PLAN

(1) ENCRYPTION BASED ON TRAPDOOR PERMUTATION

(2) RABIN OT, 1-2 OT, DEFINITIONS, REDUCTIONS

↖ OBLIVIOUS TRANSFER

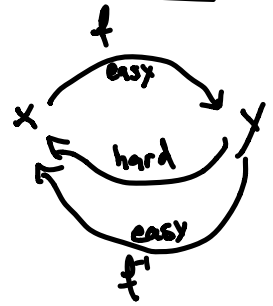
(3) IMPLEMENTATIONS OF 1-2 OT

- (a) TRAPDOOR PERMUTATION
- (b) HOMOMORPHIC ENCRYPTION

(4) COMPUTE "AND" VIA OT

ENCRYPTION VIA TRAPDOOR PERMUTATION

TRAPDOOR DEFINITION:



REMINDER: $(f, f^{-1}) \leftarrow \mathcal{F}$, A FAMILY OF FUNCTIONS, SO THAT NO ADVERSARY KNOWS f^{-1} EXCEPT W/ NEGL. PROBABILITY

KEYGEN $\longrightarrow (f, f^{-1})$

↑ 1WP ↑ TRAPDOOR

sk	pk
f^{-1}	$f: n \to n$, $p_{|p|=n}$

$$E(b, x) = (f(x), \langle p, x \rangle \oplus b)$$

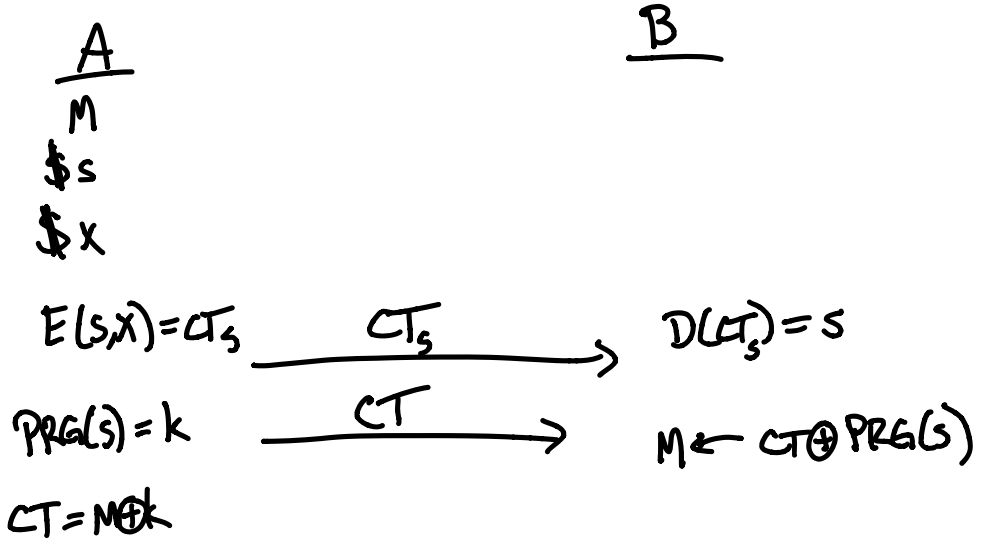
↑ SINGLE BIT ↖ RANDOMNESS

$$D(u, v) = \langle p, f^{-1}(u) \rangle \oplus v$$

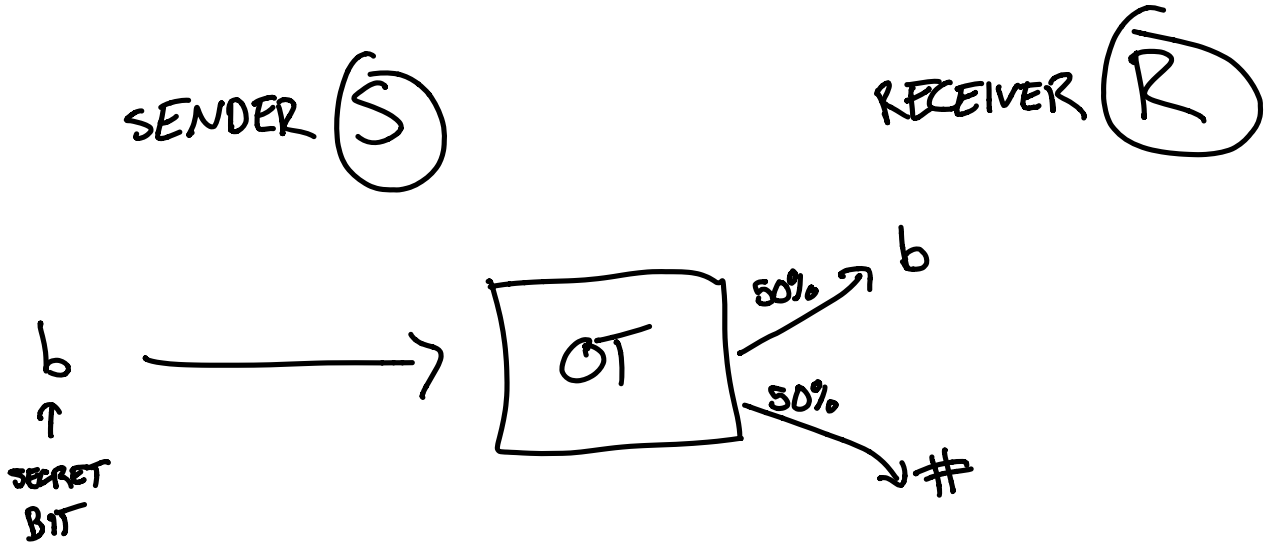
WHY IS THIS SEMANTICALLY SECURE?

ADV WHO CAN DISTINGUISH $E(0)$ FROM $E(1)$ CAN BE USED TO PREDICT THE HARDCORE BIT, BREAKING THE 1WP f. (FULL PROOF: HW EXERCISE)

THIS SCHEME IS VERY INEFFICIENT. IF $f: n \to n$, CIPHERTEXT IS n * (SIZE OF PLAINTEXT). BUT WE CAN USE IT TO ENCRYPT THE SEED OF A PRG AND USE ONE-TIME PAD W/ PRG OUTPUT.



OBLIVIOUS TRANSFER



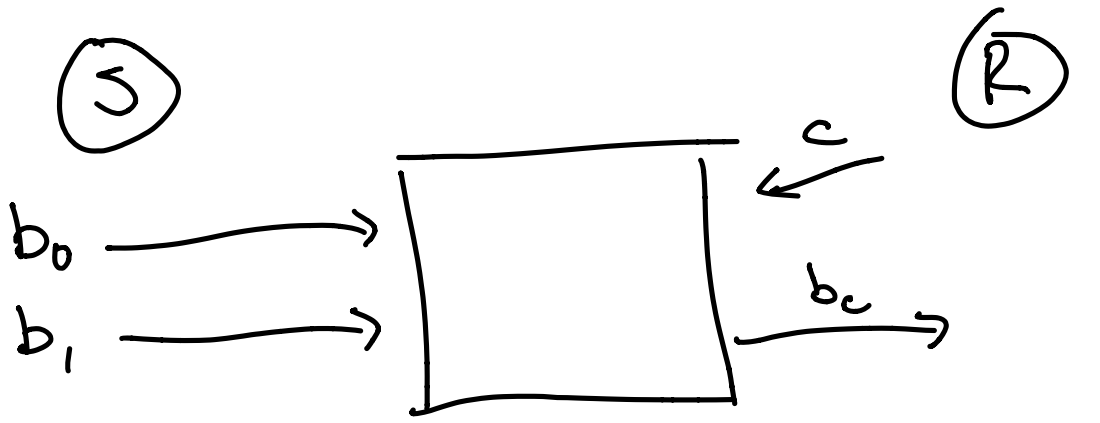
(RABIN OT)

TWO PRIVACY GUARANTEES

(1) SENDER DOESN'T KNOW IF RECEIVER LEARNS b

(2) RECEIVER, IF RECEIVES #, LEARNS NOTHING ABOUT b

ONE-OUT-OF-TWO OT



(1-2 OT)

GUARANTEES:

(1) S DOES NOT LEARN ANYTHING ABOUT c

(2) R DOES NOT LEARN ANYTHING ABOUT b_{1-c}

Interesting Fact:

Rabin OT $\iff$ 1-2 OT

Easy direction: 1-2 OT $\Rightarrow$ Rabin OT

Proof: First, note 1-2 OT gives communication in the clear by making $b_0 = b_1$. Then: implement Rabin-OT by S randomly choosing b_0 or b_1 to always be 1 (or 0). Other b_i is b from Rabin-OT.

Then, R receives b_c and S tells R in the clear which bit was # (always 1).

HARD DIRECTION: RABIN-OT $\Rightarrow$ 1-2 OT

[CLAUDE-GREPEAU]

$\vec{R} = r_1 r_2 \dots r_{3k}$ —RABIN-OT→ $r_1 \# r_3 \# \# r_6 \dots r_{3k}$

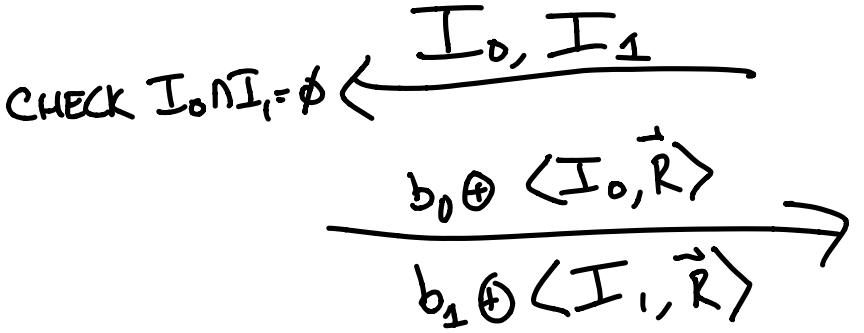
EXPECTED $\frac{3k}{2}$ BITS

$Pr[R \text{ RECEIVED } < k \text{ BITS}]$
$<$ NEGLIGIBLE (CHERNOFF BOUND)

$Pr[R \text{ RECEIVED } > 2k \text{ BITS}]$
$<$ NEGLIGIBLE

CHOOSE I_0, I_1 DISJOINT SETS OF INDICES/LOCATIONS, EACH VECTORS OF SIZE $3k$ W/ EXACTLY k ONES.

FOR ONE SET I_c, KNOWS EVERY r_i FOR $i \in I_c$. OTHER SET I_{1-c} CHOSEN ARBITRARILY.



TO RECOVER b_i, R MUST KNOW EVERY r_j FOR $j \in I_i$. TO DO THIS FOR I_0 AND I_1, MUST HAVE RECEIVED $\geq 2k$ BITS r_i, WHICH DOES NOT HAPPEN WITH OVERWHELMING PROBABILITY. SINCE HE RECEIVES $\geq k$ BITS W/ OVERWHELMING PROBABILITY, CAN LEARN b_c.

S DOES NOT LEARN c BECAUSE OF SYMMETRY: DOES NOT KNOW WHICH SET I_0 OR I_1 IS KNOWN BY R & WHICH IS CHOSEN ARBITRARILY.

Computing "And" via OT

$\text{And}(x_1, x_2)$?

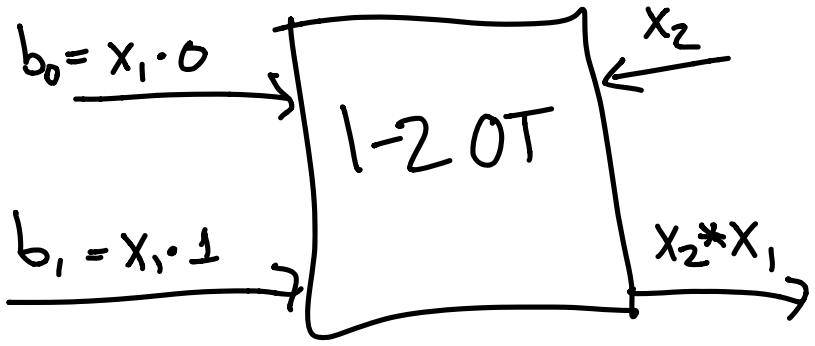
$x_1 * x_2$?

A

$x_1 \in \{0,1\}$

B

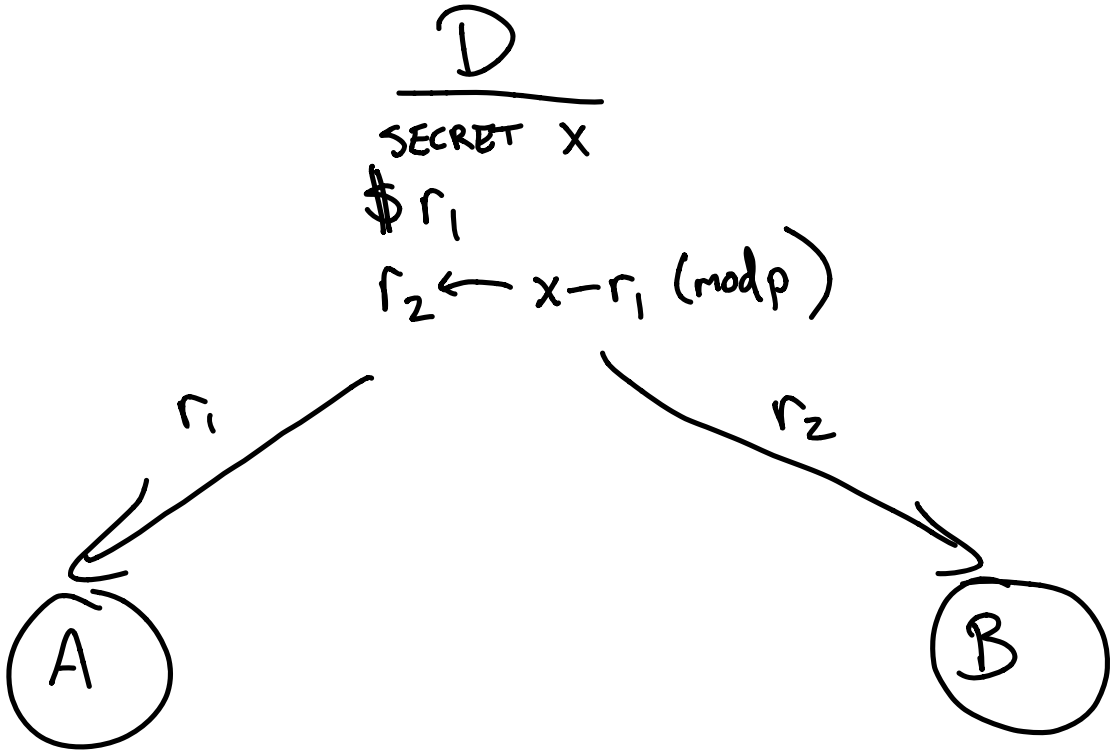
$x_2 \in \{0,1\}$



If private input is 0, do not learn whether other party's input is 0.

2-Party Secure Computation via 1-2 OT

Building Block: Additive Secret Sharing



Generalizes to any # of players.

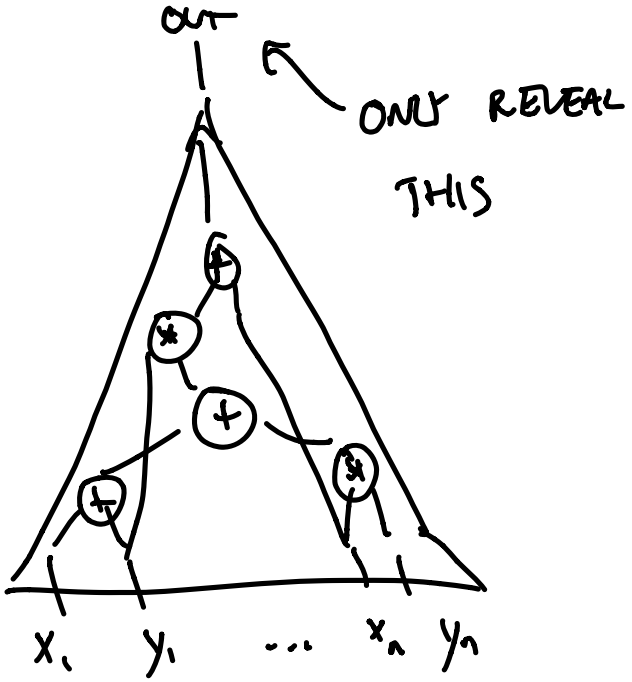
x remains secret unless all players recombine shares.

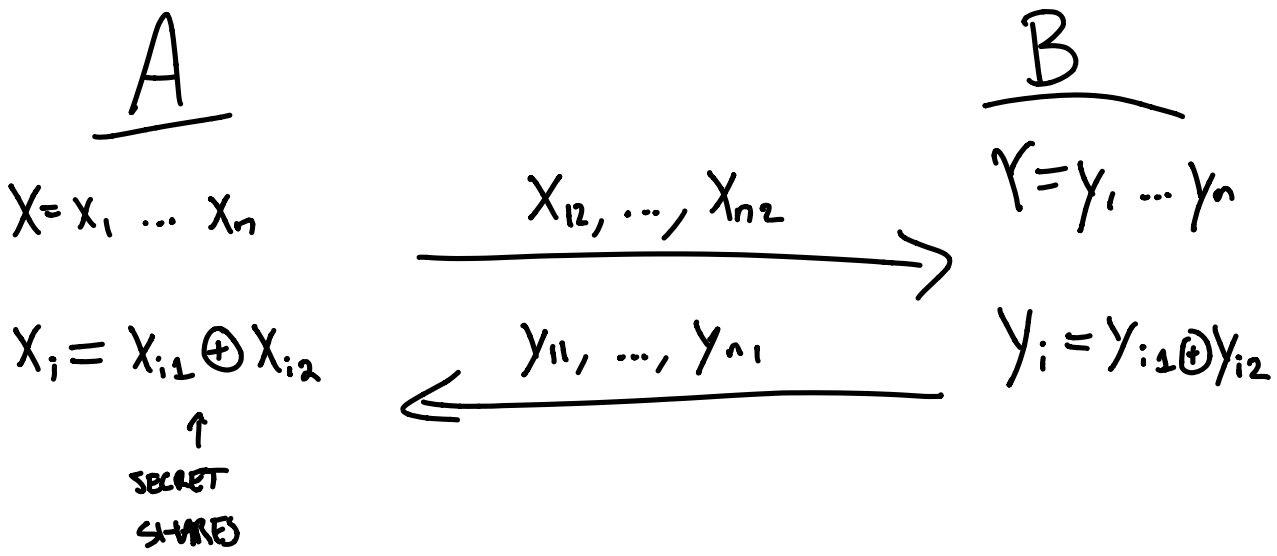
SECURE 2-PARTY COMPUTATION:

SHOULD BOTH LEARN OUTPUT $f(X,Y)$ BUT NOTHING ELSE.

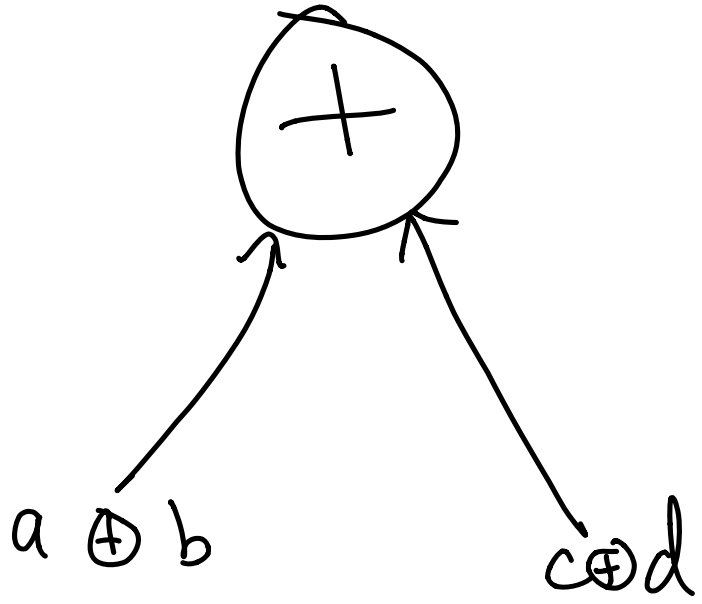
CLAIM: 1-2 OT $\Rightarrow$ 2PC

IDEA: REPRESENT f AS DIGITAL CIRCUIT COMPOSED OF (*) AND (+) GATES (MODULO 2). THEN EVALUATE EACH GATE OVER SECRET SHARED VALUES VIA 1-2 OT.





NOW, NEED TO SHOW HOW TO COMPUTE $\oplus$ GATE & $*$ GATE OVER SECRET-SHARED INPUTS TO GET A SECRET-SHARED OUTPUT.



A

a, c

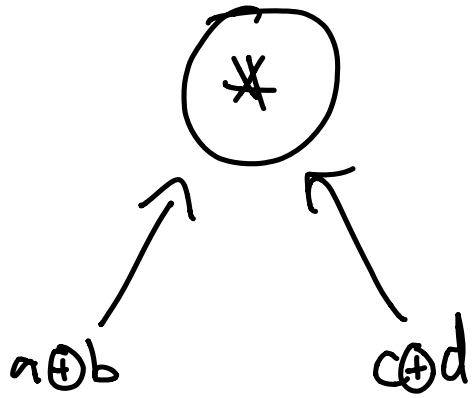
B

b, d

SOLUTION: $\text{OUTPUT} = a \oplus b \oplus c \oplus d$

$= (a \oplus c) + (b \oplus d)$

NO COMMUNICATION NECESSARY! A LOCALLY COMPUTES $a \oplus c$, B LOCALLY COMPUTES $b \oplus d$.

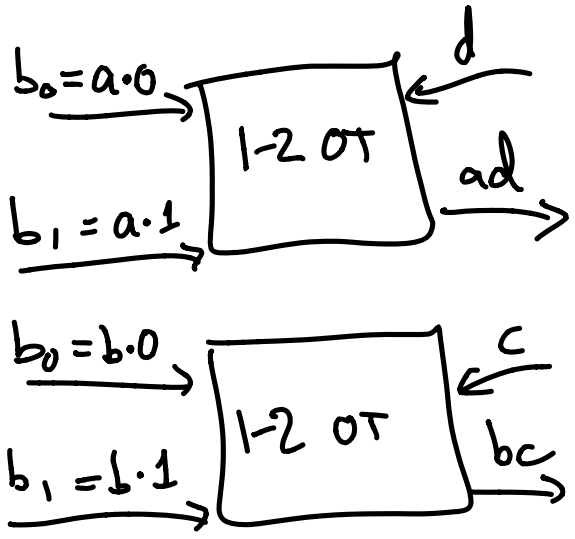


A

a, c

B

b, d



ac

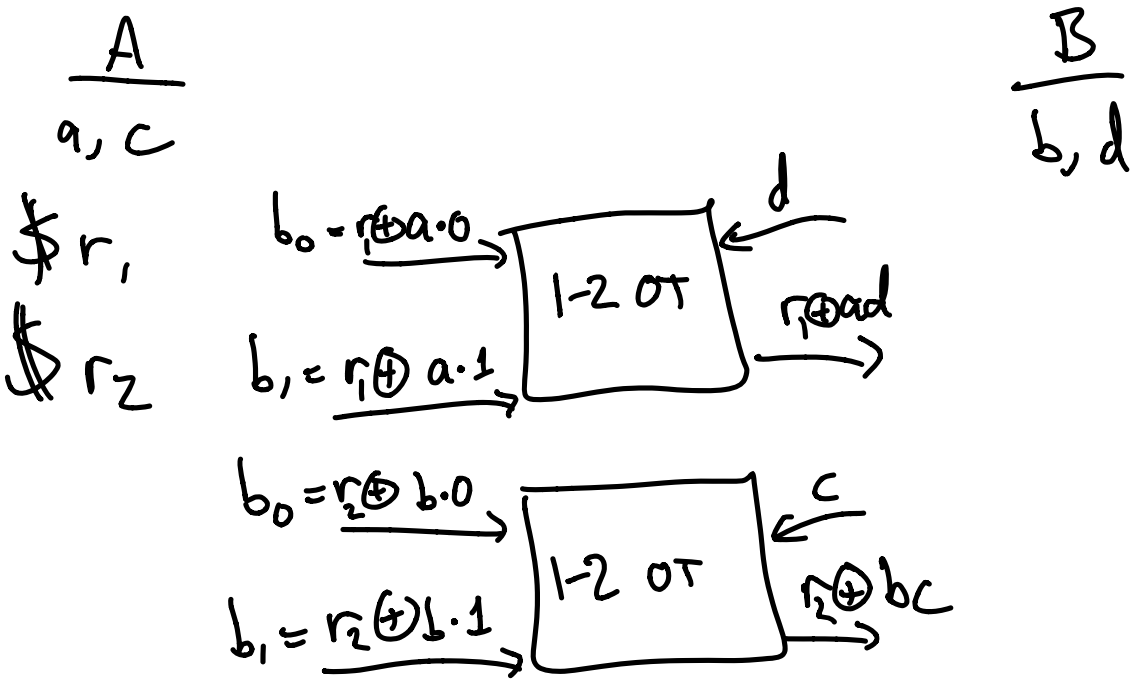
ad⊕bc⊕bd

$$ac \oplus ad \oplus bc \oplus bd = (a \oplus b) * (c \oplus d)$$

THIS ALMOST WORKS. CROSS TERMS

ad, bc MIGHT REVEAL INFORMATION!

ALICE CHOOSES RANDOM BLINDING FACTORS r_1, r_2



$r_1 \oplus r_2 \oplus ac$

$ad \oplus r_1 \oplus bc \oplus r_2 \oplus bd$

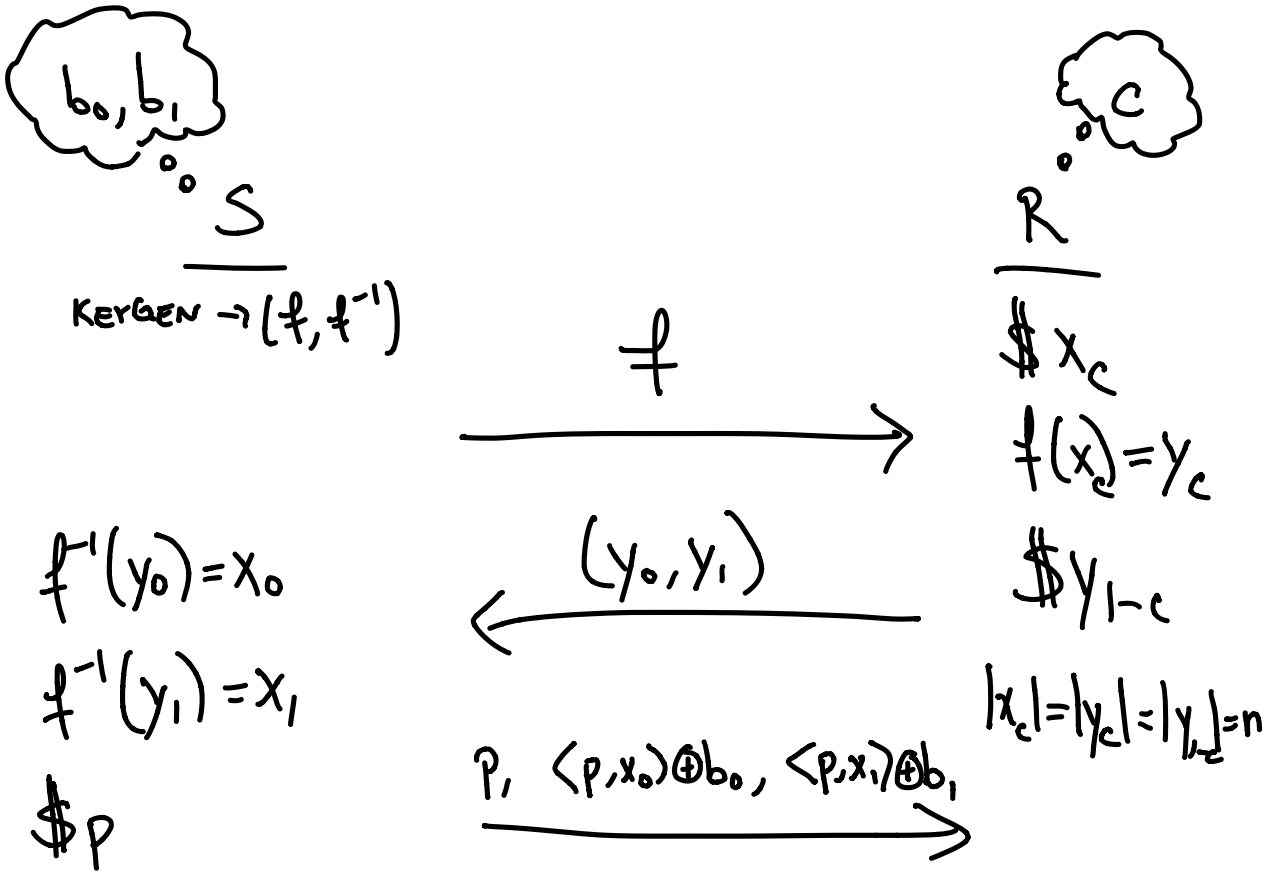
$r_1 \oplus r_2 \oplus ac \oplus ad \oplus r_1 \oplus bc \oplus r_2 \oplus bd = (a \oplus b) * (c \oplus d)$

THIS ALLOWS A, B TO SECURELY COMPUTE ⊕, ⊛ GATES W/ SECRET SHARED INPUTS/ OUTPUTS. SO THEY SECRET-SHARE $X + Y$, EVALUATE $f(X,Y)$ GATE BY GATE, + RECONSTRUCT OUTPUT.

① Implementations of 1-2-OT

ⓐ Trapdoor Permutations ⇒ 1-2-OT

[GMW 87]



As long as R behaves honestly, can recover b_c but cannot recover b_{1-c} (by hardcore bit theorem).

But malicious R can know both inverses x_0, x_1 by choosing them first & computing $y_0 = f(x_0)$ & $y_1 = f(x_1)$.

How do we force R to behave honestly? Sender tosses coins into receiver's well, & receiver proves in ZK that he behaves according to protocol & uses sender's random coins.

ADDITIONAL PROBLEM:

WHAT IF S CHOOSES f WHICH IS NOT A PERMUTATION?

SOLUTION #1: USE "CERTIFIABLE" TRAPDOOR PERMUTATION.

R CAN LOOK AT CODE OF f & KNOW IT'S A PERMUTATION.

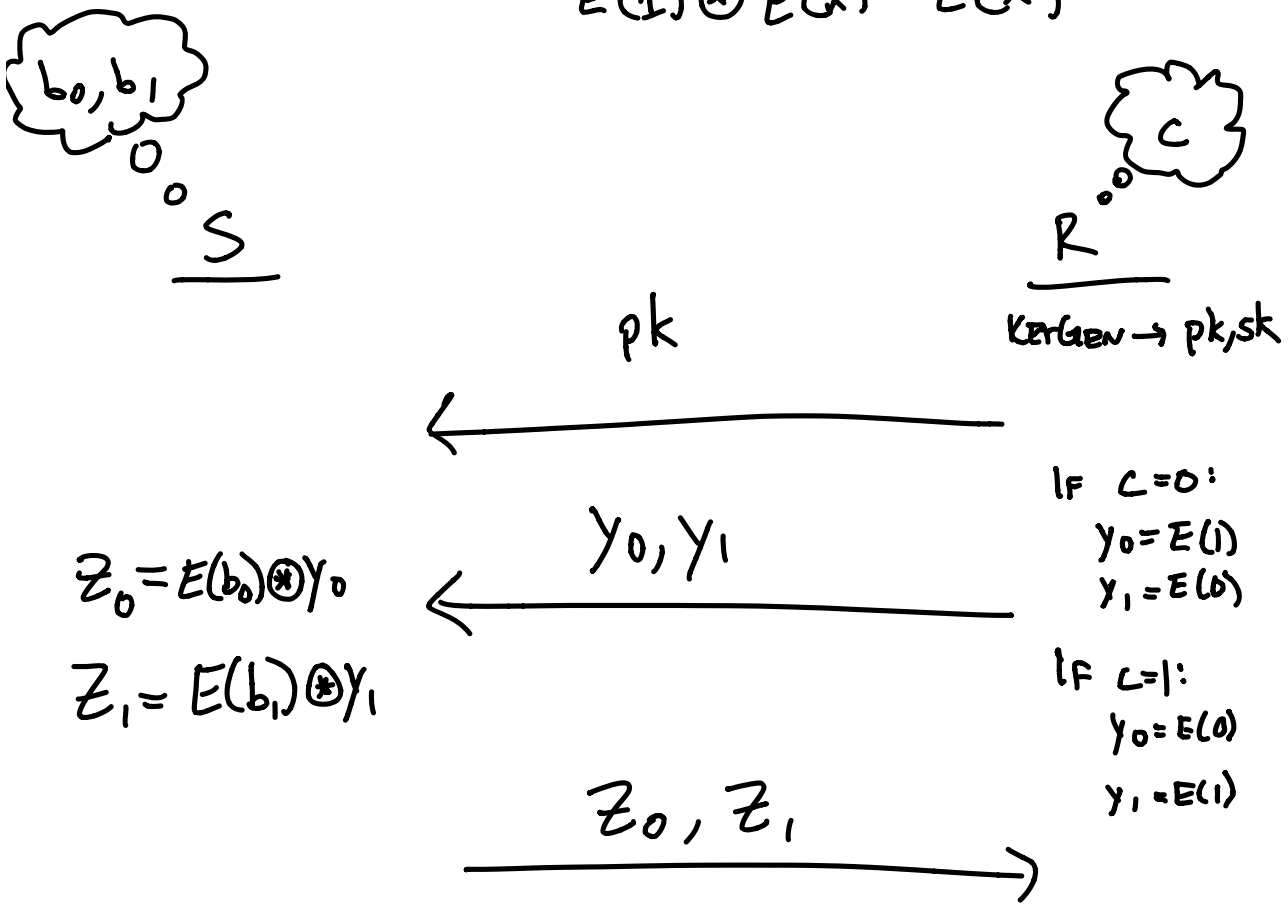
SOLUTION #2: R USES CHALLENGE-RESPONSE GAME TO CONFIRM THAT f IS ONE-TO-ONE ON "MOST" INPUTS.

(b) HOMOMORPHIC ENCRYPTION $\Rightarrow$ 1-2-OT

ASSUMPTION $E(x) \circledast E(y) = E(x * y)$

IN PARTICULAR: $E(0) \circledast E(?) = E(0)$

$E(1) \circledast E(x) = E(x)$



$$Z_c = E(b_c) \circledast E(1) = E(b_c)$$

$$Z_{1-c} = E(b_{1-c}) \circledast E(0) = E(0)$$

THUS: R DECRYPTS Z_0, Z_1 & LEARNS b_c

BUT LEARNS NOTHING ABOUT b_{1-c}

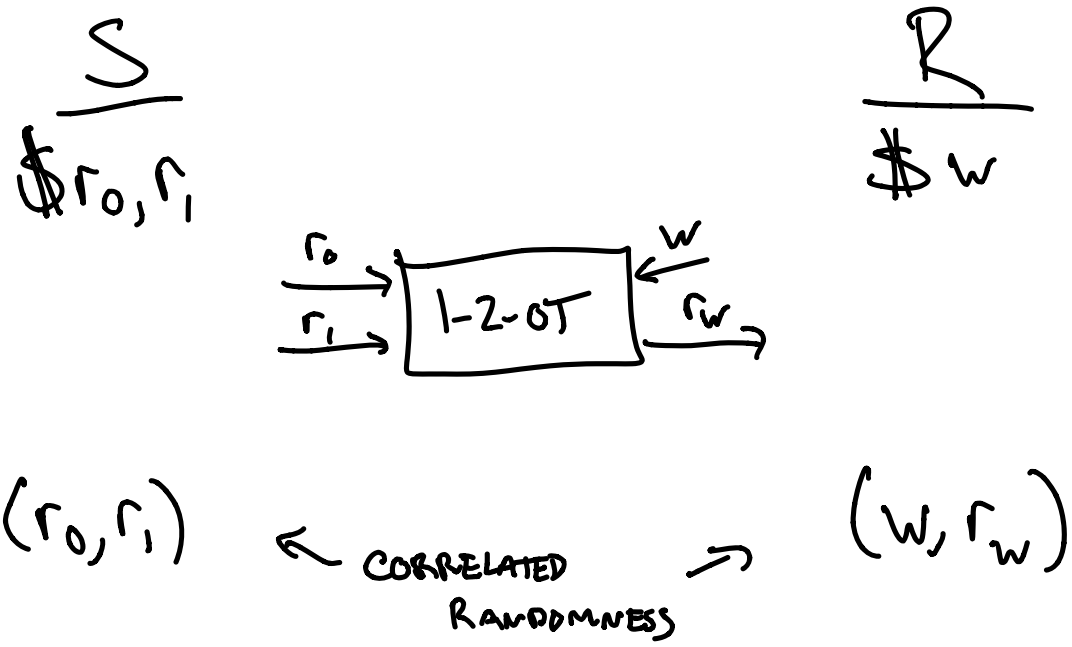
COMPARISON BETWEEN IMPLEMENTATIONS:

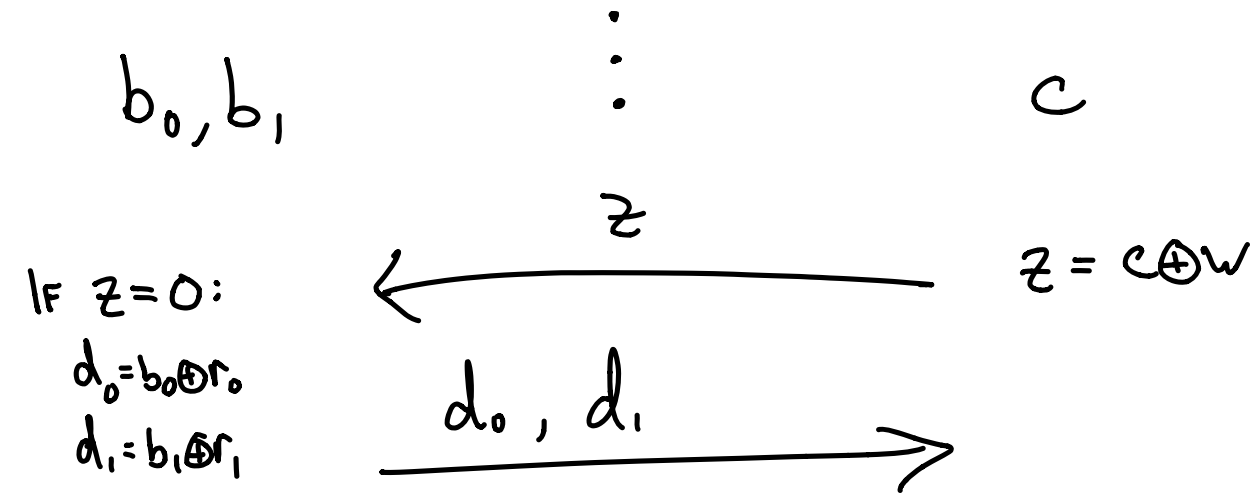
TRAPDOOR PERMUTATIONS	HE
b_0, b_1: HIDDEN COMPUTATIONALLY	b_0, b_1: HIDDEN INFORMATION THEORETICALLY
c: HIDDEN INFORMATION THEORETICALLY	c: HIDDEN COMPUTATIONALLY

IMPOSSIBLE TO HIDE BOTH INFORMATION-THEORETICALLY (HW)

(c) Beaver's Trick: Random OT

Idea: Can we "pre-compute" 1-2-OT, before knowing inputs b_0, b_1, c, and use it later?





C IS HIDDEN BECAUSE IT'S MASKED BY W, WHICH IS UNKNOWN TO S.

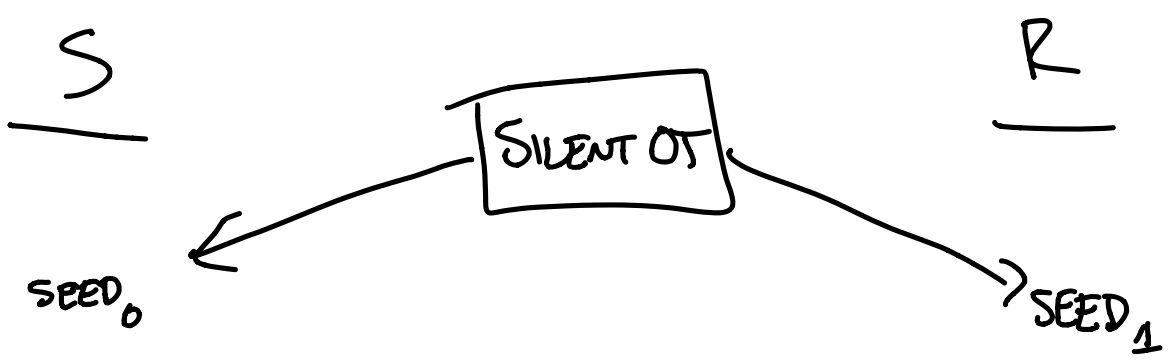
IF $C=W$: $z=0$, AND b_c CAN BE RECOVERED BY COMPUTING $d_w \oplus r_w$. b_{1-c} IS HIDDEN BECAUSE R DOESN'T KNOW r_{1-w}.

IF $C \neq W$: $Z = 1$, & b_c CAN BE RECOVERED VIA $d_w \oplus r_w$. b_{1-c} HIDDEN FOR SAME REASON.

(d) OT AT "BULK" + SILENT OT

OT EXTENSION.

COVER NEXT TIME.



$PRG(SEED_0) =$

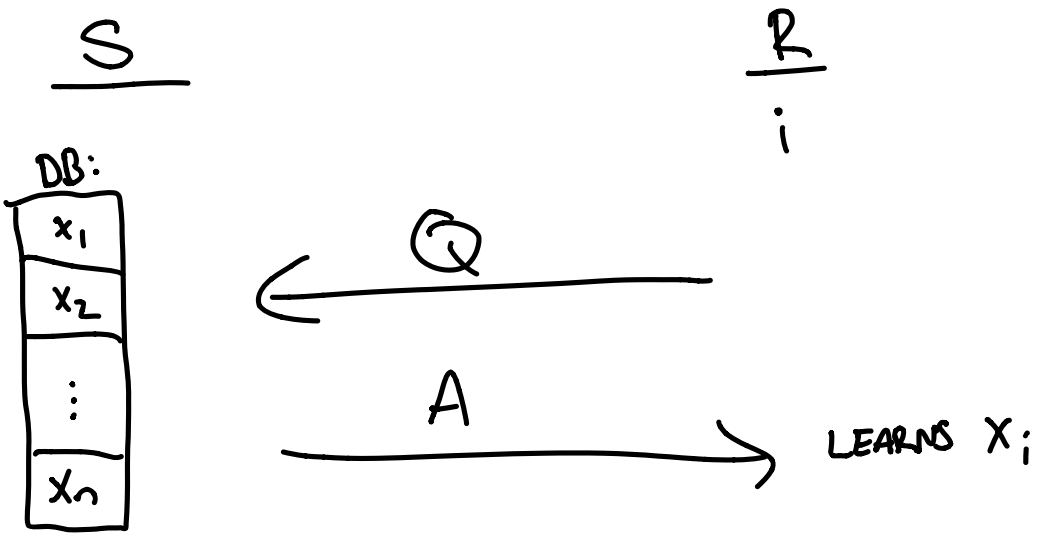
$(r_0, r_1)_1, \ldots, (r_0, r_1)_k$

$PRG(SEED_1) =$

$(w, r_w)_1, \ldots, (w, r_w)$

(2) Single Server Private Information Retrieval (PIR)

Consider "1-N-Out" (1-out-of-N):

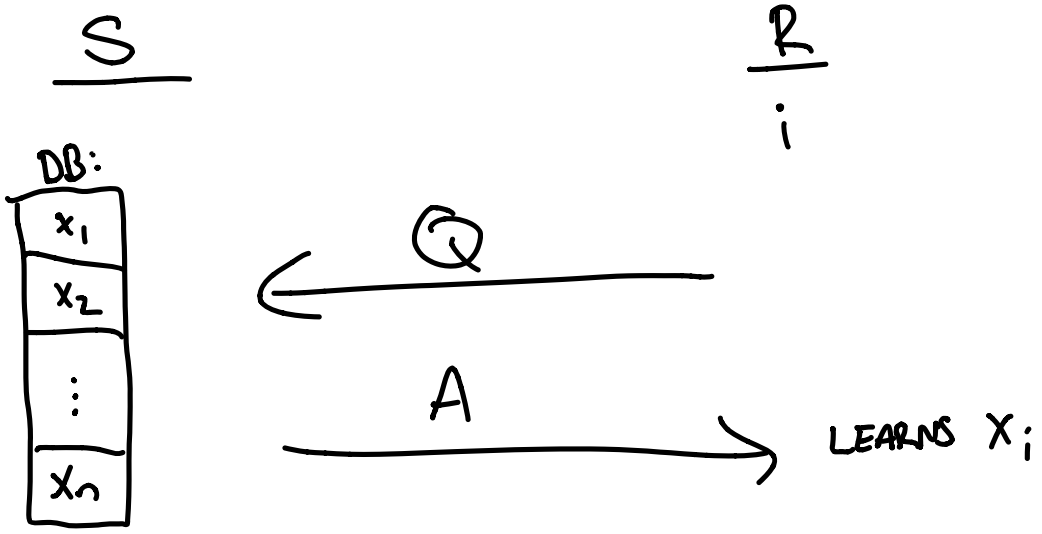


1-N-OT Security Guarantees:

- i is hidden
- x_j hidden for all $j \neq i$

THIS IS ALMOST PIR!

PIR:



PIR SECURITY GUARANTEES:

- i IS HIDDEN
- BOTH $|Q| + |A| << n$

PIR STRONGER THAN 1-N-OT!

PIR $\Rightarrow$ 1-N-OT

(b) CONSTRUCTING PIR VIA HOMOMORPHIC ENCRYPTION:

$E(x) * E(y) = E(x \oplus y)$

---

(1) S BREAKS DB INTO $\sqrt{n}$ BLOCKS, EACH OF SIZE $\sqrt{n}$.

(2) R SENDS HE pk TO S. FOR THE BLOCK CONTAINING x_i, R SENDS $y = (E(0), E(1))$. FOR ALL OTHER BLOCKS, SENDS $y = (E(0), E(0))$.

(3) S REPLACES EACH 0 IN DB WITH y_0 FOR THAT BLOCK, & REPLACES 1 WITH y_1.

(4) S MULTIPLIES THE FIRST ENTRIES IN EACH BLOCK TO GET A_1, MULTIPLIES THE SECOND IN EACH BLOCK TO GET A_2, ..., TO GET $A_{\sqrt{n}}$

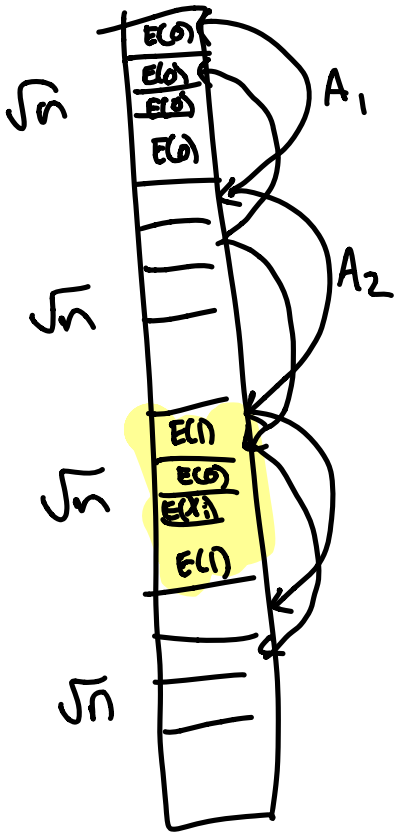
(5) S SENDS $A_1, \ldots, A_{\sqrt{n}}$ TO R, WHICH IS PRECISELY AN ENCRYPTION OF THE BLOCK CONTAINING X_i

(6) R DECRYPTS $A_1, \ldots, A_{\sqrt{n}}$ & LEARNS X_i (ALONG WITH ALL OTHER ENTRIES IN THAT BLOCK)

WHY PIR?

- i HIDDEN BECAUSE $E(0)$ LOOKS LIKE $E(1)$ TO S.
- $|Q| + |A| = 2k\sqrt{N} + k\sqrt{n} = 3k\sqrt{n} \ll n$,
  k = SIZE OF ENCRYPTION

VISUAL:

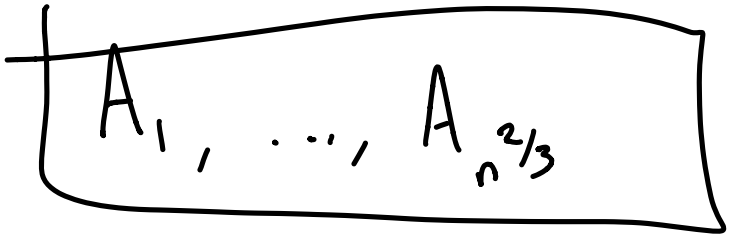


$A_1 = E(1)$, $A_2 = E(0)$,
$A_3 = E(x_1)$, $A_4 = E(1)$

CAN WE DO BETTER THAN $\sqrt{n}$ COMMUNICATION?

IDEA: $n^{1/3}$ TOTAL BLOCKS, EACH OF SIZE $n^{2/3}$

$$|Q| = k \cdot n^{1/3}$$



↑ EACH k-BIT CIPHERTEXT

USES THIS AS NEW DATABASE!

RECEIVER WANTS TO RETREIVE $E(x_i)$ FROM

THIS DATABASE, BUT THIS IS A k-BIT CIPHERTEXT. SO MUST SEND k QUESTIONS, EACH OF SIZE $k \cdot n^{1/3}$. SO TOTAL

$$|Q| = 2k^2 \cdot n^{1/3}, \quad |A| = k^2 \cdot n^{1/3},$$

$$|Q| + |A| = 3k^2 \cdot n^{1/3}.$$

COULD CONTINUE THIS RECURSIVELY.

© SINGLE-SERVER 1-ROUND PIR

$\Rightarrow$ COLLISION-RESISTANT HASH FUNCTIONS

PROOF: PICK i AT RANDOM.

$Q(i)$ IN PIR IS A

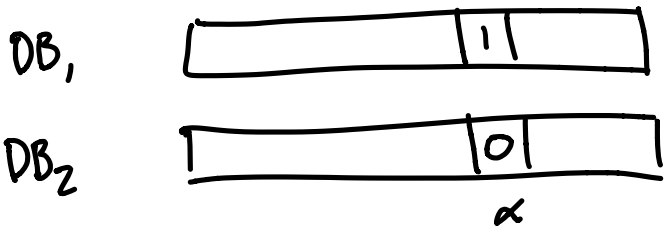
COLLISION-RESISTANT HASH FUNCTION

① IS $Q(i)$ LENGTH DECREASING?

$Q(DB) \longrightarrow \{A_i\}, |\{A_i\}| << |DB|$. ✓

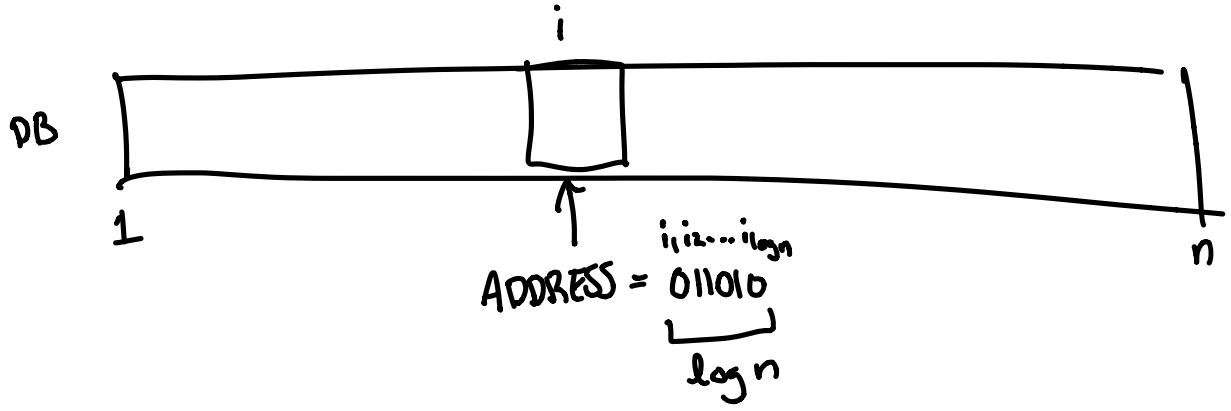
② CLAIM: NO PPT A_{in} CAN FIND DB_1, DB_2 S.T. $Q(DB_1) = Q(DB_2) = \{A_i\}$, $DB_1 \neq DB_2$.

ASSUME OTHERWISE. $DB_1 \neq DB_2$, SO

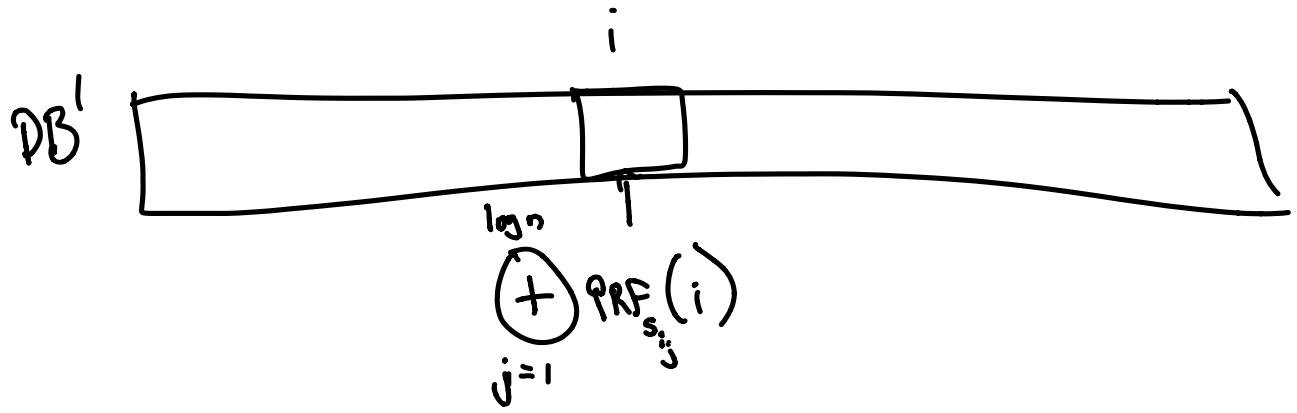


BUT PIR $Q(i)$ MUST RETURN CORRECT ANSWER ON i^{TH} BIT! SO COLLISION CANNOT OCCUR WHEN $\alpha = i$. THUS A COLLISION LEAKS INFORMATION THAT $\alpha \neq i$, WHICH BREAKS SECURITY OF PIR.

② PIR $\Rightarrow$ 1-N-OT



$(S_0, S_1)_j$ FOR EACH BIT OF ADDRESS (PRF KEYS)



"MASKED DB"

Now, run PIR on masked DB'.

Then, use 1-2-OT to get correct PRF key for each bit of address i to "unmask" only a single location.