“MPC in the Head”

Yuval Ishai
Technion and UCLA
Back to the 1980s

- Zero-knowledge proofs for NP [GMR85, GMW86]
- Computational MPC with no honest majority [Yao86, GMW87]
- Unconditional MPC with honest majority [BGW88, CCD88, RB89]
- Unconditional MPC with no honest majority assuming ideal OT [Kilian88]

- Are these unrelated?
Message of this talk

• Honest-majority MPC is useful even when there is no honest majority!
• Establishes unexpected relations between classical results
• New results for MPC with no honest majority
• New application domains for honest-majority tools and techniques
Allison

Research interests:
- zero-knowledge proofs
- efficient two-party protocols

Bernard

Research interests:
- information-theoretic cryptography
- honest-majority MPC

Some relevance

No relevance?
Allison

Research interests:
- zero-knowledge proofs
- efficient two-party protocols

Want to hear about my latest and coolest VSS protocol?

Bernard

Research interests:
- information-theoretic cryptography
- honest-majority MPC

what a dork...
Helping make the match

• Add to Allison’s world a simple ideal functionality
  – Ideal commitment oracle for ZK (Com-hybrid model)
  – Ideal OT oracle for general protocols (OT-hybrid model)
• Makes unconditional (and UC) security possible
  – Analogous to secure channels in Bernard’s world
• Why should Allison be happy?
  – Generality: Com or OT can be realized in a variety of models, under a variety of assumptions
  – Efficiency: Com or OT can be realized with little overhead
    • Essentially free given preprocessing [BG89]
    • Cheap preprocessing: fast OT […,PVW08,…], faster OT extension [Bea96,IKNP03…]

• Still: Why should Bernard’s research be relevant?
Helping make the match

• Add to Allison’s world a simple ideal functionality
  – Ideal commitment oracle for ZK (Com-hybrid model)
  – Ideal OT oracle for general protocols (OT-hybrid model)

• Makes unconditional security possible

• Why should Allison be happy?
  – Generality: Com or OT can be realized in a variety of models, under a variety of assumptions
  – Efficiency: Com or OT can be realized with little overhead

  • Essentially free given preprocessing [BG89]
  • Cheap preprocessing for fast OT [...] PVW08,...], faster OT extension [Bea96, IKNP03...]

• Still: Why should Bernard’s research be relevant?

A high level idea:

  • Run MPC “in the head”.
  • Commit to generated views.
  • Use consistency checks to ensure honest majority.
Zero-knowledge proofs

• Goal: ZK proof for an NP-relation $R(x,w)$
  – Completeness
  – Soundness
  – Zero-knowledge

• Towards using MPC:
  – define n-party functionality
    $$g(x; w_1,\ldots,w_n) = R(x, w_1 \oplus \ldots \oplus w_n)$$
  – use any 2-secure, perfectly correct protocol for $g$
    • security in semi-honest (passive adversary) model
    • honest majority when $n \geq 5$
Given MPC protocol $\pi$ for $g(x; w_1, \ldots, w_n) = R(x, w_1 \oplus \ldots \oplus w_n)$

accept iff output=1 & $V_i, V_j$ are consistent
Analysis

• Completeness: √
• Zero-knowledge: by 2-security of $\pi$ and randomness of $w_i, w_j$. (Note: enough to use $w_1, w_2, w_3$)
• **Soundness:** Suppose $R(x, w)=0$ for all $w$.
  
  - either (1) $V_1,\ldots,V_n$ consistent with protocol $\pi$
  - *or* (2) $V_1,\ldots,V_n$ not consistent with $\pi$

  (1) $\Rightarrow$ outputs=$0$ (perfect correctness)
  $\Rightarrow$ Verifier rejects

  (2) $\Rightarrow$ for some $(i,j)$, $V_i,V_j$ are inconsistent.
  $\Rightarrow$ Verifier rejects with prob. $\geq 1/n^2$. 

In fact, proof of knowledge
Analysis

**Prover**
- commit to views $V_1, \ldots, V_n$
- $w = w_1 \oplus \ldots \oplus w_n$
- random $i, j$
- open views $V_i, V_j$

**Verifier**
- accept iff output = 1
- $V_i, V_j$ are consistent

**Communication complexity:**
\[
\leq (\text{comm. complexity} + \text{rand. complexity} + \text{input size}) \text{ of } \pi.
\]
Extensions

• **Variant**: Use 1-secure MPC
  – Open one view and one incident channel
• Extends to **OT-based MPC**
  – Simple consistency check when $t \geq 2$
  – Slightly more involved with $t=1$ [HV16,IKPSY16]
• Extends to MPC with **error**
• **Variant**: Directly get $2^{-k}$ soundness error via security in malicious model (active adversary)
  – Two clients, $n=O(k)$ servers
  – $\Omega(n)$-security with abort
  – Broadcast is “free”
• Realize **Com** using a one-way function
Applications

- Simple ZK proofs using:
  - (1,3) semi-honest MPC \([\text{BGW88, CCD88}]\) or \([\text{Mau02}]\)
  - (2,3) or even (1,2) semi-honest MPC\(^\text{OT}\) \([\text{GMW87, GV87, GHY87}]\)
- Practical ZK proofs ("ZKBoo" \([\text{GMO16}]\))
- ZK proofs with \(O(|R|)+\text{poly}(k)\) communication
  - Using efficient MPC + AG codes \([\text{DI06, CC06}]\)
- Many good ZK protocols implied by MPC literature
  - ZK for linear algebra \([\text{CD01,}...]\)
General 2-party protocols [IPS08]

• Life is easier when everyone follows instructions...

• GMW paradigm [GMW87]:
  – semi-honest-secure $\pi \rightarrow$ malicious-secure $\pi'$
  – use ZK proofs to prove “sticking to protocol”

• Non-black-box: ZK proofs in $\pi'$ involve code of $\pi$
  – Typically considered “impractical”
  – Not applicable at all when $\pi$ uses an oracle
    • Functionality oracle: OT-hybrid model
    • Crypto primitive oracle: black-box PRG
    • Arithmetic oracle: black-box field or ring

• Is there a “black-box alternative” to GMW?
A dream goal

• Possible for some fixed $f$
  – e.g., OT [IKLP06,Hai08]
• Impossible for general $f$
  – e.g., ZK functionalities [IKOS07]
Idea

• Combine two types of “easy” protocols:
  – Outer protocol: honest-majority MPC
  – Inner protocol: semi-honest 2-party protocol
    • possibly in OT-hybrid model

• Both are considerably easier than our goal
• Both can have information-theoretic security
Secure against malicious adaptive adversary corrupting one client and t=ck servers, for some constant c>0.

Security with abort suffices.

Straight-line simulation.

Example: “BGW-lite”
Inner protocol

Client A holds input x

OT

Client B holds input y

Secure against semi-honest adversary
(Adaptive security w/erasures)
Example: “GMW-lite”
Combining the two protocols

OT calls by inner protocol are “risky”

outer protocol for f

Player virtualization

oblivious watch lists
A closer look at server emulation

• Assume servers are deterministic
  – This is already the case for natural protocols
  – Can be ensured in general with small overhead
• In outer protocol, server $i$
  – gets messages from A and B
  – sends messages to A and B
  – may update a secret state
• Captured by reactive 2-party functionality $F_i$
  – Inputs = incoming messages
  – Outputs = outgoing messages
• Use semi-honest protocol for $F_i$
  – Distribute server between clients
  – “Local” computations do not need to be distributed.
A closer look at watchlists

• Inner protocol can’t prevent clients from cheating by sending “bad messages”
• Watchlist mechanism ensures that cheating does not occur too often
  – Client doesn’t know which instances of inner protocol are watched
  – Two cases:
    • Client cheats in $\leq t$ instances
      $\Rightarrow$ cheating is tolerated by $t$-security of outer protocol
    • Client cheats in $> t$ instances
      $\Rightarrow$ will be caught with overwhelming probability

• Non-interactive form of “cut-and-choose”
Setting up the watchlists

- Each client picks $n$ long one-time pads $R_i$
- $|R_i| = \text{length of messages} + \text{randomness in execution of } i\text{-th inner protocol}$
  - Short PRG seed suffices for computational security
- Each client uses OT to select $\sim t/2$ of the other client’s pads $R_i$
- Implemented via Rabin-OT for each server
  - Reduces to a constant number of $(1,2)$ string-OTs per server for any rational probability $p$
  - With overwhelming probability, $p \pm 0.01$ fraction of $R_i$ are received
Using the watchlists

- Consider here B watching A
  - A watches B symmetrically

- A uses sequential parts of each $R_i$ to mask her (progressive) view of the i-th inner protocol
  - If B obtained $R_i$, he has full view of i-th inner protocol
  - Can detect (and abort) as soon as A cheats
  - What about ideal OT calls in inner protocol?
    - Cheating caught w/prob $\frac{1}{2}$ if OT inputs are random
    - Use OT to random-OT reduction
Example

• Consider a “BGW-style” outer protocol
• Each server performs two types of computations:
  – Send $a_i b_i + z_i$ to A, where $a_i$ is a secret received from A and $b_i, z_i$ are secrets received from B
    • $O(|C|)$ such computations overall
    • Can be implemented by simple inner protocols
      – unconditionally using OT [GMW87,IPS09]
      – using homomorphic encryption (e.g., Paillier)
      – using coding assumptions and OT [NP99,IPS09]
  – Send to A a public linear combination of secrets sent by B (and vice versa)
    • Can be implemented via local computation of B
• Gives efficient protocols for arithmetic computations
Simulation (rough idea)

• Suppose A is corrupted in final protocol
• Main simulator runs outer simulator to
  – extract input of A
  – generate outer protocol messages from B
  – generate full view of inner protocols watched by A (requires corrupting ~ t/2 servers)
  – generate A’s inputs and outputs in other inner protocols (communication of A with servers)
    • feed to inner simulator to generate inner protocol view
    • valid as long as A does not deviate from inner protocol
• Main simulator can observe deviation from inner protocol
  – When A cheats on i-th inner protocol, outer simulator corrupts i-th server and main simulator aborts w/prob. p
A general protocol compiler

- **Given a $m$-party functionality $F$**
  - Get an honest-majority-secure outer protocol $\Pi$ for the functionality $F$ (with $m$ clients and $k$ servers)
  - Get a semi-honest-secure inner protocol $\rho^{OT}$ for a $m$-party functionality $G^{\Pi}$ corresponding to the servers’ program in $\Pi$

  ($G^{\Pi}$ is a reactive functionality defined black-box w.r.t $\Pi$)

- **Our ($m$-party) protocol $\Phi^{OT}$, with black-box access to $\Pi$ and $\rho$, is a malicious-secure protocol for $F$.**
Applications

• Revisiting the classics
  – BGW-lite + GMW-lite ➔ Kilian

• Efficient MPC with no honest majority
  – $O(1)$ bits per gate in OT-hybrid model (+ additive term)
  – All crypto can be pushed to preprocessing

• **Constant-round** MPC$^{\text{OT}}$ (t<n) using **black-box** PRG
  – Extending 2-party “cut-and-choose” Yao

• Efficient OT extension in malicious model

• Constant-rate b.b. reduction of OT to semi-honest OT

• Secure arithmetic computation over black-box fields/rings

• Protocols making black-box use of homomorphic encryption
More “MPC in the Head”: OT combiners and OT extractors

- **OT combiners** [HKNRR05]
  - Given \( n \) instances of OT, of which \( t \) are faulty, produce \( m \) good OTs
  - Can be obtained via honest-majority MPC [HIKN08,IPS08]
    - **Outer protocol**: honest-majority MPC for \( m \) OTs
    - **Inner protocol**: OT-based 2-party protocol for emulating MPC server
  - Used for constant-rate **OT from noisy channels** [HIKN08,IKOPSW11]

- **OT extractors** [IKOS09]
  - Generalize OT combiners by allowing global leakage
  - Construction makes an ad-hoc use of suitable “outer protocol” and “inner protocol”
  - Yield constant-rate OT protocols from imperfect noisy channels, constant-rate OT from (computational) “\( \theta \)-Hiding assumption”.
OT Extractor

OT Combiner

Randomness extractor

Extractor for bit-fixing sources

Arithmetic codes

Random codes
More “MPC in the Head”: Non-Interactive Secure Computation

• Goal: Protect non-interactive OT-based protocols against malicious sender

• Challenge: allow Receiver to detect when Sender’s OT inputs are inconsistent with protocol
More “MPC in the Head”: Non-Interactive Secure Computation

- An MPC-based approach [IKOPS11]

Protect against “correlated abort” attacks by encoding receiver’s input [Kil98,LP07,IKOPS11]
Further research I

• Find other useful “black-box” connections
• Formalized via oracle game:
  – Protocol move:
    given oracle g, get (arbitrary) protocol oracle \( \pi_g \)
  – Build move:
    given oracle f, build oracle g
  – Goal: given oracle f, obtain a protocol \( \pi_f \) in a “strong” model using only protocol moves in “weaker” model(s)

• Previous examples
  – ZK from MPC:
    build – protocol – build
  – New protocol compiler:
    protocol – build – protocol - build
Further Research

• Other useful “black-box” connections?
  – Formalized via “MPC transformations” framework [IKPSY16]
  – Gives hope for proving negative results

• Find leaner versions of protocol compilers
  – Weaker outer protocol?

• Minimize constants in constant-rate protocols
  – Better “arithmetic codes”?

• Optimize for practical efficiency?
  – Many degrees of freedom!
  – Progress made in [LOP11]