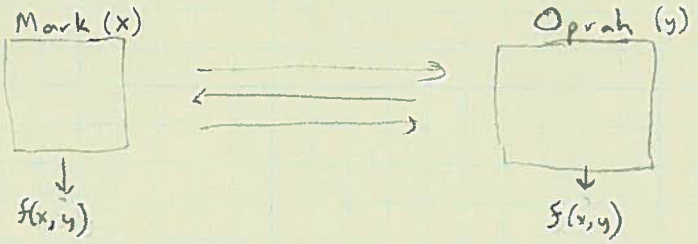
Garbled Circuits

Andrew Yao
- Turing award winner
- PhD in physics and CS (second in 3 years)
- Taught at MIT, Stanford, Berkeley, Princeton
- Now runs "Yao Class" at Tsinghua
- Many influential early papers:
  - ↳ Results on OWFs, PRGs, now standard
  - ↳ Too many other cool results to list

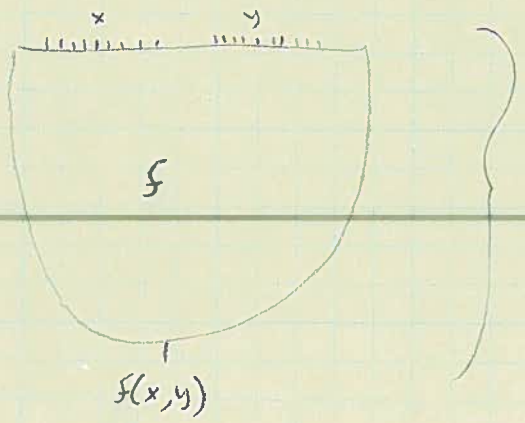
Today, we discuss one: "Yao's Millionaire's Problem"
↳ Introduced in talk in 1986
- Two millionaires sit on yacht in French Polynesia



- Intuitively, both learn $f(x,y)$, "but nothing else"
- Using this idea, can jointly compute $f(x,y)$ for any f s.t. nothing leaks except $f(x,y)$.
  - → Compute RSA modulus s.t. neither knows factors
  - → x = DB, y = indices of rows $f(x,y)$ = sum of selected rows

Circuits

We need to represent f as a boolean circuit (wlog!)
- → AND, OR, NOT gates
- → Useful b/c evaluating a circuit is "data oblivious"



Yao's protocol does not hide f.

⇒ Wlog let f be deterministic (why is this wlog?)

Yao's "Garbled Circuits"

Let us define what it means for 2PC to be secure.

→ We are going to stick w/ "honest but curious" adv for now
↳ Later will discuss full security.

Two parties $\{1, 2\}$ w/ inputs $\{x, y\}$ s.t. $|x| = |y|$

Defn Protocol Π securely computes $f(x, y)$ if $\exists$ ppt simulators S_1, S_2 s.t. $\forall\ x, y \in \{0,1\}^n$

$$\{S_1(x, f(x,y))\} \overset{c}{\approx} \{\text{view}_1^{\Pi}(x,y)\}$$

(Same for party 2.)

Informally: Each party can simulate its view of the protocol execution given only

1) It's private input
2) The value $f(x,y)$.

} "What leaks"

$\Rightarrow$ Π is secure if it leaks $f(x,y)$ and nothing else!

This is similar to ZK defns, except that we allow a little extra leakage

Defn Π is correct if for all $x, y \in \{0,1\}^n$

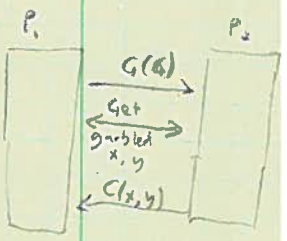
$$\{\text{output}^{\Pi}(x,y)\} \overset{c}{\approx} \{f(x,y)\}$$

[Note: Can also have players 1 and 2 receive distinct outputs that the other player doesn't learn, ... also not too hard.]

Garbled Circuits

High-level idea:

* P_1 generates a "garbled" version of ckt $C(x,y)$
  ↳ hides all internal wire values
* P_2 uses O.T. to fetch enough info to evaluate $G(x,y)$ without leaking y to P_1 ("Garbled input wires")
* P_2 evaluates $G(x,y)$, sends result to P_1



First, lets see what P_1 does to garble C.

We need a semantically secure enc scheme s.t.

1) hard to generate "valid" cts w/o knowing key
2) given key, easy to check whether ct is valid.

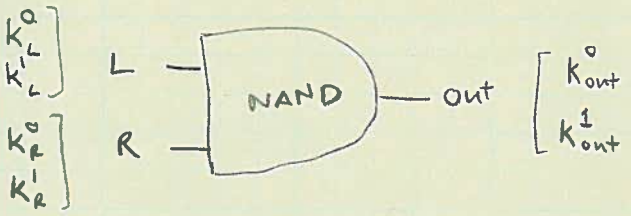
Using PRF $F: \underbrace{\{0,1\}^n}_{\text{key}} \times \underbrace{\{0,1\}^n}_{\text{input}} \to \underbrace{\{0,1\}^{2n}}_{\text{output}}$

$$E(k,m) := \begin{cases} r \| \xleftarrow{R} \{0,1\}^n \\ (r, \text{"}m \| 0^n\text{"} \oplus f_k(r)) \end{cases}$$

$$D(k, ct=(r,c)) := \begin{cases} \text{if last } n \text{ bits of } f_k(r) \oplus c \; != 0 : \text{fail} \\ \text{else output first } n \text{ bits of } f_k(r) \oplus c \end{cases}$$

Garbled Ckts

For each gate in the circuit we use 6 keys



We want the property that: given K_L^α, K_R^β, can compute only $K_{out}^{(\alpha \text{ NAND } \beta)}$

α	β	α NAND β
0	0	1
0	1	1
1	0	1
1	1	0

encrypt

$$C_{00} = E\left(K_L^0, E\left(K_R^0, K_{out}^1\right)\right)$$

$$C_{01} = E\left(K_L^0, E\left(K_R^1, K_{out}^1\right)\right)$$

$$C_{10} = E\left(K_L^1, E\left(K_R^0, K_{out}^1\right)\right)$$

$$C_{11} = E\left(K_L^1, E\left(K_R^1, K_{out}^0\right)\right)$$

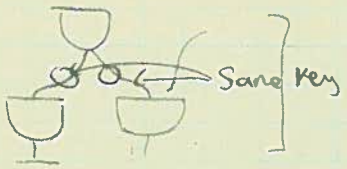
shuffle

C_{10}
C_{11}
C_{00}
C_{01}

"Garbled gate"

Garbled Ckts

* To garble C, choose pair of keys for each wire in ckt ("0" key and "1" key)... some subtlety...



* Output gate has encryptions of 0/1 → not keys in encrypted truth table

Claim: Let $x \in \{0,1\}^n$, $y \in \{0,1\}^n$ be inputs
Given a set of keys

$$\left(k^{x_1}_{1,1}, \ldots, k^{x_n}_{1,n}\right), \left(k^{y_1}_{2n+1,1}, \ldots, k^{y_n}_{2n,n}\right)$$

corresponding to input wires of ckt, can compute $C(x,y)$.

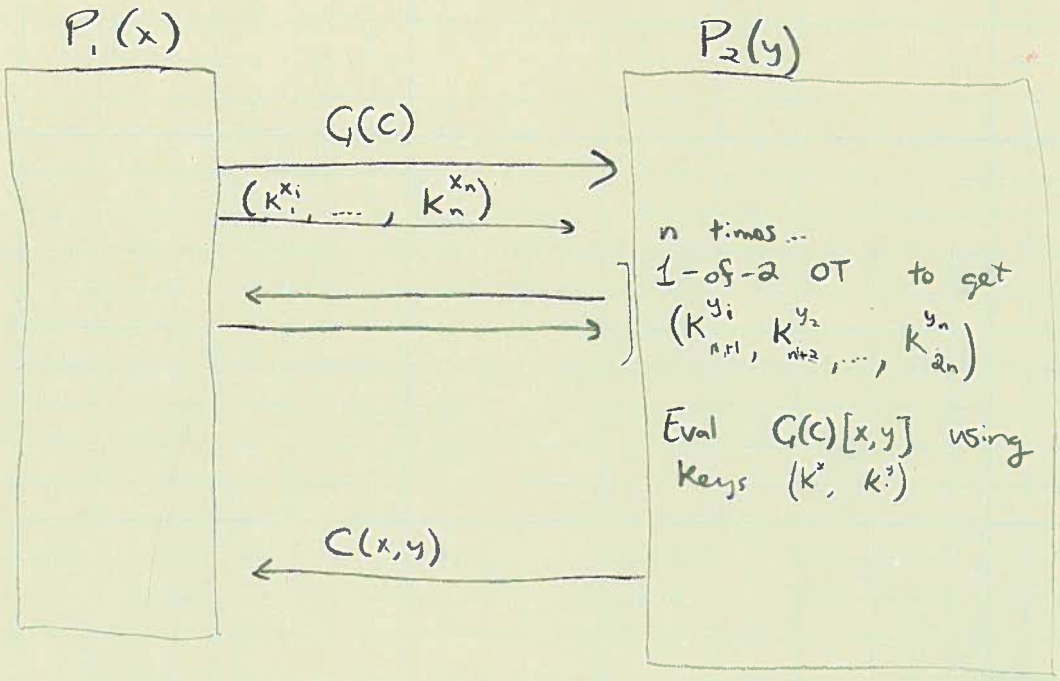
Idea: By induction on gate index in ckt:

Base case: Given key to output gate, can recover output value.

Induction: Given keys $(k^{\alpha}_{L}, k^{\beta}_{R})$ to gate i, can decrypt ct in truth table corresponding to gate $i-1$.

Now have keys for all subsequent gates
⇒ Can compute $C(x)$ by induction hypothesis.

Garbled Ckts for 2PC



Correctness Idea: If P_1, P_2 honest, both parties learn $C(x,y)$.

Security Pf Idea:

P_1 is dishonest: need Sim s.t. $Sim_1(x, f(x,y)) \overset{c}{\approx} \{view_1^{\pi}(x,y)\}$

- Simulate OTs using OT simulator
- Can output $f(x,y)$ as given to Sim

P_2 is dishonest:

- Can generate a bogus G.C. that looks like the real one
- Bogus GC outputs $f(x,y)$ on all inputs
  ↳ Generate bogus GC, then use hybrid arg to show its comp indist to real
- Simulation of OT step comes from ot Protocol.

Malicious Security

Q: Why are we not done?

A: Many problems. * E.g. P_1 sends garbling of ckt $\hat{C} \neq C$. Output of $\hat{C}$ might leak all of P_2's secret input!

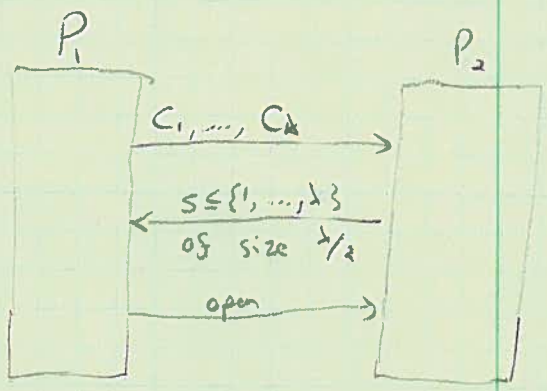
* OT protocol might not be fully secure either!

Proving security against fully malicious advs is very subtle.... "Cryptographers rarely sleep well"

We show one piece of a full maliciously secure GC.

"Cut-and-Choose"

- Very simple (but surprising!) technique
- P_1 creates λ GCs
- P_2 asks P_1 to "open" $\lambda/2$ of them
  - ↳ P_1 sends all labels for subset S.
- P_2 checks all opened ckts are good
- P_1 is convinced that a majority of the remaining ckts are good



Soundness

- If a cheating P_1 wins, $\geq \frac{1}{4}$ ckts must be bad
- Then in step 1, P_2 opens a bad ckt w.p.

$$\Pr\left[\begin{matrix}\text{all opened}\\ \text{good}\end{matrix}\right] = \Pr[C_1 \text{ good} \cap C_2 \text{ good} \cap \ldots \cap C_{\lambda/2} \text{ good}]$$

$$= \Pr[C_1 \text{ good}]^{\lambda/2}$$

$$= \left(\frac{3}{4}\right)^{\lambda/2} \leftarrow \text{negligible in } \lambda$$

$\Rightarrow$ Cut-and-choose is insufficient.... need to worry about labels too!

Optimizations

Many tricks to reduce size of GCs.

Examples: "Free XOR" [KS08]

↳ No need to store garbled XOR gates

"Random Permutation instead of PRF [Bellare et al 2013]

↳ Faster (?) on AES hw

[?]

"Point and Permute" [Malkhi et al 2004]

↳ add extra info to labels to avoid 4 decryption attempts per gate

⋮

Many other nice tricks

⇨ JustGarble reports ~25 cycles/gate to evaluate
~250 to garble
64 bytes/gate ckt size

→ Practical for smallish ckts in HBC setting

↳ setting closer to practical in malicious setting too (see David's paper)