Miscellany

Lecture 27

The Importance of Being Shallow

Circuit Depth

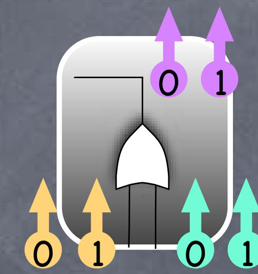
- Functions f: $\{0,1\}^* \rightarrow \{0,1\}^*$ are often represented as circuit families (boolean or arithmetic)
  - Family of circuits $\mathbf{C} = \{ C^n \}_{n\geq 1}$
    - Each circuit is a DAG, with n input wires. Will restrict ourselves to circuits with 2-input gates
    - For each input size n there is a separate circuit C^n (w.l.o.g., same output size for each fixed input size)
- Depth of a DAG: length of the longest root-to-leaf path
- $\mathbf{C}$ said to have "constant depth" if depth(C^n) ≤ c, for all n
- $\mathbf{C}$ in class NC^i if depth(C^n) ≤ $c \cdot \log^i n$, for some c
  - Note: In NC^0 circuits each output wire connected to a constant number of input wires

Depth and Interaction

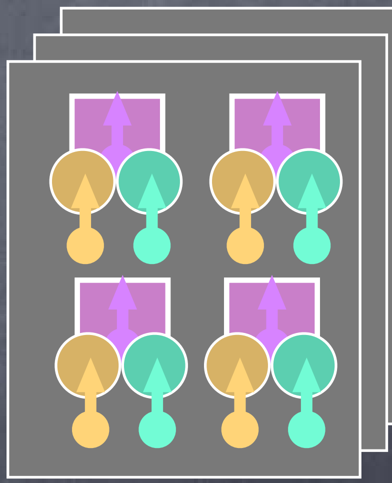
- Recall the GMW and BGW protocols
- Gate-by-gate evaluation of a circuit (DAG)
- Gates can be evaluated in any order as long as we respect a topological sort
- Can parallelise by grouping gates into <u>levels</u>
  - Number of rounds of interaction = number of levels
  - Smallest number of levels = depth of the circuit
- Moral: Functions with shallow circuits are quicker to evaluate
- Can sometimes do better by working with low-depth "randomized encoding" of functions than directly with their own circuits
  - e.g., 2-party semi-honest setting

Recall

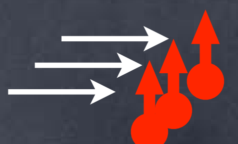
Garbled Circuits



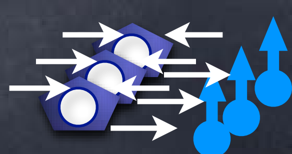
- Recall: Each wire w has two keys ($K_{w=0}$ and $K_{w=1}$). Each garbled gate has 4 boxes with keys for the output wire, locked with keys for input wires
  - Locking: $Enc_{K_{x=a}}(Enc_{K_{y=b}}(K_{w=g(a,b)}))$



- Information-theoretic garbling: why not just use information-theoretic encryption?
  - One-time pad: $Enc_K(m) = m \oplus K$
  - But $K_{x=a}$ used to encrypt two values in a gate, $Enc_{K_{y=0}}(K_{w=g(a,0)})$ and $Enc_{K_{y=1}}(K_{w=g(a,1)})$



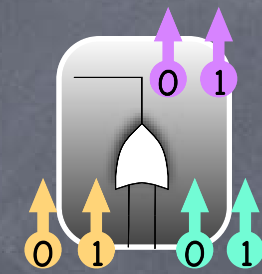
  - If the wire x fans out to t gates, encrypts 2t values
  - Can we still use a one-time pad?

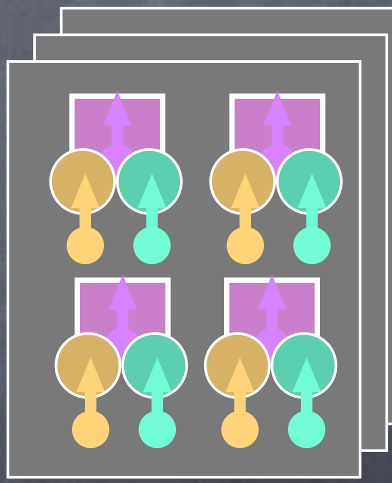


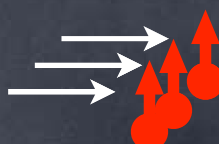
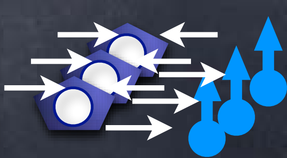
Information-Theoretic Garbled Circuits

- Recall: Each wire w has two keys ($K_{w=0}$ and $K_{w=1}$). Each garbled gate has 4 boxes with keys for the output wire, locked with keys for input wires
  - Locking: $Enc_{K_{x=a}}(Enc_{K_{y=b}}(K_{w=g(a,b)}))$
- Encrypting 2t messages ≡ encrypting a long message
  - Suppose fan-out bounded by t. Then for wires w_i at depth i, enough to have $|K_{w_i=a}| = 2t\ |K_{w_{i-1}=c}|$
  - Key-size at depth d = $O((2t)^d)$ (with 1-bit key at the output)
- Polynomial sized if d is logarithmic and t constant
- Information-theoretic garbled circuits possible for shallow circuits (NC^1)

Alternate constructions avoid bound on t





Recall

Gentry-Sahai-Waters

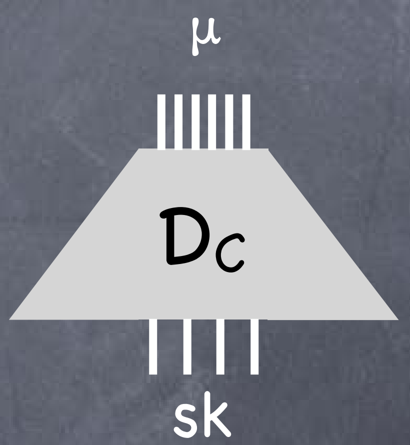
- Supports messages $\mu \in \{0,1\}$ and NAND operations up to an a priori bounded depth of NANDs
- Public key $M \in \mathbb{Z}_q^{m\times n}$ and private key $\underline{\mathbf{z}}$ s.t. $\underline{\mathbf{z}}^T M$ has small entries
- $Enc(\mu) = M^T R + \mu G$ where $R \leftarrow \{0,1\}^{m\times km}$ (and $G \in \mathbb{Z}_q^{n\times km}$ the matrix to reverse bit-decomposition)
- $Dec_z(C)$: $\underline{\mathbf{z}}^T C = \underline{\delta}^T + \mu \underline{\mathbf{z}}^T G$ where $\underline{\delta}^T = e^T R$
- $NAND(C_1,C_2)$: $G - C_1 \cdot B(C_2)$ (G is a (non-random) encryption of 1)
  - $\underline{\mathbf{z}}^T C_1 \cdot B(C_2) = \underline{\mathbf{z}}^T C_1 \cdot B(C_2) = (\underline{\delta}_1^T + \mu_1 \underline{\mathbf{z}}^T G)\ B(C_2)$
    $= \underline{\delta}_1^T B(C_2) + \mu_1 \underline{\mathbf{z}}^T C_2 = \underline{\delta}^T + \mu_1\mu_2 \underline{\mathbf{z}}^T G$
    where $\underline{\delta}^T = \underline{\delta}_1^T B(C_2) + \mu_1 \underline{\delta}_2^T$ has small entries
  - In general, error gets multiplied by km. Allows depth $\approx \log_{km} q$

Only "left depth" counts, since $\underline{\delta} \leq k \cdot m \cdot \underline{\delta}_1 + \underline{\delta}_2$

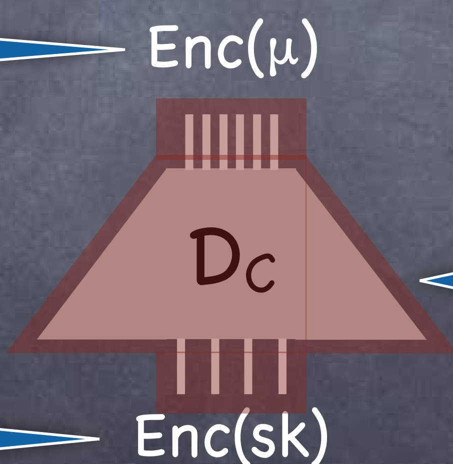
Recall

Bootstrapping

- To refresh a given ciphertext C. Also given an encryption of sk (in the public-key). Let D_C be s.t. $D_C(sk) := Dec(C,sk)$.
- Refresh(C,Enc(sk)) = HomomEval(D_C, Enc(sk))
- Need depth of D_C to be strictly less than the depth allowed by the homomorphic encryption scheme



Refreshed: Doesn't depend on how unfresh C was, but only on the depth of D_C



Homomorphic evaluation in the ciphertext space

Fresh encryption of sk, provided along with the public key

Recall

Bootstrapping for iO

- iO candidate from multi-linear map candidates, using matrix programs
  - Polynomial sized iO if polynomial-sized matrix programs
  - Barrington's Theorem: NC^1 functions have polynomial-sized matrix programs (with 5x5 matrices)
  - Can "bootstrap" from this to all polynomial-sized circuits/ polynomial-time computable functions, assuming Fully Homomorphic Encryption (with decryption in NC^1)

Bootstrapping for iO

- Idea: Carry out FHE (for polynomial depth) evaluation, and use obfuscated program to do decryption
  - Ciphertext will encode the function C, and input m can be given in the clear
  - Let U_m denote a (deep) circuit s.t. $U_m(C) = C(m)$
  - Obfuscation: (σ,π) where σ=FHE-Enc(C) and π=iO(P) where P is a low-depth program that decrypts an FHE ciphertext σ^*, but only if it is obtained by evaluating U_m homomorphically on σ (for some input m)
    - How can P ensure this without computing U_m itself?
    - P takes a <u>proof</u> that $\sigma^* = F(m') :=$ FHE-Eval$(U_{m'},\sigma)$ for some m'
      - Proof: σ^* and all wire values in circuit evaluating F(m'). Can verify each gate separately (in NC^0), and AND the results (in NC^1) to get the full verification result

Bootstrapping for iO

- Obfuscation: (PK,σ,π) where σ=FHE-Enc$_{PK}(C)$ and $\pi=iO(P)$
  - $P(\sigma^*,\varphi)$ = FHE-Dec$_{SK}(\sigma^*)$ if Verify$(\sigma^*,\varphi)=1$
  - Proof φ is for the claim: $\underline{\exists\ m' \text{ s.t. } \sigma^* = \text{FHE-Eval}_{PK}(U_{m'},\sigma)}$
- Evaluation: Compute σ^* and φ using m. Run $\pi(\sigma^*,\varphi)$ to get C(m)
- Secure? Need to hide representation of C
- But π may not hide the FHE decryption key SK!
- Idea: Have multiple representations of P s.t. some representations don't reveal SK or anything beyond C's functionality
- Will have $\sigma=(\sigma_1,\sigma_2)$, with $\sigma_i \leftarrow$ FHE-Enc$_{PK_i}(C)$. And the claim proven is $\underline{\exists\ m' \text{ s.t. } \sigma_1^* = \text{FHE-Eval}_{PK_1}(U_{m'},\sigma_1) \wedge \sigma_2^* = \text{FHE-Eval}_{PK_2}(U_{m'},\sigma_2)}$

Bootstrapping for iO

- Obfuscation: $(PK_1, PK_2, \sigma_1, \sigma_2, \pi)$ where $\sigma_i \leftarrow$ FHE-Enc$_{PK_i}(C)$ and π=iO(P_1)
  - $P_1(\sigma_1^*, \sigma_2^*, \varphi)$ = FHE-Dec$_{SK_1}(\sigma_1^*)$ if Verify$(\sigma_1^*, \sigma_2^*, \varphi)$=1
  - Proof φ for claim $\underline{\exists\ m' \text{ s.t. for } i=1,2,\ \sigma_i^* = \text{FHE-Eval}_{PK_i}(U_{m'}, \sigma_1)}$
- Evaluation: Compute σ_1^*, σ_2^*, φ using m. Run $\pi(\sigma_1^*, \sigma_2^*, \varphi)$ to get C(m)
- Consider functionally equivalent C_1 and C_2 and following "hybrids"
- 1. Obfuscation of C_1 : $\sigma_i \leftarrow$ FHE-Enc$_{PK_i}(C_1)$ and π=iO(P_1)
- 2. Uses $\sigma_i \leftarrow$ FHE-Enc$_{PK_i}(C_i)$
- 3. Uses π=iO(P_2) where P_2 uses SK_2 to decrypt σ_2^*
- 4. Uses $\sigma_i \leftarrow$ FHE-Enc$_{PK_i}(C_2)$
- 5. Uses π=iO(P_1). This is an honest obfuscation of C_2.

(1) ≈ (2): FHE security for SK_2

(2) ≈ (3): By iO. P_1, P_2 functionally equivalent!

(3) ≈ (4): FHE security for SK_1

(4) ≈ (5): Again by iO.

Discussion

That's All Folks!