Secure Multiparty Computation

Roberto Guanciale
KTH

Goal

- Secure Multyparty Computation
  - (privacy-preserving computation)
- Yao's Millionaires' Problem
  - Two millionaires, Alice and Bob, who are interested in knowing which of them is richer without revealing their actual wealth
  - F(d1,d2) = (d1 > d2)
- Generalization
  - N participants p1, … , pn
  - Have private data d1, …, dn
  - Want to compute the public function F(d1, … , dm)
  - While preserving input secrecy

Yao, Protocols for secure computations (1982)

Content

- Oblivious transfer: F(d1,d2) = d1[d2]
- Garbled circuits: n = 2
- (not today) Additive secret sharing: n = 3
- (not today) Shamir secret sharing: n > 2
- (not today) Homomorphic encryption
- ...
- **In the following we assume honest but curious adversary!**
- ...

Oblivious Transfer

Even, Goldreich, and Lempel

A Randomized Protocol for Signing Contracts (1985)

Oblivious Transfer

- Alice has two messages : $d_1 = [m_0, m_1]$
- Bob has one bit : $d_2 = b$
- Bob wishes to receive m_b without Alice learning b
- Alice wants to ensure that the receiver receives only one of the two messages

Oblivious Transfer

1. Alice generates pub PU priv PR keys and sends PU
2. Alice generates and sends two randoms $X=[X_0, X_1]$
3. Bob generates random r and blinds its encryption with X_b
    a. Bob sends (finite field arithmetic) $v = X_0 + Enc(r, PU)$ (let $b=0$)
4. Alice does not know if Bob has chosen X_0 or X_1
    a. $r_0 = Dec(v - X_0, PR)$ $r_1 = Dec(v - X_1, PR)$
    b. $r_0 = r$ $r_1 = Dec(X_0 + Enc(r, PU) - X_1, PR)$
5. Alice sends $m_0' = m_0 + r_0$ and $m_1' = m_1 + r_1$
6. Bob computes $m_0 = m_0' - r$ and discard m_1'

Oblivious Transfer

1. Alice generates pub PU priv PR keys and sends PU
2. Alice generates and sends two randoms $X=[X_0, X_1]$
3. Bob generates random r and blinds its encryption with X_b
   a. Bob sends (finite field arithmetic) $v = X_0 + Enc(r, PU)$ (let $b=0$)
4. Alice does not know if Bob has chosen X_0 or X_1
   a. $r_0 = Dec(v - X_0, PR)$ $r_1 = Dec(v - X_1, PR)$
   b. $r_0 = r$ $r_1 = Dec(X_0 + Enc(r, PU) - X_1, PR)$
5. Alice sends $m_0'=m_0+r_0$ a $m_1'=m_1+r_1$
6. Bob computes $m_0=m_0'-r$ card m_1'

One of the two r_i is correct. The other is a “random” number that cannot be controller by Bob

Oblivious Transfer

1. Alice generates pub PU priv PR keys and sends PU
2. Alice generates and sends two randoms $X=[X_0, X_1]$
3. Bob generates random r and blinds its encryption with X_b
    a. Bob sends (finite field arithmetic) $v = X_0 + Enc(r, PU)$ (let $b=0$)
4. Alice does not know if Bob has chosen X_0 or X_1
    a. $r_0 = Dec(v - X_0, PR)$ $r_1 = Dec(v - X_1, PR)$
    b. $r_0 = r$ $r_1 = Dec(X_0 + Enc(r, PU) - X_1, PR)$
5. Alice sends $m_0' = m_0 + r_0$ and $m_1' = m_1 + r_1$
6. Bob computes $m_0 = m_0' - r$ and discard m_1'

$m_1' - r$ will result in a random number

Garbled Circuits

Yao, How to generate and exchange secrets (1986)

Goldreich, Cryptography and Cryptographic Protocols (2003)

Garbled circuits

Used for 2-party computations

1. Function F(d1, d2) is compiled to a boolean circuit C
   a. Possibly by a third party
2. Alice garbles (encrypts) the circuits and d1
   a. Alice sends the circuit to Bob
3. Bob receives the encryption of d2 from Alice via OT
4. Bob executes the circuit
   a. Bob sends the result to Alice
5. Alice decrypts the result

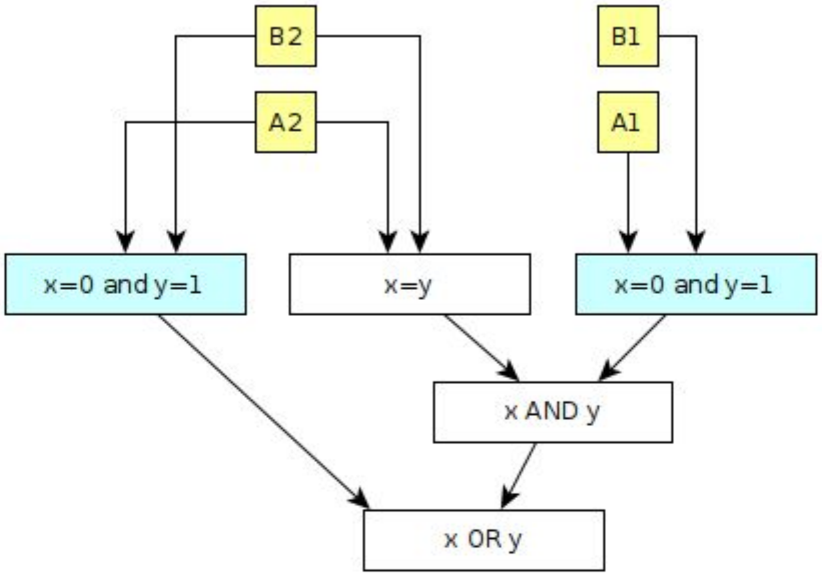
Garbled circuits

Used for 2-party computations

1. Function F(d1, d2) is compiled to a boolean circuit C
    a. Possibly by a third party
2. Alice garbles (encrypts) the circuits and d1
    a. Alice sends the circuit to Bob
3. Bob receives the encryption of d2 from Alice via OT
4. Bob executes the circuit
    a. Bob sends the result to Alice
5. Alice decrypts the result

1. Compilation to boolean circuits

- $A_1 < B_1$
  - X_1 = not A_1 and B_1
- $A_2A_1 < B_2B_1$
  - R_2 = not A_2 and B_2
  - $S_{\_2}$= (A_2 == B_2)
  - $T_{\_1}$= $S_{\_2}$ and X_1
  - X_2= $T_{\_1}$ or R_2
- ...



Garbled circuits

Used for 2-party computations

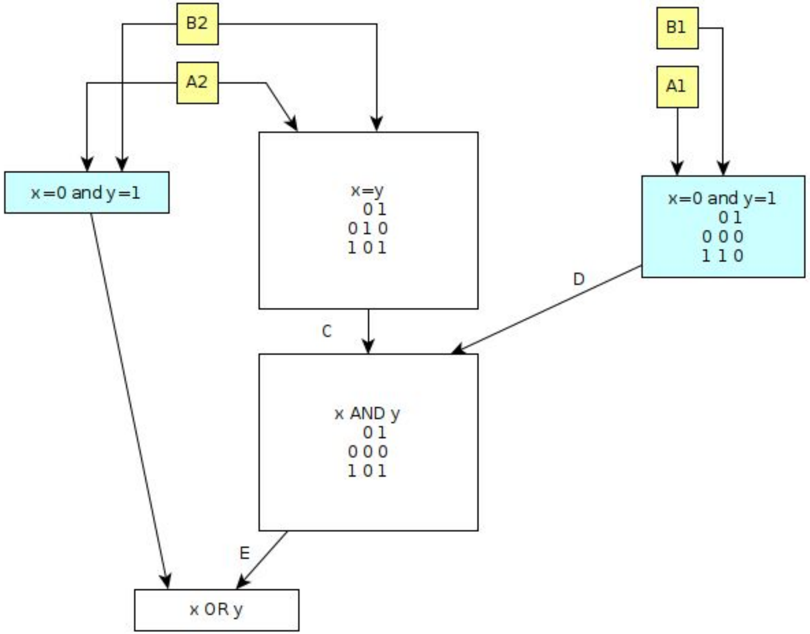
1. Function F(d1, d2) is compiled to a boolean circuit C
   a. Possibly by a third party
2. Alice garbles (encrypts) the circuits and d1
   a. Alice sends the circuit to Bob
3. Bob receives the encryption of d2 from Alice via OT
4. Bob executes the circuit
   a. Bob sends the result to Alice
5. Alice decrypts the result

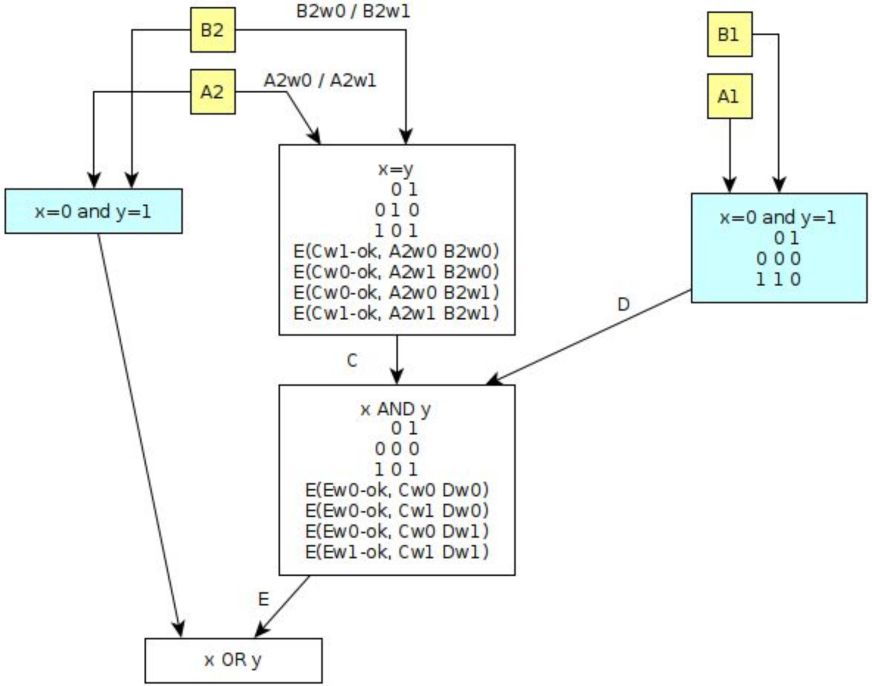
2. Circuit Garbling (Alice)

1. For every wire w, generate X_w^0 and X_w^1 to represent false and true
2. For every gate with input wire a,b and output wire c, substitute the truth table
3. Encrypt the table
4. Permute the table

a.	0 0 -> 0	$X_a^0\ X_b^0 \to X_c^0$	$Enc(X_c^0{:}OK,\ X_a^0\ X_b^0)$	$Enc(X_c^1{:}OK,\ X_a^1\ X_b^1)$
b.	0 1 -> 0	$X_a^0\ X_b^1 \to X_c^0$	$Enc(X_c^0{:}OK,\ X_a^0\ X_b^1)$	$Enc(X_c^0{:}OK,\ X_a^0\ X_b^0$
c.	1 0 -> 0	$X_a^1\ X_b^0 \to X_c^0$	$Enc(X_c^0{:}OK,\ X_a^1\ X_b^0)$	$Enc(X_c^0{:}OK,\ X_a^1\ X_b^0)$
d.	1 1 -> 1	$X_a^1\ X_b^1 \to X_c^1$	$Enc(X_c^1{:}OK,\ X_a^1\ X_b^1)$	$Enc(X_c^0{:}OK,\ X_a^0\ X_b^1)$

2. Circuit Garbling (Alice)





Garbled circuits

Used for 2-party computations

1. Function F(d1, d2) is compiled to a boolean circuit C
   a. Possibly by a third party
2. Alice garbles (encrypts) the circuits and d1
   a. Alice sends the circuit to Bob
3. Bob receives the encryption of d2 from Alice via OT
4. Bob executes the circuit
   a. Bob sends the result to Alice
5. Alice decrypts the result

3. Encryption of d2

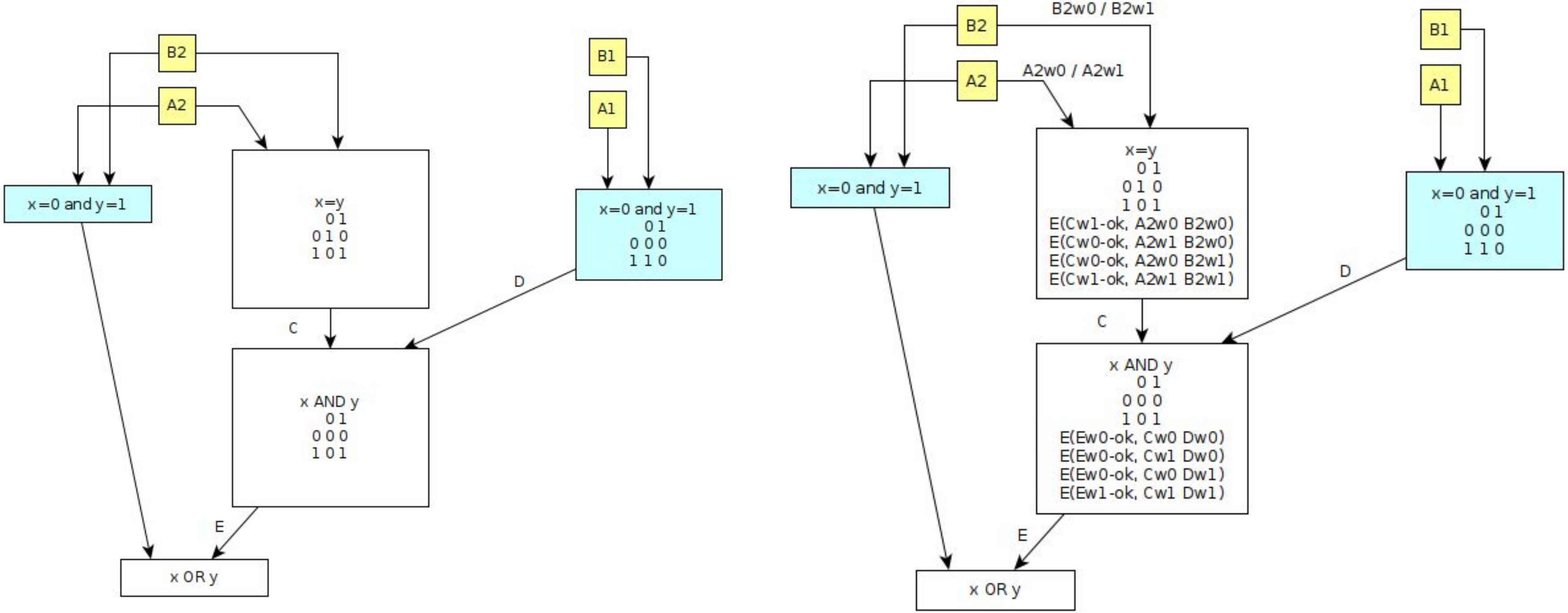
- For every bit d[i] of d2
  - Alice has X_w^0 and X_w^1 of the corresponding wires
  - Bob uses OT to get $X_w^{d[i]}$

Garbled circuits

Used for 2-party computations

1. Function F(d1, d2) is compiled to a boolean circuit C
    a. Possibly by a third party
2. Alice garbles (encrypts) the circuits and d1
    a. Alice sends the circuit to Bob
3. Bob receives the encryption of d2 from Alice via OT
4. Bob executes the circuit
    a. Bob sends the result to Alice
5. Alice decrypts the result

2. Circuit Garbling (Alice)

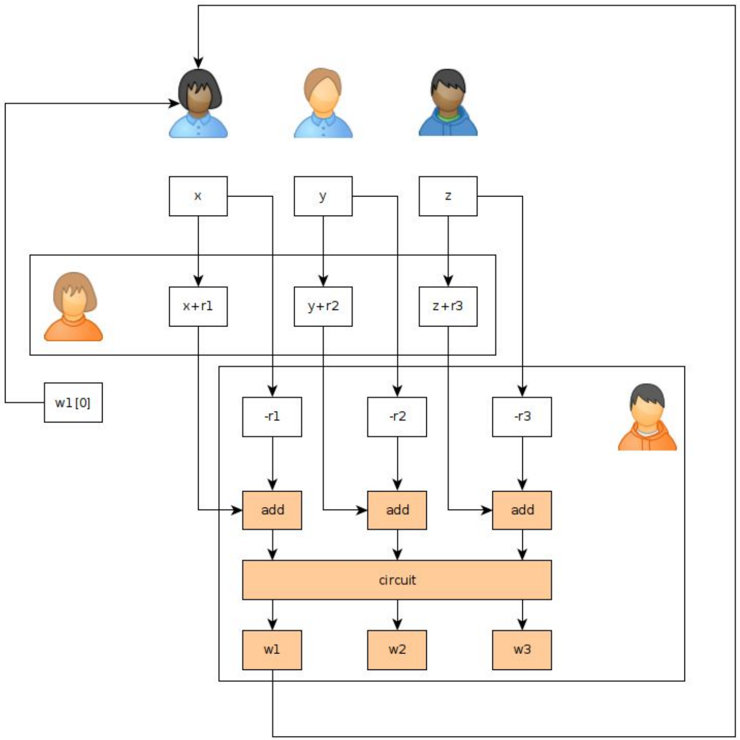


Garbled circuits

Used for 2-party computations

1. Function F(d1, d2) is compiled to a boolean circuit C
   a. Possibly by a third party
2. Alice garbles (encrypts) the circuits and d1
   a. Alice sends the circuit to Bob
3. Bob receives the encryption of d2 from Alice via OT
4. Bob executes the circuit
   a. Bob sends the result to Alice
5. Alice decrypts the result

If more than 2 Parties...



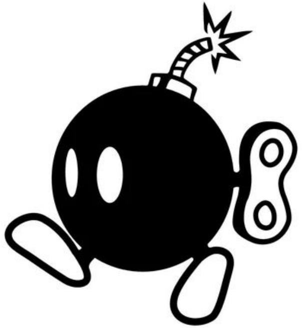
Secret Sharing

How to securely perform computations on secret-shared data

Secret sharing

---

1. Shamir secret sharing
   a. y out of x
2. Additive secret sharing
   a. y out of y



Additive secret sharing: 3 party computation

- S1 + S2 + S3 = V
- Secure if 2 parties collude
- To input (share) V you generate two random numbers a,b and set S1=a, S2=b, S3=V-a-b
- Addition
  - V1 + V2 =
  - (S1 + S2 + S3) + (T1 + T2 + T3) =
  - (S1 + T1) + (S2 + T2) + (S3 + T3)
- Multiplication by scalar
  - n*V = n * (S1 + S2 + S3) = n*S1 + n*S2 + n*S3

Additive secret sharing: 3 party computation

- `P1 knows s1; P2 knows s2`
- `(s1+x1)*(s2+x2) =`

  `s1*s2 + x1*s2 + x2*s1 + x1*x2 =`

  `s1*s2 + x1(s2+x2) + x2(s1+x1) - x1*x2`

- `s1*s2 = (s1+x1)(s2+x2) - x1(s2+x2) + -x2(s1+x1) + x1*x2`

1. `P3 generates two random x1,x2`
2. `P3 sends  x1 -> P1, x2 -> P2`
3. `P1 sends (s1+x1) -> P2`   `P2 sends (s2+x2) -> P1`

Additive secret sharing: 3 party computation

- Multiplication
  - V1 * V2 =
  - (S1 + S2 + S3) * (T1 + T2 + T3) =
  - (S1*T1) + (S1*T2) + (S1*T3) + (S2*T1) + (S2*T2) + (S2*T3) + (S3*T1) + (S3*T2) + (S3*T3)

Sharemind

https://sharemind.cyber.ee/sharemind-mpc/

Questions?