Secure Two-party Computation

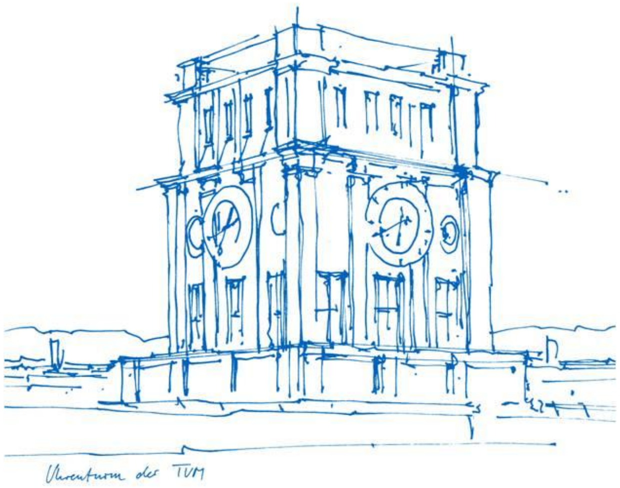
Jan Lauinger

Technical University of Munich

TUM Department of Electrical and Computer Engineering

Associate Professorship of Embedded Systems and Internet of Things

Munich, September 2024



What is secure computation?

Collaboratively compute a function and maintain input secrecy

- **Multi-party Computation (MPC)**: MPC is the joint computation of a public function $f(\mathbf{x})=y$ between M parties with input x_M such that no party learns the private inputs of the counterparties
  - The **adversary** is assumed to corrupt T parties.
- **Secure two-party computation (2PC)**: Secure 2PC uses M=2 and T=1 such that two parties with private inputs x_1, x_2 can collaboratively compute $f(x_1, x_2)$ without learning any other x_i

Adversarial behavior model

Assumption

- **Semi-honest parties** honestly follow the protocol specification
  - Attack/Try to learn from exchanged parameters

- **Malicious adversaries** arbitrarily deviate from the protocol specification
  - Inject false values such that the opponent accepts values without notice
  - Selective-failure attack: Inject false values, then observe and learn from failure
    - (e.g. know secret permutation & corrupt a row, learn which row was evaluated, learn information on inputs)

MPC and cryptographic building blocks

Constructions

- **Garbled circuits**
  - Based on oblivious transfer (OT), OT cost depends on input size
  - Two-party computation
  - Constant communication (independent of circuit depth)

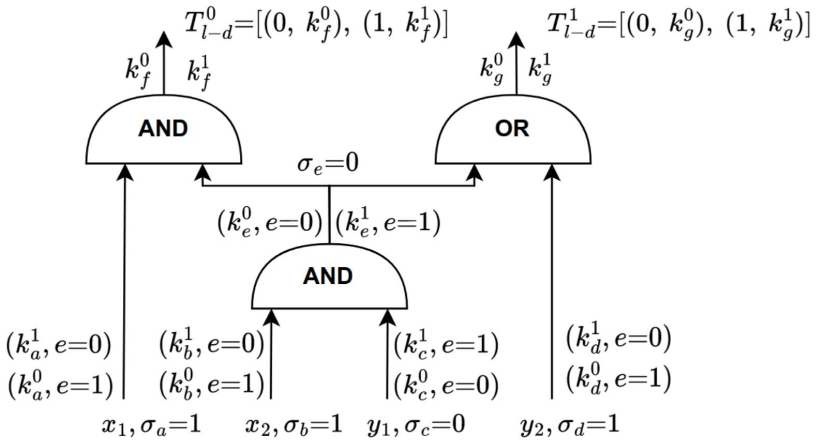
- **GMW, BGW, CCD**
  - Based on secret sharing
  - Multi-party computation
  - Boolean (AND, OR) or arithmetic (MUL, ADD) circuits
  - Gate-by-gate computation in multiple oblivious communication rounds for MUL operation

SoK: Compilers for MPC
https://www.cis.upenn.edu/~stevez/papers/HHNZ19.pdf

2PC based on garbled circuits

Garbled circuit parameters

- **Parties**: Garbler and evaluator
- **Boolean circuit**: AND, OR gates
- **Wire labels**: k, i, e, sigma
- **Private labels**: i (internal), sigma
- **Public labels**: e (external), k^i
- **Random labels**: sigma, k^i



	$G(\mathcal{C}^{0,(1,2)}_{\text{AND}})$
0,0	$E_{k^1_b}(E_{k^0_c}(k^0_e \| 0))$
0,1	$E_{k^1_b}(E_{k^1_c}(k^1_e \| 1))$
1,0	$E_{k^0_b}(E_{k^0_c}(k^0_e \| 0))$
1,1	$E_{k^0_b}(E_{k^1_c}(k^0_e \| 0))$

	$G(\mathcal{C}^{1,(0,1)}_{\text{AND}})$
0,0	$E_{k^1_a}(E_{k^0_e}(k^0_f))$
0,1	$E_{k^1_a}(E_{k^1_e}(k^1_f))$
1,0	$E_{k^0_a}(E_{k^0_e}(k^0_f))$
1,1	$E_{k^0_a}(E_{k^1_e}(k^0_f))$

	$G(\mathcal{C}^{1,(1,3)}_{\text{OR}})$
0,0	$E_{k^0_e}(E_{k^1_d}(k^1_g))$
0,1	$E_{k^0_e}(E_{k^0_d}(k^0_g))$
1,0	$E_{k^1_e}(E_{k^1_d}(k^1_g))$
1,1	$E_{k^1_e}(E_{k^0_d}(k^1_g))$

External labels e

Protocol to evaluate semi-honest 2PC circuit

Example: Garbler input **x**=[x_1=1,x_2=0], Evaluator input **y**=[y_1=0,y_2=1], $\sigma_a=1,\sigma_b=1,\sigma_c=0,\sigma_d=1$

1. **Garbler garbles**: sample sigmas, k_L^i with L ∈ {a,b,c,d,e,f,g} (16B with aes128), compute e_L = sigma xor i, compute G tables, send (**G,** circuit C, (k^1_a, e=0), (k^0_b, e=1), $T_{l\text{-}d}$)

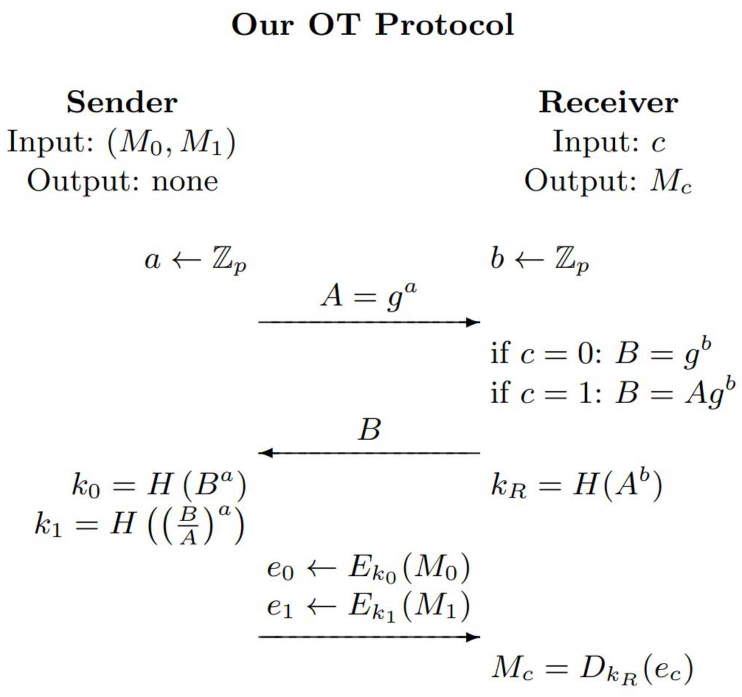
2. **Garbler & Evaluator using OT**: for every input wire corresponding to y input bit, run
   1-out-of-2 OT with (m_1=[k^0_L, e], m_2=[k^1_L, e]).

   $OT_{c,d}$ with b_c=0, b_d=1 yields (k^0_c, e=0), (k^1_d, e=0)

Oblivious Transfer

Transfer 2 messages without learning which message is picked

- 1-out-of-2 OT with 2 messages k_0, k_1
- Evaluator obtains (k_b ,e), with b ∈ {0,1}



The simplest protocol for OT:
https://eprint.iacr.org/2015/267.pdf

Protocol to evaluate semi-honest 2PC circuit

Example: Garbler input **x**=[x_1=1,x_2=0], Evaluator input **y**=[y_1=0,y_2=1], σ_a=1,σ_b=1,σ_c=0,σ_d=1

2. **Garbler & Evaluator using OT**: for every input wire corresponding to y input bit, run 1-out-of-2 OT with (m_1=[k^0_L, e], m_2=[k^1_L, e]).

   $OT_{c,d}$ with b_c=0, b_d=1 yields (k^0_c, e=0), (k^1_d, e=0)

3. **Evaluator use G tables to evaluate**

   Use G_{AND0} and (k^0_b, e=1), (k^0_c, e=0) -> row (1,0) to obtain (k^0_e, e=0)

   Use G_{AND1} and (k^1_a, e=0), (k^0_e, e=0) -> row (0,0) to obtain (k^0_f)

   Use G_{OR1} and (k^0_e, e=0), (k^1_d, e=0) -> row (0,0) to obtain (k^1_g)

   Use $T_{l\text{-}d}$ map to obtain out_0=0 from k^0_f and out_1=1 from k^1_g

Protocol to evaluate semi-honest 2PC circuit

Example: Garbler input $\mathbf{x}=[x_1=1,x_2=0]$, Evaluator input $\mathbf{y}=[y_1=0,y_2=1]$, $\sigma_a=1,\sigma_b=1,\sigma_c=0,\sigma_d=1$

3. **Evaluator use G tables to evaluate**

   …

   Use $T_{l\text{-}d}$ map to obtain $out_0=0$ from k^0_f and $out_1=1$ from k^1_g

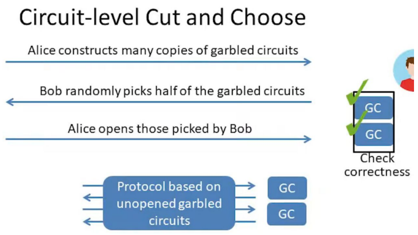
4. **Share output back to garbler**

   With ($out_0=0$, $out_1=1$) garbler knows wire keys at output labels

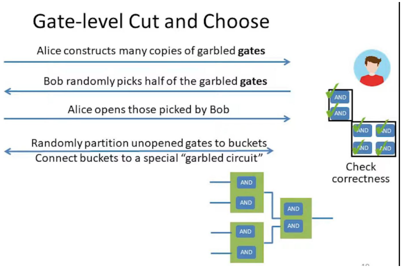
   However, ambiguity of wire keys in G_{AND1} together with OT obfuscates input of evaluator

From semi-honest to maliciously secure 2PC

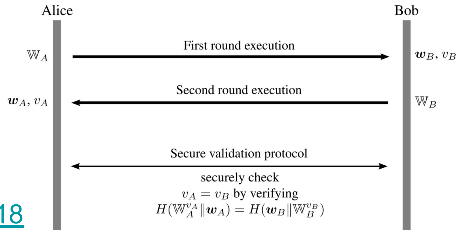
Techniques to secure 2PC against malicious adversaries



- **Cut-and-choose**: many copies of garbled circuits, validate random subset, use unopened circuits
  - Exist at a circuit and gate level



- **Dual execution**: two rounds semi-honest 2pc + secure validation
- **Authenticated garbling**: malicious secret sharing of GC permutation bits
  - Based on TinyOT protocol



Xiao Wang: Authenticated Garbling for Efficient Maliciously Secure
https://www.youtube.com/watch?v=8zCqki-ilZM

Quid-Pro-Quo-tocols
https://ieeexplore.ieee.org/abstract/document/6234418

Code example

MPC repository

Thank You

Questions?