HW2

Adam Bowers, Dalton Cole, Sammie Bush

November 29, 2017

PseudoCode

Let l be the bit length of u&v, and let 1 be most significant bit and l least significant bit. Part 1 Party one randomly chooses functionality F=u¿v or u¡v He first computes

- for i=1 to l
  - $E_{pk}(u_i * v_i) \leftarrow SM(E_{pk}(u_i), E_{pk}(v_i))$
    first party one computes the product of current bit of v and u
  - if selected F: was u>v
    $W_i \leftarrow E_{pk}(u_i) * E_{pk}(u_i * v_i)^{n-1}$
    compute $u_i - u_i * v_i$ if there is any $W_i = 1$ we know for i $u_i = 1$ & $v_i = 0$
    $\Gamma_i \leftarrow E_{pk}(v_i - u_i) * E_{pk}(r_i)$ where $r_i \in Z_n$
    compute $v_i - u_i + r_i$
  - else
    $W_i \leftarrow E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-1}$
    compute $v_i - u_i * v_i$ if there is any $W_i = 1$ we know for i $v_i = 1$ & $u_i = 0$
    $\Gamma_i \leftarrow E_{pk}(u_i - v_i) * E_{pk}(r_i)$ where $r_i \in Z_n$
    compute $u_i - v_i + r_i$
  - $G_i \leftarrow E_{pk}(u_i \oplus v_i)$
    now compute xor of two bits if 0 they're the same so first 1 tells us first different bit
  - $H_i \leftarrow H_{i-1}^{s_i} * G_i$; where $s_i \in Z_n$ and $H_0 = E_{pk}(0)$
    now mask the xor based on previous bits and a random number, Hs will be 0 till first 1 then every term past that will be a random value based on s values.
  - $\Phi_i \leftarrow E_{pk}(-1) * H_i$
    shift the domain of by n-1 **Not sure what this is for? to mask output for party two? two make sure 1s are 0 and every other value is a random value?**
  - $L_i \leftarrow W_i * \Phi_i^{t_i}$ where $t_i \in Z_n$
    if W_i is 1 $\Phi_i^{t_i}$ will be 0 thus L_i will be 1 otherwise it will be some random value
- $\Gamma' \leftarrow \pi_1(\Gamma)$
- $L' \leftarrow \pi_2(L)$
  we permute the outputs so P2 can't tell anything from indices of bits
- send Γ'&L' to P2

Part 2
Party two computes

- $M \leftarrow D_{sk}(L'_i)$
  decrypt our Ls

- if there exists an $M_i = 1$

  $\alpha \leftarrow 1$

- else

  $\alpha \leftarrow 0$
  assign alpha based on L values. If there is a 1 we know the selected F function of P1 is true but not what it is

- $M'_i \leftarrow \Gamma_i^{\alpha'} for 1 \leq i \leq l$
  if we have a 1 in M then P1 gets *Gamma* back, otherwise they get random values

- send M' and $E_{pk}(\alpha)$ to P1

Part 3

- $M \leftarrow \pi^{-1}(M')$
  un permute M

- for i= 1 to l

  $\lambda_i \leftarrow M_i * E_{pk}(\alpha)^{N-r_i}$
  if alpha = 1 we subtract the random value added to L in part one
  to get either $v - u$ or $u - v$ depending on f chosen

  if F = $u > v$

  $E_{pk}(min(u,v)_i) \leftarrow E_{pk}(u_i) * \lambda_i$
  if $u > v$ $\lambda = v - u$ if $v > u$ otherwise 0
  therefore $u_i + \lambda_i$ gives us bit of highest value

  else

  $E_{pk}(min(u,v)_i) \leftarrow E_{pk}(v_i) * \lambda_i$
  if $v > u$ $\lambda = u - v$ if $u > v$ otherwise 0
  therefore $v_i + \lambda_i$ gives us bit of highest value

- concat $E_{pk}(min(u,v)_i)$ and Party one has $E_{pk}(min(u,v))$ as required

Two Phase Version

Let l be the bit length of u&v, and let 1 be most significant bit and l least significant bit. Furthermore let BD be a secure Binary Decomposition function & SM be a secure multiplication function

Phase one of the protocol returns E(0) if u is the minimum and E(1) if v is the minimum or equal to u.

(Minimum(E(u),E(v))
where E(u) and E(v) are encrypted bitwise Part 1
Party one randomly chooses functionality F=u¿v or u¡v He first computes

- for i=1 to l

- $E_{pk}(u_i * v_i) \leftarrow SM(E_{pk}(u_i), E_{pk}(v_i))$
  first party one computes the product of current bit of v and u
- if selected F: was u>v

  $W_i \leftarrow E_{pk}(u_i) * E_{pk}(u_i * v_i)^{n-1}$

  compute $u_i - u_i * v_i$ if there is any $W_i = 1$ we know for i $u_i = 1$ & $v_i = 0$
- else

  $W_i \leftarrow E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-1}$

  compute $v_i - u_i * v_i$ if there is any $W_i = 1$ we know for i $v_i = 1$ & $u_i = 0$
- $G_i \leftarrow E_{pk}(u_i \oplus v_i)$
  now compute xor of two bits if 0 they're the same so first 1 tells us first different bit
- $H_i \leftarrow H_{i-1}^{s_i} * G_i$; where $s_i \in Z_n$ and $H_0 = E_{pk}(0)$
  now mask the xor based on previous bits and a random number, Hs will be 0 till first 1 then every term past that will be a random value based on s values.
- $\Phi_i \leftarrow E_{pk}(-1) * H_i$
  shift the domain of by n-1 **Not sure what this is for? to mask output for party two? two make sure 1s are 0 and every other value is a random value?**
- $L_i \leftarrow W_i * \Phi_i^{t_i}$ where $t_i \in Z_n$
  if W_i is 1 $\Phi_i^{t_i}$ will be 0 thus L_i will be 1 otherwise it will be some random value

* $L' \leftarrow \pi_2(L)$
  we permute the output so P2 can't tell anything from indices of bits
* send L' to P2

Part 2
Party two computes

* $M \leftarrow D_{sk}(L'_i)$
  decrypt our Ls
* if there exists an $M_i = 1$

  $\alpha \leftarrow 1$
* else

  $\alpha \leftarrow 0$

  assign alpha based on L values. If there is a 1 we know the selected F function of P1 is true but not what it is
* send $E_{pk}(\alpha)$ to P1

Part 3

if alpha is one function is true 0 false, map that to
0 if u is min otherwise 1

* if F = $u > v$

  $E_{pk}(result) \leftarrow E_{pk}(alpha)$

- else

    $E_{pk}(result) \leftarrow (E_{pk}(alpha) * E_pk(-1))^N - 1$

  if F was $u > v$ alpha already has 0 if u is min 1 if v

  for the other function we have to subtract 1 and to get 0 if u is minimum 1 otherwise

  return $E_pk(result)$

In phase two party one uses minimum function to get min index and returns that value
Return Minimum
P1 computes

- $E_{pk}(u) = BD(E_pk(u))$
- $E_{pk}(v) = BD(E_pk(v))$
- $MinIndice = Minimum(E_{pk}(u), E_{pk}(v))$
- for i=1 to l

    $\Gamma_i \leftarrow E_{pk}(v_i - u_i) * E_{pk}(r_i)$ where $r_i \in Z_n$

  compute $v_i - u_i + r_i$
- $\Gamma' \leftarrow \pi_1(\Gamma)$
- send Γ' and $MinIndice$ to P2

P2 computes

- $MinIndice \leftarrow D_{sk}(MinIndice)$
- $M'_i \leftarrow \Gamma_i^{(MinIndice}$ for $1 \leq i \leq l$
- return M'

P1 computes

- $M \leftarrow \pi^{-1}(M')$
- $\lambda_i \leftarrow M_i * E_{pk}(MinIndice)^{N-r_i}$ for i= 1 to l

  remove random terms from the difference so we can return the minimum
- $\Gamma \leftarrow E_{pk}(u) * \lambda$

  this returns u if u is minimum otherwise v
- $E_{pk}(min(E_{pk}(u), E_{pk}(v))) = \Gamma$

Thus party one has minimum encrypted element between u & v as required

Description

Example