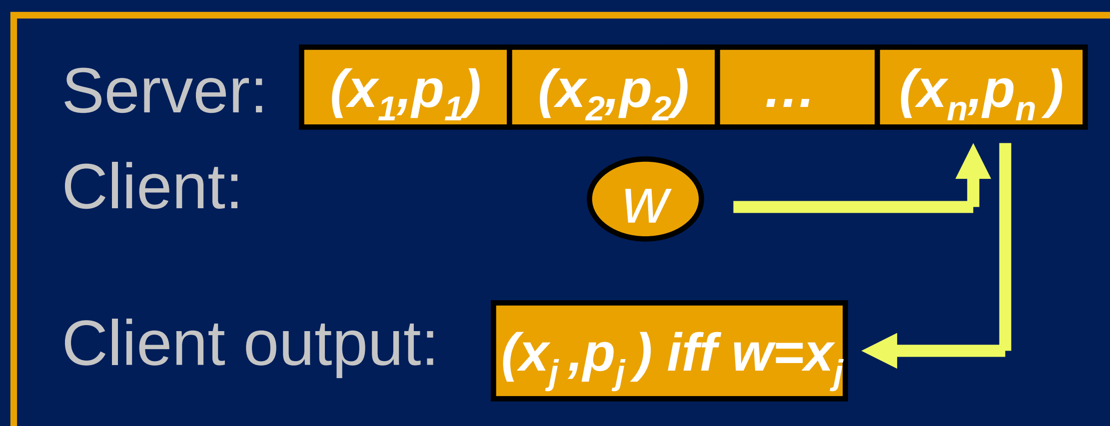
Motivation: Sometimes, OT is not enough

- Bob ("Application Service Provider")
  - Advises merchants on credit card fraud
  - Keeps list of fraudulent card numbers
- Alice ("Merchant")
  - Received a credit card, wants to check if fraudulent
  - Wants to hide credit-card details from Bob, vice-versa
- Use OT?
  - Table of $10^{16} \approx 2^{53}$ entries, 1 if fraudulent, 0 otherwise?

Keyword Search (KS): definition

- Input:
  - Server: database $X=\{ (x_i,p_i) \}, 1 \le i \le N$
    - x_i is a keyword (e.g. number of a corrupt card)
    - p_i is the payload (e.g. why card is corrupt)
  - Client: search word w (e.g. credit card number)

- Output:
  - Server: nothing
  - Client:
    - p_i if $\exists i : x_i = w$
    - otherwise nothing



Keyword Search from data structures? [KO,CGN]

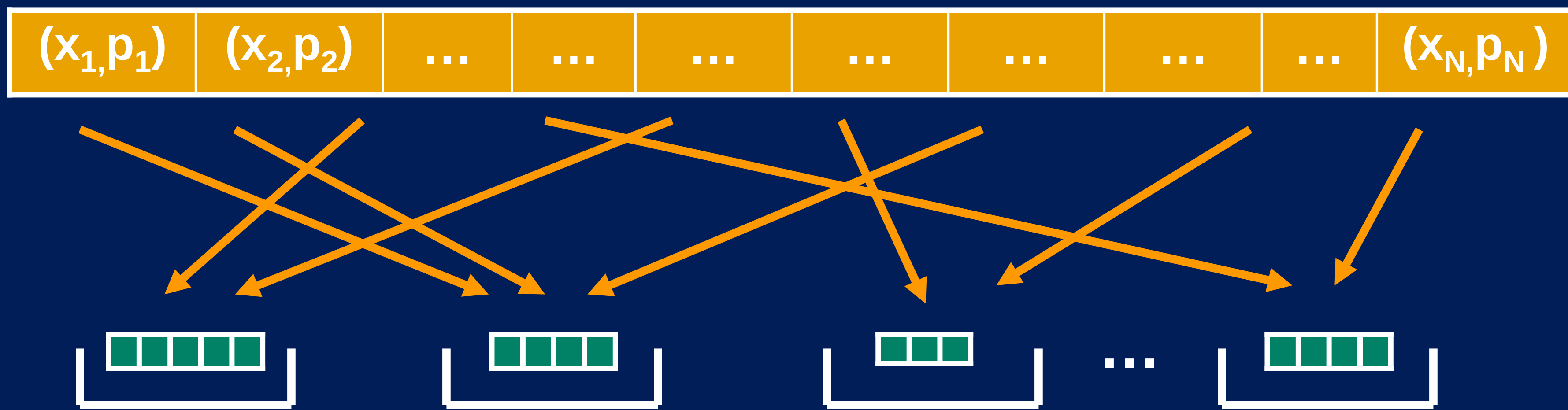
- Take any efficient query-able data structure
  - Hash table, search tree, trie, etc.

- Replace direct query with OT / PIR

- Achieves client privacy

We’re done?

Keyword Search from hashing + OT [KO]



- Use hash function *H* to map (x_i,p_i) *to bin* $H(x_i)$
- Client uses OT to read bin *H(w)*

- Multiple per bin: no server privacy: client gets > 1 elt
- One per bin, N bins: no server privacy: H leaks info
- One per bin, >> N bins: not efficient

Keyword Search

- Variants
  - Multiple queries
  - Adaptive queries
  - Allowing setup
  - Malicious parties

- Prior Work
  - OT + Hashing = KS without server privacy [KO]
    - Add server privacy using trie and many rounds [CGN]
  - Adaptive KS [OK]
    - But, setup with linear communication, RO model, one-more-RSA-inversion assumption

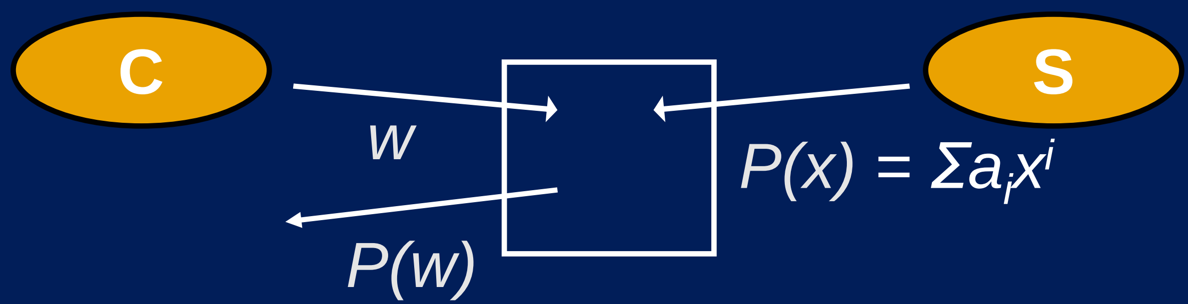
Keyword Search: Results

- Specific protocols for KS
  - One-time KS based on OPE (homomorphic encryption)
  - First 1-round KS with sublinear communication
- Adaptive KS by generic reduction
  - Semi-private KS + oblivious PRFs
- New notions and constructions of OPRFs
  - Fully-adaptive (DDH- or factoring-specific)
  - T-time adaptive (black-box use of OT)

Keyword Search based on Oblivious Evaluation of Polynomials

Specific KS protocols using polynomials

- Tool: Oblivious Polynomial Evaluation (OPE) [NP]



  – Privacy: Server: nothing about *w*. Client: nothing but *P(w)*

1-round KS protocol using polynomials

- OPE implementation based on homomorphic encryption
  - Given $E(x)$, $E(y)$, can compute $E(x+y)$, $E(c \cdot x)$, w/o secret key

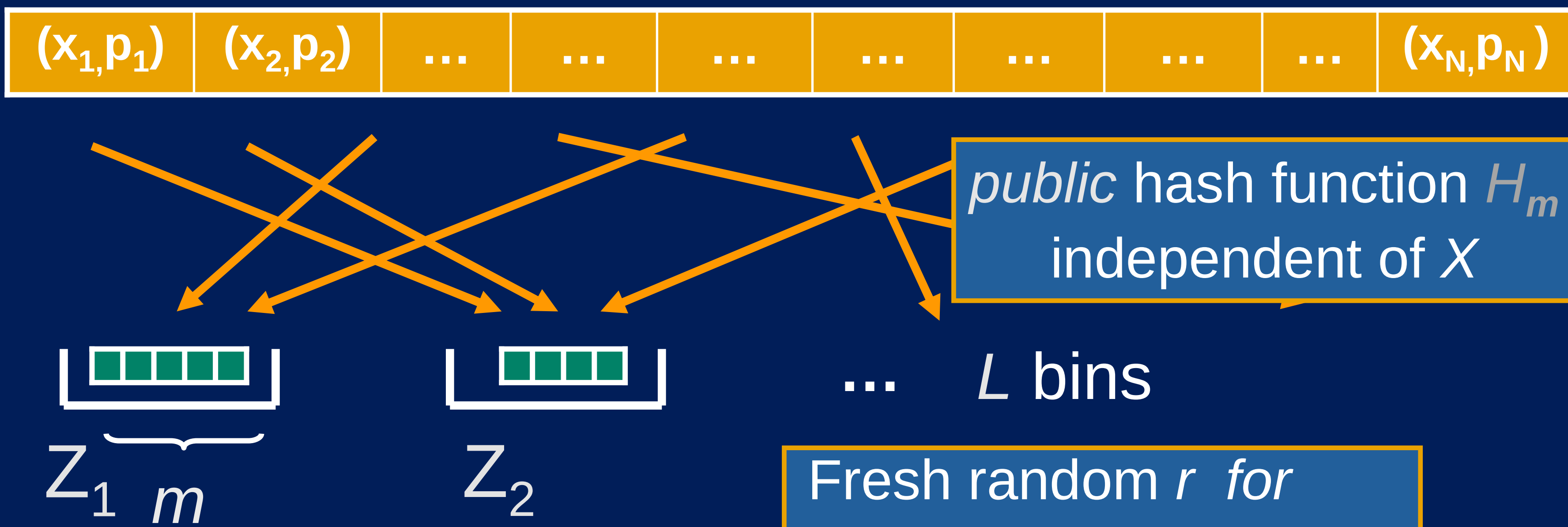
- Server defines on input $X=\{(x_i, p_i)\}$,
  - $Z(x) \;=\; r \cdot P(x) \;+\; Q(x)$, with fresh random r $\forall$ x_i

If $x_i \in X$: $\quad 0 \quad + \quad p_i \,|\, 0^k$

If $x_i \notin X$: $\quad$ rand

- Client/server run OPE of $Z(w)$, overhead O(N)
  - C sends: $E(w), E(w^2), \ldots, E(w^d), PK$
  - S returns: $E(r \cdot \Sigma p_i w^i + \Sigma q_i w^i) = E(r \cdot P(w) + Q(w)) = E(Z(w))$

Reducing the overhead using hashing…



Fresh random r *for*

$Z_j(x) = r \cdot P_j(x) + Q_j(x)$

- Client sends input for L OPE's of degree *m*
- Server has $E(Z_1(w)), \ldots, E(Z_L(w))$
- Client uses PIR to obtain OPE output from bin $H(w)$

- Comm: O(m = log N) + PIR overhead (polylog N)
- Comp: O(N) server, O(m = log N) client

What about malicious parties?

- Efficient 1 round protocol for non-adaptive KS
  - Only consider privacy: server need not commit or know DB
  - Similar relaxation used before in like contexts (PIR, OT)
- Privacy against a malicious server?
  - Server only sees client's interaction in an OT / PIR protocol
- Malicious clients?
  - Message in OPE might not correspond to polynomial values
  - Can enforce correct behavior with about same overhead
  - 1 OPE of degree-m polynomials $\rightarrow$ m OPEs of linear poly's

Keyword Search based on Oblivious Evaluation of Pseudo-Random Functions

Semi-Private Keyword Search

- Goal: Obtain KS from semi-private KS + OPRF

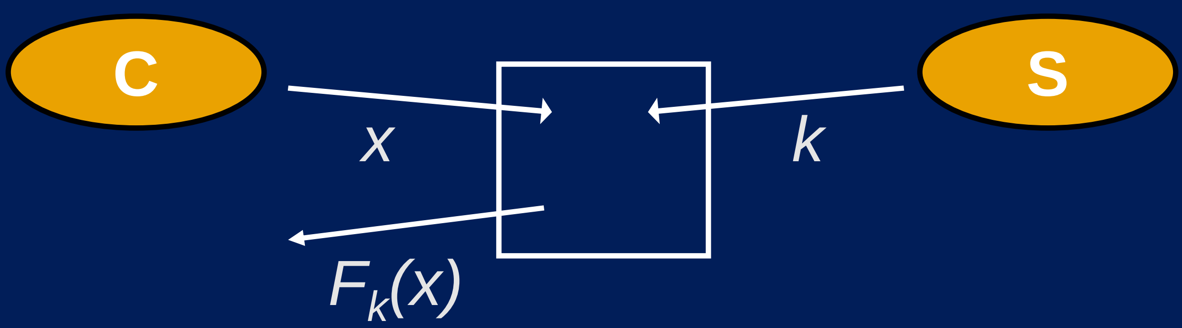
- Semi-Private Keyword Search (PERKY [CGN])
  - Provides privacy for client but not for server
  - Similar privacy to that of PIR

- Examples
  - Send database to client: *O(N)* communication
  - Hash-based solutions + PIR to obtain bin
  - Use any fancy data structure + PIR to query

Oblivious Evaluation of Pseudo-Random Functions

- Pseudo-Random Function: $F_k : \{0,1\}^n \rightarrow \{0,1\}^n$
  - Keyed by k (chooses a specific instantiation of F)
  - Without k, the output of F_k cannot be distinguished from that of a random function

- Oblivious evaluation of a PRF (OPRF)



  - Client: PRF output, nothing about k
  - Server: Nothing

KS from Semi-Private KS + OPRF

(x_1,p_1)	(x_2,p_2)	…	…	…	…	…	…	…	(x_N,p_N)

S chooses k,
defining $F_k(\cdot)$

$\forall\ (x_i,\ p_i) \in X,$

Let $\underline{x'_i \mid p'_i \leftarrow F_k(x_i)}$

Let $\boxed{(\underline{x_i},\ p_i)} \leftarrow (\underline{x'_i},\ \underline{p_i \oplus p'_i})$

(x'_1,p'_1)	(x'_2,p'_2)	…	$\boxed{\dots}$	…	…	…	…	…	(x'_N,p'_N)

key, payload masked by PRF

- Client
  - Uses OPRF to compute $\underline{x' \mid p' \leftarrow F_k(w)}$
  - Uses *semi-private KS* to obtain $\boxed{(x_i,\ p_i)}$ where $\underline{\boxed{x_i} = x'}$
  - If entry in database, recovers $p_i = \underline{\boxed{p_i} \oplus p'}$

KS from Semi-Private KS + OPRF

(x_1,p_1)	(x_2,p_2)	…	…	…	…	…	…	…	(x_N,p_N)

S chooses k,
defining $F_k(\cdot)$

$\forall\ (x_i, p_i) \in X,$

Let $\quad x'_i \mid p'_i \leftarrow F_k(x_i)$

Let $\quad \boxed{(x_i, p_i)} \leftarrow (x'_i\,,\, p_i \oplus p'_i)$

(x'_1,p'_1)	(x'_2,p'_2)	…	…	…	…	…	…	…	(x'_N,p'_N)

key, payload masked by PRF

- Security
  - Preserved even if client obtains *all* pseudo-database…
  - Requires that client can't determine output of OPRF other than at inputs from legitimate queries

Weaker OPRF definition suffices for KS

- Strong OPRF: Secure 2PC of PRF functionality
  - No info leaked about **key k** for **arbitrary f_k**, other than what follows from legitimate queries
  - Same OPRF on multiple inputs w/o losing server privacy

- Relaxed OPRF: No info about **outputs** of **random f_k**, other than what follows from legitimate queries
  - Does not preclude learning partial info about k
  - Query set size bounded by t for t-time OPRFs
  - Indistinguishability: Outputs on unqueried inputs cannot be distinguished from outputs of random function

Other results: constructions of OPRF

- OPRF based on non-black-box OT [Y,GMW]

- OPRF based on specific assumptions [NP]
  - E.g., based on DDH or factoring
  - Fully adaptive
  - Quite efficient

- OPRF based on black-box OT
  - Uses relaxed definition of OPRF
  - Good for up to *t* adaptive queries

OPRF based on DDH ["scaled up" NP]

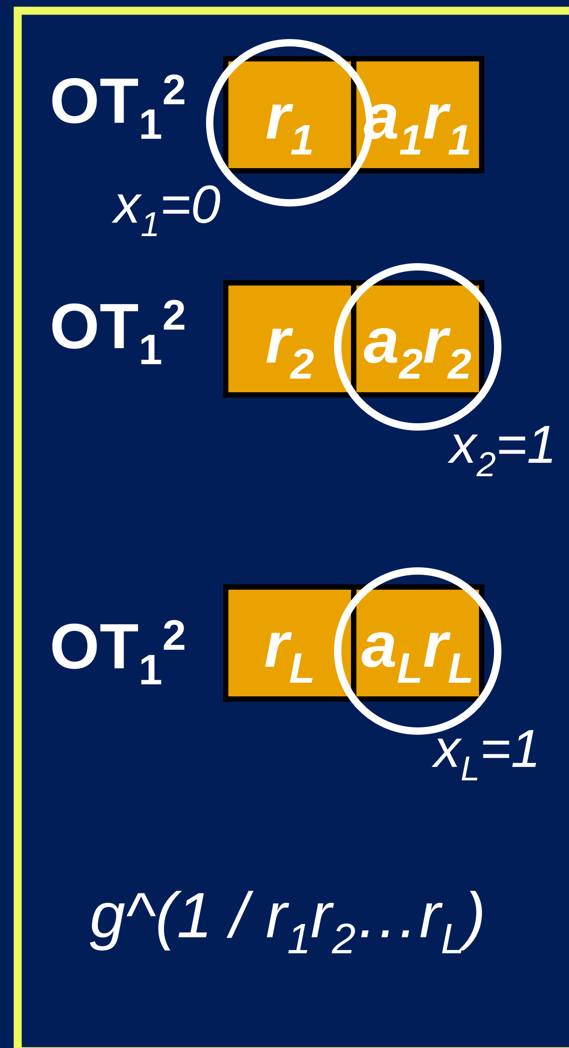
- The Naor-Reingold PRF:
  - Key k = [a_1, …, a_L]
  - Input x = $x_1\, x_2\, x_3 \ldots x_L$

$$F_k(x) = g \wedge (\textstyle\prod_{x_i=1} a_i)$$

  - Pseudorandom based on DDH

- OPRF based on PRF + OT
  - Server: [a_1, …, a_L], [r_1, …, r_L]
  - Client: $x = x_1\, x_2\, x_3 \ldots x_L$
  - L OT's: r_i if $x_i = 0$, $a_i r_i$ otherwise

$$(g^{1/r_1 \ldots r_m})^{(r_1)\ldots(a_m r_m)} = g^{\prod_{x_i=1} a_i} = F_k(x)$$

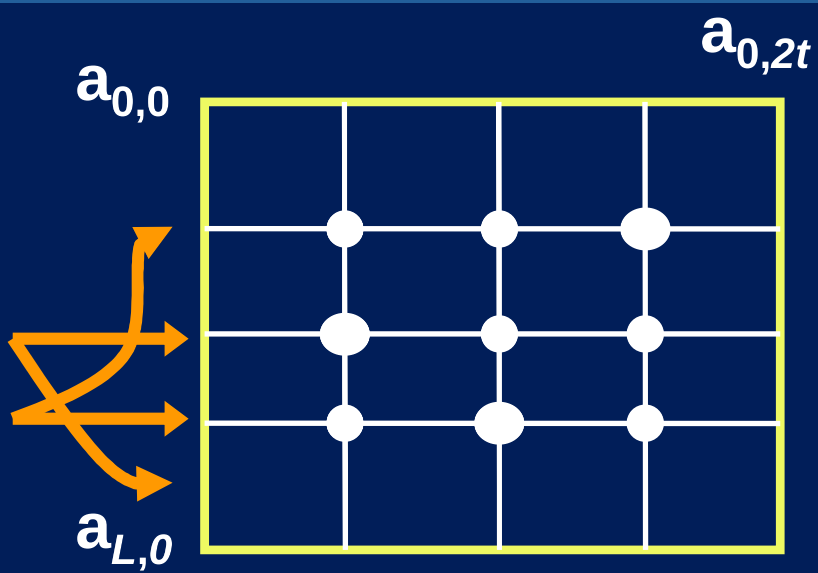


Relaxed OPRF based on OT

- Server key: L x $2t$ matrix
- Client input: $x = \{ x_1, x_2, \ldots , x_L \}$

$$F_k(x) = \otimes g_{i,x_i}$$

- Client gets L keys using OT_1^{2t}
- After t calls, learns t^L keys



- Map inputs to locations in L-dimensions using a (t+2)-wise independent, *secret* mapping h
- Client first obliviously computes $h(x)$, then $F(h(x))$
- Learns t of $2t$ keys in L dimensions
- Probability that other value uses these keys is $(1/2)^L$

Conclusions

- Keyword search is an important tool
- We show:
  - Efficient constructions based on OPE
  - Generic reduction to OPRF + semi-private KS
    - Fully-adaptive based on DDH
    - Black-box reduction via OT, yet only good for t invoc’s
- Open problem:
  - Black-box reduction to OT good for poly invoc’s?

Thanks….