

MiniMPC: Threshold Schemes for (and from) MiniCrypt

Xiao Wang (MPC MINIONS, Northwestern University)

MPTS2026 Jan 28th

Team



Hongrui Cui (Shanghai Jiao Tong University)

Chun Guo (Shandong University)

Xiaojie Guo (Shanghai Qi Zhi Institute)

David Heath (UIUC)

Jonathan Katz (Google)

Vlad Kolesnikov (Georgia Institute of Technology)

Alex Malozemoff (Galois Inc.)

Samuel Ranellucci (Coinbase)

Mike Rosulek (Oregon State University)

Lance Roy (Aarhus University)

Xiao Wang (Northwestern University)

Chenkai Weng (Arizona State University)

Kang Yang (State Key Laboratory of Cryptology)

Yu Yu (Shanghai Jiao Tong University, Shanghai Qi Zhi Institute)

Team



Hongrui Cui (Shanghai Jiao Tong University)

Chun Guo (Shandong University)

Xiaojie Guo (Shanghai Qi Zhi Institute)

David Heath (UIUC)

Jonathan Katz (Google)

Vlad Kolesnikov (Georgia Institute of Technology)

Alex Malozemoff (Galois Inc.)

Samuel Ranellucci (Coinbase)

Mike Rosulek (Oregon State University)

Lance Roy (Aarhus University)

Xiao Wang (Northwestern University)

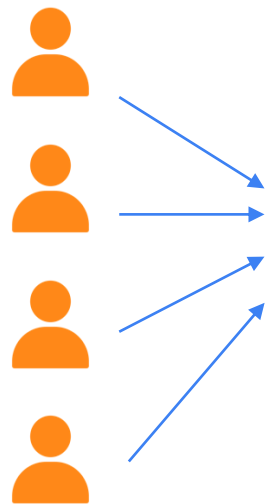
Chenkai Weng (Arizona State University)

Kang Yang (State Key Laboratory of Cryptology)

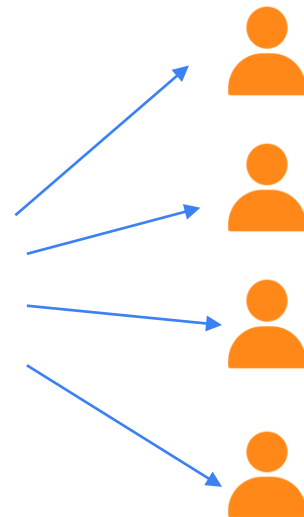
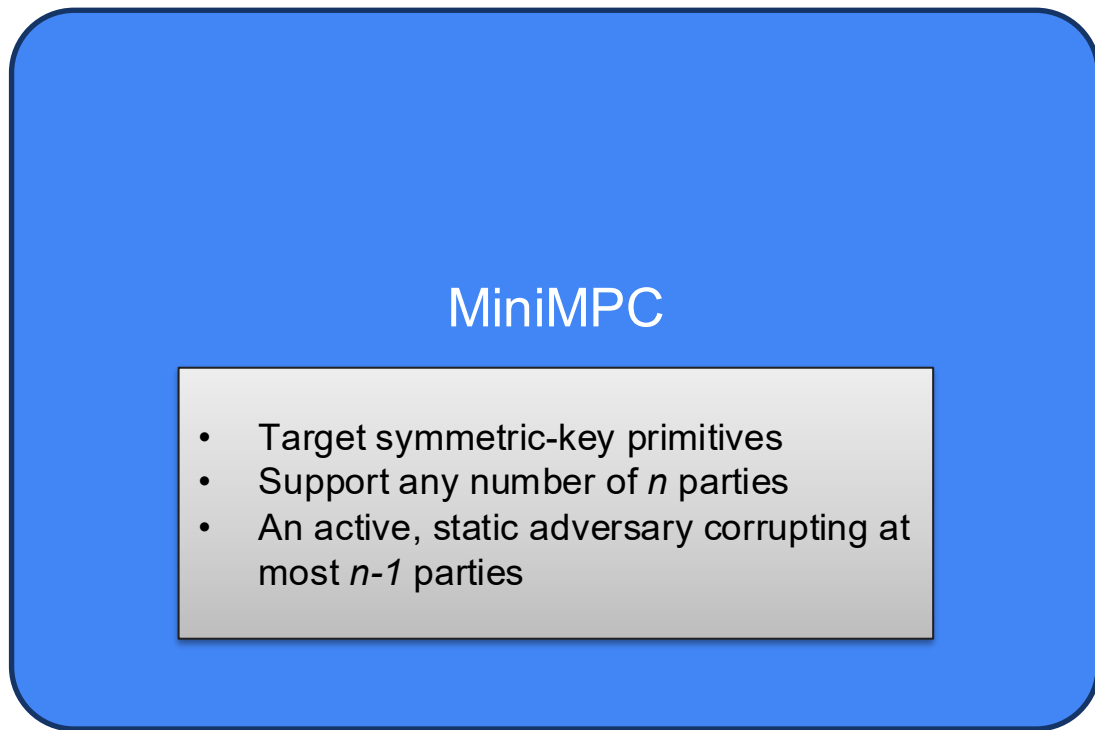
Yu Yu (Shanghai Jiao Tong University, Shanghai Qi Zhi Institute)

Junior researchers highlighted

Overview

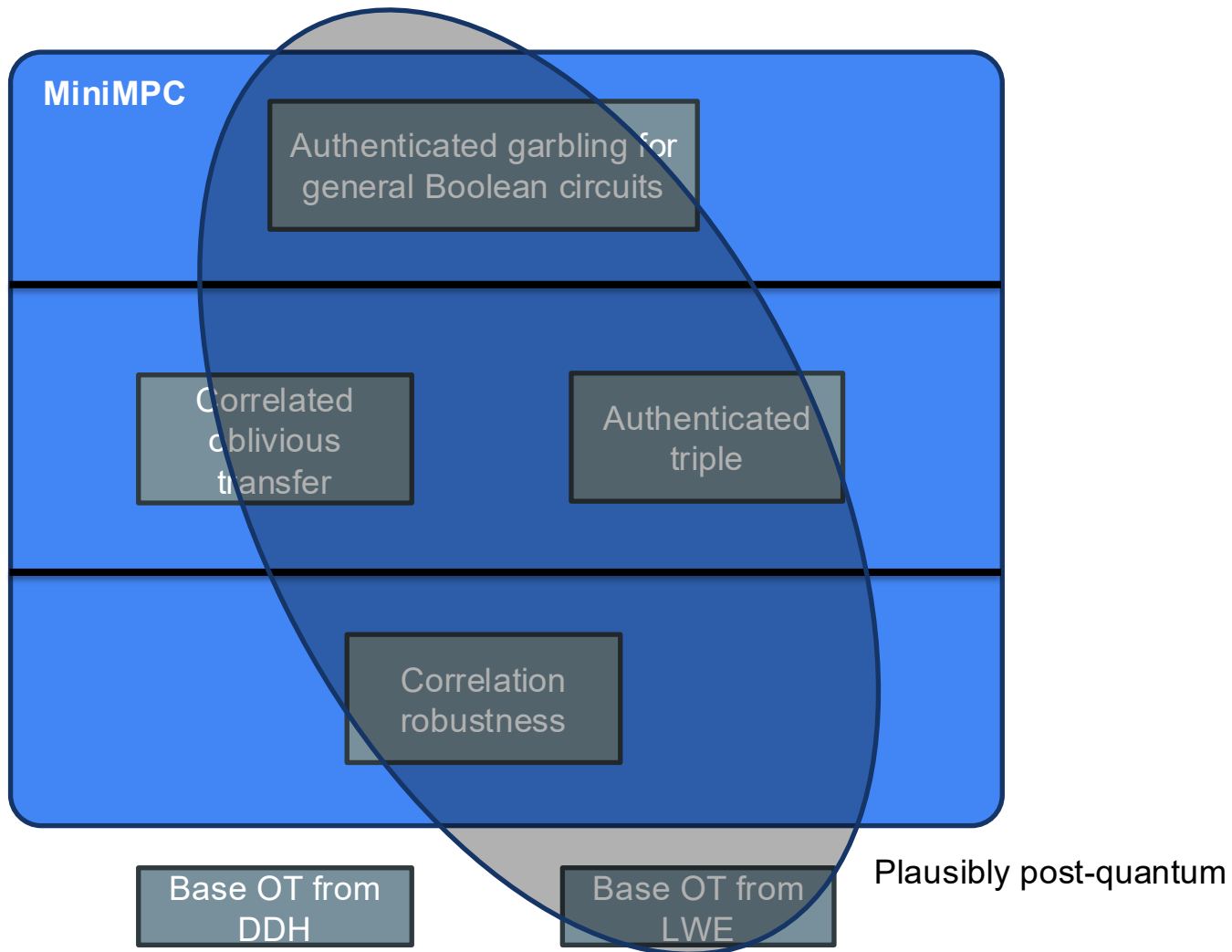


Authenticated share
of keys



Authenticated share
of ciphertext

Overview



Why the title?

Threshold Schemes for (and from) MiniCrypt

- We mostly focus on threshold symmetric-key primitives
 - More suitable to be computed in Boolean circuits
- We only use symmetric-key primitives
 - We provide implementations of base OT but we don't treat them as part of our submission

Our Philosophy

- **Modular**
 - Make building blocks reusable by other teams and future applications
 - Non-trivial in some cases
- **Conservative**
 - Only needs random oracle and ideal cipher
 - All instantiated using standardized primitives
- **Security**
 - UC security
 - Concrete security level
- **Efficiency**
 - Hardware-oriented protocol optimizations
 - Instruction-level accelerations

Authenticated Bits

MAC x

$$x \oplus x = x\Delta_B$$

x Key



A bit : x



Authentication
key: Δ_B

- AKA correlated OT (COT)
- A lot of COT can be generated from a small set of “base OT” – COT extension
 - SoftspokenOT: generate one COT with λ/k bits and $O(2^k/k)$ computation.

Authenticated Shares

Δ_A

Δ_B

MAC x_1

$$x = x_1 \oplus x_2$$

x_1 Key



Key x_2



x_2 MAC

Only knows x_1

Only knows x_2

$$x_1 \oplus x_1 = x_1 \Delta_B$$

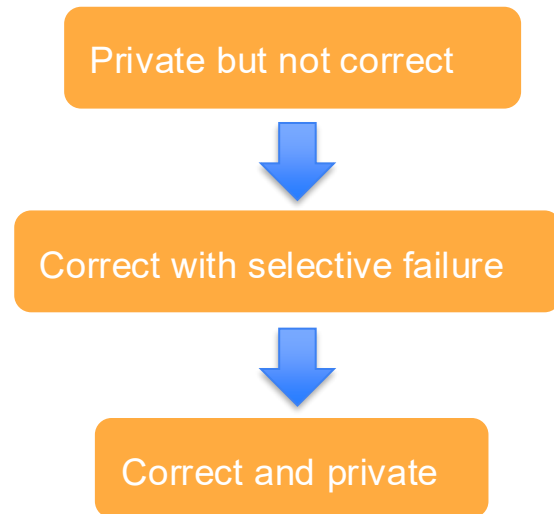
$$x_2 \oplus x_2 = x_2 \Delta_A$$

Authenticated triples

[NNOB12,FKOS15,WRK17,KRRW18]



$$z_1 \oplus z_2 = (x_1 \oplus x_2) \wedge (y_1 \oplus y_2)$$



Selective-failure Attack in Semi-honest Yao

garbled table

$$H(L_{\alpha,0}, L_{\beta,0}) \oplus L_{\gamma,0}$$

$$H(L_{\alpha,0}, L_{\beta,1}) \oplus L_{\gamma,0}$$

$$H(L_{\alpha,1}, L_{\beta,0}) \oplus L_{\gamma,0}$$

$$H(L_{\alpha,1}, L_{\beta,1}) \oplus L_{\gamma,1}$$

Corrupt

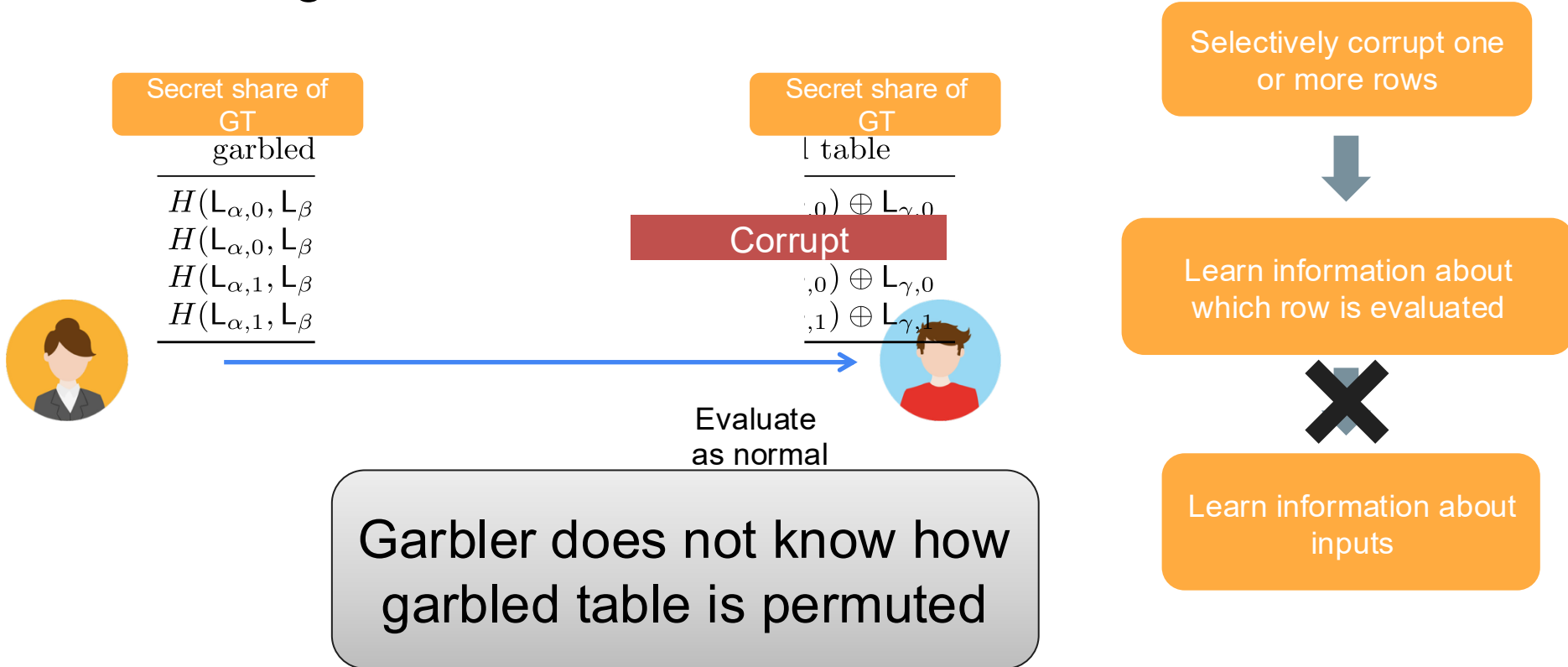
Selectively corrupt one or more rows

Learn information about which row is evaluated

Learn information about inputs

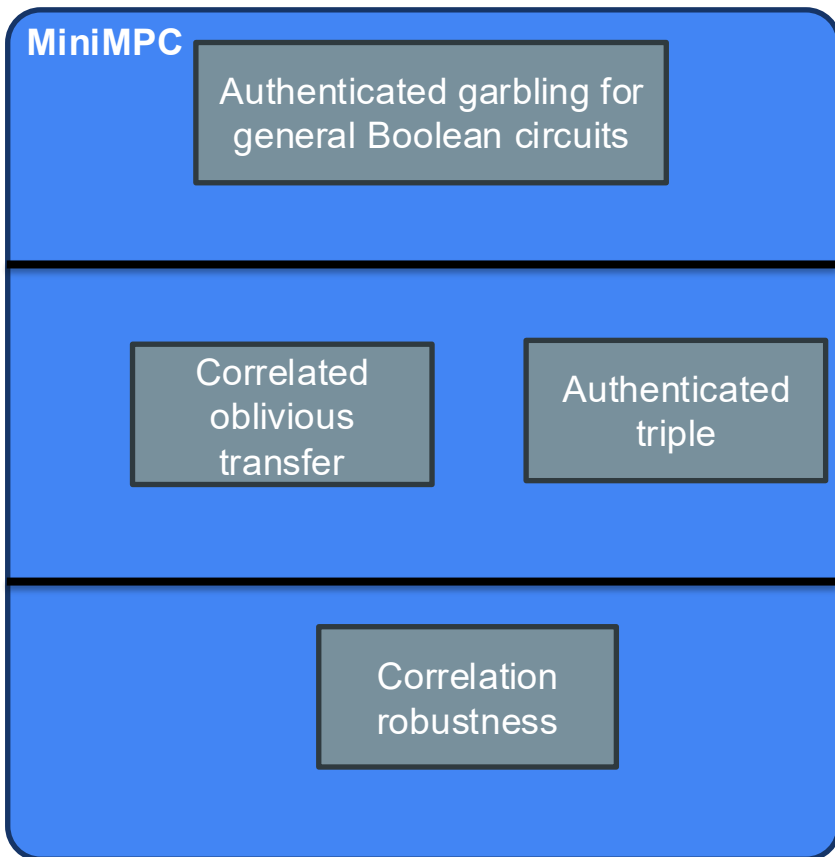
Garbler knows how rows are permuted

Preventing Selective-failure Attack in authenticated GC



Authenticated GC

- One authenticated AND can be used to distributivity garble one AND gate



Correlation Robustness

- We need to “hash” everywhere
 - COT
 - Authenticated triple
 - Authenticated GC
- Random Oracle is sufficient (and convenient) but not efficient.
 - We are hashing tens of millions of times per second
- Use correlation robust hash function [BHKR13] but
 - Take care of concrete security [GKWWY20]
 - Take care of composition [GKWY20, ??]

Plan and Progress

- Protocol Design & Proof (99%)
 - Write Up (80%)
 - Implementation (40%)
 - Benchmark (0% 😊)
-
- If more revisions/rounds/iterations:
 - Fancier protocols
 - Tailored protocols
 - Other assumptions