مجله‌ی ریاضی شریف
دوره‌ی سوم، شماره‌ی دوم (بهار ۱۴۰۳)

محاسبات چندجانبه‌ی امن

امیرحسین ندیری و یاشار طالبی‌راد

چکیده. در این مقاله، نخست محاسبات چندجانبه‌ی امن را تعریف می‌کنیم و به بررسی کاربردهای آن می‌پردازیم. سپس، ویژگی‌های ابتدایی پروتکل‌های محاسبات چندجانبه را بیان و مقالات مرتبط و روش‌های موجود را معرفی و مرور خواهیم کرد. در پایان نیز، به تعدادی از کارهای آتی مرتبط با محاسبات چندجانبه‌ی امن اشاره خواهیم کرد.

۱. مقدمه

در محاسبات چندجانبه‌ی امن (MPC)[۱]، تعدادی شرکت‌کننده وجود دارند که هر کدام از آن‌ها دارای یک مقدار مخفی هستند و می‌خواهند به طور مشترک مقدار یک تابع را بر اساس مقادیر ورودی مخفی خود، بدون آشکارسازی آن ورودی‌ها، محاسبه کنند. برای اولین بار، محاسبات چندجانبه‌ی امن در قالب مسأله‌ی میلیونرها[۲] مطرح شد. این مسأله شرایطی را بیان می‌کند که در آن دو فرد ثروتمند بدون افشاکردن میزان دارایی خود و بدون استفاده از شخص سوم معتمد، می‌خواهند بفهمند کدام‌یک ثروتمندتر هستند.

در سال‌های اخیر با وجود پیشرفت‌های روزافزون الگوریتم‌های محاسبات چندجانبه‌ی امن، به ویژه در زمینه‌ی کاهش تأخیر و هزینه‌ی ارتباطات، تعداد بسیار کمی از الگوریتم‌های MPC یافت شده‌اند که در شبکه‌هایی با تعداد شرکت‌کنندگان بسیار زیاد، عمل‌کرد مناسبی دارند. این اتفاق بسیار ناخوشایند است؛ زیرا محاسبات چندجانبه‌ی امن می‌تواند در حل بسیاری از مشکلات موجود در سیستم‌های توزیع‌شده[۳] به‌طور قابل توجهی کمک‌کننده باشد. برای نمونه می‌توان به مسائلی مانند ایجاد الگوریتم‌های یادگیری عمیق بر روی داده‌هایی که در میان خوشه‌های[۴] بزرگ سیستم‌ها قرار دارند، در حالی که صاحبان داده‌ها تمایل ندارند داده‌های خود را به صورت خام به اشتراک بگذارند، یا برگزاری یک حراجی در شبکه‌ی کاملاً توزیع‌شده‌ای مانند بیت‌تورنت[۵] تنها با کمک اعضا[۶] اشاره کرد.

نکته‌ی دیگری که درباره‌ی سیستم‌های توزیع‌شده در مقیاس بزرگ وجود دارد این است که هر کدام از این سیستم‌ها میزان محدودی منابع دارند و وجود یک تعادل در توزیع فشار[۷] شبکه بر روی تمام شرکت‌کنندگان امری بسیار مهم است. هم‌چنین باید توجه داشت که وجود سیستم‌های مخرب[۸] در شبکه‌های بسیار بزرگ توزیع‌شده، به دلیل مکانیزم‌های کنترل پذیرش ضعیف[۹]، یک اتفاق محتمل است.

۲. تعاریف

هدف الگوریتم‌های محاسبات چندجانبه‌ی امن، محاسبه‌ی یک تابع از مقادیر ورودی افراد به صورت امن است؛ بدون آنکه مقدار ورودی هر فرد به هر طریقی برای بقیه مشخص شود. به بیان ساده‌تر شرکت‌کنندگان $P_1, P_2, \ldots, P_n$ و مقادیر $d_1, d_2, \ldots, d_n$ را در نظر بگیرید که هر d_i تنها در اختیار P_i است و هدف محاسبه‌ی $f(d_1, d_2, \ldots, d_n) = (y_1, y_2, \ldots, y_n)$ است.

[1] Secure Multi-Party Computation
[2] Millionaire's Problem
[3] Distributed Systems
[4] Clusters
[5] BitTorrent
[6] Peers
[7] Load
[8] Adversary
[9] Admission Control Mechanisms

است، به طوری که هر y_i خروجی الگوریتم P_i باشد.
دو ویژگی پایه‌ای از این دسته از محاسبات انتظار می‌رود:

(۱) افشا نشدن ورودی خصوصی هر شرکت‌کننده: در این محاسبات هر کدام از شرکت‌کننده‌ها تنها از خروجی دیگران و مقدار ورودی خود آگاه می‌شود. برای مثال در مسأله‌ی میلیونرها که پیشتر بیان شد، هر شخص نباید به میزان ثروت شخص دیگر دست یابد.
(۲) صحت مقدار خروجی محاسبه‌شده توسط تابع: در این محاسبات اگر تعدادی از شرکت‌کنندگان از پروتکل اصلی محاسبات منحرف شوند و رفتار مخرب از خود نشان دهند، نباید بتوانند باعث انحراف دیگر شرکت‌کنندگان درست‌کار به سوی ایجاد خروجی غلط شوند.

با این‌حال، در بعضی مواقع شرط دوم را ضعیف‌تر در نظر می‌گیریم و به جای تضمین پایان‌یافتن محاسبات با خروجی صحیح برای شرکت‌کنندگان درست‌کار، اجازه متوقف شدن در صورت تشخیص خطا را به آن‌ها می‌دهیم.
به عملیات محاسبه‌ی یک تابع n متغیره با یک خروجی به صورت توزیع‌شده، چنان‌که دو شرط بالا را داشته باشد، محاسبه‌ی امن تابع[1] گفته می‌شود که الگوریتم‌های زیادی برای آن وجود دارد. در بعضی از صورت‌بندی‌های مسأله، به جای این که خروجی تابع f به شکل یک بردار n تایی باشد، n تابع مختلف $f_1, f_2, \ldots, f_n$ را در نظر می‌گیرند که خروجی هر کدام از f_i ها یک عدد (همان y_i) است. در این صورت مسأله‌ی MPC به n مسأله‌ی محاسبه‌ی امن تابع تبدیل می‌شود.

با وجود تفاوت‌های بسیار در ویژگی‌ها و نحوه‌ی عمل‌کرد راه‌حل‌های مختلف محاسبات چندجانبه‌ی امن، این سیستم‌ها سه نقش اساسی در بین شرکت‌کنندگان خود دارند که هر شرکت‌کننده می‌تواند یک یا چند نقش داشته باشد:

- شرکت‌کننده‌ی ورودی‌دهنده که اطلاعات محرمانه را به محاسبه‌کنندگان می‌دهد.
- شرکت‌کننده‌ی دریافت‌کننده که نتایج را بصورت پاره‌ای یا کامل از محاسبه‌کنندگان دریافت می‌کند.
- شرکت‌کننده‌ی محاسبه‌کننده که به‌طور مشترک با دیگر محاسبه‌کنندگان، محاسبات را انجام می‌دهد.

دقت کنید راه‌حلی بدیهی که می‌توان برای محاسبه‌ی یک تابع در نظر گرفت این است که هر کدام از n نفر مقدار خود را به یک فرد مورد اعتماد مانند T بدهند و بعد از آن T مقدار تابع را با استفاده از این n ورودی محاسبه کرده و y را به افراد بدهد. یعنی در حقیقت همه‌ی n نفر ورودی‌دهنده و دریافت‌کننده هستند و T محاسبه‌کننده است. از آن‌جا که وجود چنین شخص مورد اعتمادی در عمل امکان‌پذیر نیست، الگوریتم‌های محاسبات چندجانبه‌ی امن به میان آمدند که ما را از وجود شخص مورد اعتماد برای انجام محاسبات بی‌نیاز کنند و از همان n نفر برای دریافت‌کردن و محاسبه‌کردن استفاده کنند.

هر پروتکل MPC با ۴ ویژگی اصلی مشخص می‌شود:

- عملکرد: روش کلی اجرای پروتکل.
- نوع امنیت: نشان‌دهنده‌ی سطح امنیت و میزان اطلاعات بیشتری است که یک مخرب می‌تواند در یک سیستم MPC نسبت به همان پروتکل با استفاده از یک شرکت‌کننده‌ی قابل اعتماد به دست آورد.
- مدل مخرب‌ها: نشان می‌دهد که این پروتکل در برابر چه نوع مخرب‌هایی (مثلاً مخرب فعال[2] یا منفعل[3]، یا تقسیم‌بندی مخرب‌ها بر اساس توان محاسباتی آن‌ها) مقاوم است.
- مدل شبکه: نشان می‌دهد این پروتکل در چه نوع شبکه‌هایی (مثلاً از نظر هماهنگی یا توپولوژی) قابل اجراست.

به عنوان یک مثال ساده می‌توان به مسأله‌ی زیر اشاره کرد:
شبکه‌ای از شرکت‌کنندگان $A_1, A_2, \ldots, A_n$ را در نظر بگیرید که هر کدام یک مقدار مخفی مانند d_i دارند و تمایل ندارند داده‌های خود را به صورت خام به اشتراک بگذارند، اما تصمیم دارند که $\sum_{i=1}^{N} d_i$ که برابر مجموع مقادیر مخفی آن‌ها است را محاسبه کنند.

همان‌طور که گفته شد، یک راه بسیار ساده استفاده از یک شرکت‌کننده‌ی خارجی مورد اعتماد به عنوان محاسبه‌کننده است که هر شرکت‌کننده‌ی دیگر ورودی خود را به او بدهد و او پس از محاسبه‌ی مجموع، مقدار خروجی را به آن‌ها بازگرداند. با

[1] Secure function evaluation
[2] Active Adversary
[3] Passive Adversary

این‌حال روشن است که این روش مبتنی بر وجود یک شخص مورد اعتماد است که هدف اصلی ما در MPC از بین بردن این نیاز است.

به عنوان راه‌حلی مستقل از شرکت‌کننده‌ی قابل اعتماد می‌توان از این روش استفاده کرد: A_1 عدد تصادفی r را انتخاب می‌کند و مقدار $r+d_1$ را به A_2 می‌دهد و در ادامه هر شرکت‌کننده مقدار دریافتی را با مقدار خود جمع می‌کند و به شرکت‌کننده‌ی بعدی ارسال می‌کند تا در انتها مجموع مقادیر و r به شرکت‌کننده‌ی آغازین برسد و شرکت‌کننده‌ی آغازین که از مقدار r آگاه است، با کم‌کردن آن می‌تواند به مقدار مجموع مورد نظر دست یابد.

۳. روش‌ها و الگوریتم‌ها

اولین بار یائو[1] در سال ۱۹۸۲ مسأله‌ی میلیونرها را که سابقاً به آن اشاره شد، در [۱] مطرح کرد. چهار سال بعد خود او در [۲] مستقیماً برای حل مسأله‌ی محاسبه‌ی امن تابع در حالت خاص دو نفره راه‌حلی ارائه داد. یک سال بعد در [۳] الگوریتم حل مسأله‌ی محاسبه‌ی امن تابع برای توابع دلخواه و n متغیره ارائه شد، که به الگوریتم GMW[2] مشهور است. این الگوریتم از پروتکل‌های دانایی صفر[3] استفاده می‌کرد و صحت الگوریتم در حالت اکثریت درست‌کار[4] تضمین شده بود.

یکی از مهم‌ترین قضایایی که شرایطی را درباره‌ی امکان‌پذیری انجام محاسبات چندجانبه‌ی امن مطرح می‌کرد، در سال ۱۹۸۸ در [۴] بیان و اثبات شد. این قضیه بیان می‌کند هر تابع n متغیره را می‌توان با استفاده از n پردازه[5] طوری محاسبه کرد که:

(۱) اگر خطایی[6] رخ ندهد، هیچ زیرمجموعه‌ی کمتر از $\frac{n}{3}$ عضوی از شرکت‌کنندگان اطلاعاتی درباره ورودی شرکت‌کنندگان دیگر دریافت نکنند.

(۲) اگر خطای بیزانتین[7] داشته باشیم، هیچ زیرمجموعه‌ی کمتر از $\frac{n}{3}$ عضوی از شرکت‌کنندگان نتوانند محاسبه‌ی تابع را دچار اشکال کنند و همچنین نتوانند اطلاعاتی درباره ورودی شرکت‌کنندگان دیگر دریافت کنند.

همچنین، دو کران داده شده بهینه هستند. به طور کلی در مسأله‌های امنیتی در سیستم‌های توزیع‌شده و به‌خصوص در MPC، مخرب‌ها را به صورت مجموعه‌ای کنترل شده توسط یک هماهنگ‌کننده مرکزی در نظر می‌گیرند. این مخرب‌ها در دو نوع دسته‌بندی می‌شوند:

(۱) مخرب منفعل: مخرب‌هایی که t نفر را کنترل می‌کنند و می‌توانند شرایط داخلی و مقدار متغیرهای آن‌ها را ببینند. این مخرب‌ها طبق پروتکل عمل می‌کنند و بنابراین اشکالی در اجرای الگوریتم ایجاد نمی‌کنند. در حقیقت تنها نگرانی درباره‌ی مخرب‌های منفعل این است که ورودی بقیه شرکت کنندگان را بفهمند.
شرط اول بیان می‌کند که در حالتی که مخرب‌ها منفعل باشند تعدادشان (یا به طور دقیق‌تر تعداد افراد تحت کنترلشان که همان t است) باید کمتر از $\frac{n}{2}$ نفر باشد تا اطلاعات اضافه‌ای دریافت نکنند. در صورتی که مخرب منفعل با t نفر تحت کنترل خود نتواند ورودی بقیه شرکت‌کنندگان یک محاسبه را به دست آورد، آن محاسبه را t-private می‌نامیم.

(۲) مخرب فعال: مخرب‌هایی که t نفر را کنترل می‌کنند و می‌توانند حالت داخلی آن t نفر را ببینند. این مخرب‌ها لزوما طبق پروتکل عمل نمی‌کنند.
شرط دوم بیان می‌کند که در حالتی که مخرب‌ها فعال باشند تعدادشان باید کمتر از $\frac{n}{3}$ نفر باشد تا اطلاعات اضافه‌ای دریافت نکنند و الگوریتم به درستی اجرا شود. در صورتی که مخرب فعال با t نفر تحت کنترل خود نتواند ورودی بقیه‌ی شرکت‌کنندگان یک محاسبه را به دست آورد، آن محاسبه را t-secure می‌نامیم.

---
[1] Yao
[2] Goldreich, Micali, Wigderson
[3] Zero-Knowledge
[4] Honest Majority
[5] Process
[6] Fault
[7] Byzantine

[۴] همچنین الگوریتمی بیان می‌کند که بتوان یک تابع f را با استفاده از n نفر با شرایط بالا در محیط هماهنگ[۱] محاسبه کرد. در این مقاله گفته‌شده که می‌توانیم فرض کنیم که تابع f چندجمله‌ای است و در نتیجه یک مدار حسابی[۲] برای آن وجود دارد. فهم این الگوریتم و اکثر الگوریتم‌های مقالات مرتبط دیگر نیاز به آشنایی با مفهوم الگوریتم تسهیم راز[۳] دارد که روشی را بیان می‌کند که با استفاده از آن می‌توان یک راز مانند S را طوری بین n نفر به سهم‌های $S_1, S_2, \ldots, S_n$ تقسیم کرد به طوری که:

(۱) دانستن حداقل k مورد از سهم‌ها باعث محاسبه‌ی S می‌شود.

(۲) دانستن $k-1$ مورد از سهم‌ها هیچ اطلاعاتی درباره‌ی S نمی‌دهد.

این الگوریتم را یک تسهیم راز با پارامترهای (k,n) یا تسهیم راز با آستانه‌ی k می‌نامند. شامیر[۴] الگوریتم تسهیم رازی را در [۵] ارائه داده است که به تسهیم راز شامیر[۵] مشهور است.

توجه کنید که الگوریتم‌های تسهیم راز فرض می‌کنند شخصی که می‌خواهد رازش را به اشتراک بگذارد سهم‌ها را به طور صحیح محاسبه و تقسیم می‌کند. با حذف این فرض دسته‌ی دیگری از الگوریتم‌های تسهیم راز را خواهیم داشت که به تصدیق‌پذیر[۶] یا به اختصار VSS مشهور هستند. این الگوریتم‌ها تسهیم راز را حتی با فرض وجود چنین افرادی ممکن می‌سازند. الگوریتم مربوط به قضیه‌ی قبل به طور کلی از ۳ مرحله تشکیل شده است:

(۱) مرحله‌ی ورودی که در آن هر نفر با استفاده از یک الگوریتم تسهیم راز مانند الگوریتم شامیر ورودی خود را به n قسمت تقسیم کرده و به هر نفر سهم مربوط به او را می‌دهد.

(۲) مرحله‌ی محاسبه[۷] که در آن هر نفر مدار محاسباتی f را گیت به گیت شبیه‌سازی کرده و مقدار محاسبه‌شده‌ی هر گیت را به عنوان یک راز مشترک بین افراد نگه‌داری می‌کند.

(۳) در مرحله‌ی نهایی، مقدار تابع f (خروجی مدار) که محاسبه شده به چند سهم تقسیم می‌شود و بین افرادی که قرار است بتوانند مقدار نهایی تابع را به کمک هم محاسبه کنند تقسیم می‌شود.

در [۶] که در همان سال منتشر شد، الگوریتمی بیان شده است که با استفاده از آن، انجام هر پروتکل MPC امکان‌پذیر است، اگر حداقل $\frac{۲n}{۳}$ از اعضا درست‌کار باشند. این مقاله برای اولین بار بدون استفاده از تکنیک‌های رمزنگاری و با استفاده از یک VSS جدید، از اثرگذاری افراد مخرب جلوگیری کرد. این دستاوردی بزرگ بود، چرا که اولین‌بار بود که فهمیدن ورودی‌های بقیه‌ی اعضا از نظر ریاضی غیرممکن بود، درحالی که روش‌ها و تکنیک‌های رمزنگاری حل این مسأله را صرفاً از نظر محاسباتی سخت و غیرممکن می‌ساختند.

الگوریتم معرفی‌شده در [۶] از دو مرحله تشکیل شده است:

(۱) مرحله‌ی تعهد[۸] که در آن از الگوریتم تسهیم راز تصدیق‌پذیر جدیدی استفاده شده که با استفاده از آن، افراد به ورودی خودشان متعهد می‌شوند و راهی برای عوض کردن آن در مراحل بعدی ندارند.

(۲) مرحله‌ی محاسبه که در آن بعد از این که همه به ورودی خود متعهد شدند و هر نفر اطلاعات مورد نیازش را از ورودی‌های افراد دیگر گرفت، هر شخص به صورت محلی[۹] با استفاده از سهم‌های بقیه مقدار تابع را محاسبه می‌کند.

دقت کنید در مرحله‌ی اول چون از الگوریتم تسهیم راز تصدیق‌پذیر استفاده شده، در صورتی که شخصی خرابکار عددی را به اشتباه متعهد شود افراد متوجه می‌شوند و اقدامات لازم را انجام می‌دهند. هر دوی این مقالات وجود کانال‌های امن برای فرستادن یا دریافت اطلاعات بین هر جفت از اعضا را فرض می‌گیرند. این دو مقاله و الگوریتم‌های ارائه‌شده در آن‌ها، مقدمه‌ای شدند برای طراحی الگوریتم‌هایی بر مبنای آن‌ها که وقتی با تکنیک‌هایی برای تسریع انجام اعمال اولیه ترکیب می‌شوند، بهینگی آن‌ها به حدی می‌رسد که در عمل قابل پیاده‌سازی و استفاده باشند. برای مثال در [۱۲] دو کاربرد MPC به طور کامل بررسی شده است که عبارت‌اند از:

---

[1] Synchronous
[2] Arithmetic Circuit
[3] Secret Sharing
[4] Shamir
[5] Shamir's Secret Sharing Scheme
[6] Verifiable
[7] Computation
[8] Commitment
[9] Local

(۱) انواعی از مزایده که در آن‌ها پیشنهاد هر شخص باید به هر دلیل مخفی بماند و برای سایر شرکت‌کنندگان فاش نشود.

(۲) ارزیابی[1] شرکت‌ها که در آن هر شرکت می‌خواهد خودش را طوری با بقیه‌ی شرکت‌ها مقایسه کند که اطلاعات هیچ شرکتی برای شرکت‌های دیگر فاش نشود.

در سال ۱۹۹۰ در [۷] پروتکلی موسوم به BMR[2] بر مبنای الگوریتم GMW و یک VSS جدید ارائه شد که زمان اجرای آن بر حسب عمق مدار تابع f خطی بود. همچنین روش کار الگوریتم به طور کلی مشابه با الگوریتم GMW بود. این الگوریتم در سال ۲۰۰۸ در [۸] با پروتکل دیگری به نام BGW[3] ترکیب شد و بسیار بهینه‌تر شد. تا این زمان همه‌ی الگوریتم‌های ارائه‌شده برای تعداد کمی نفر (n کوچک) عملی بودند و با زیاد شدن n، حجم محاسبات و یا زمان اجرای الگوریتم بسیار زیاد می‌شد، تا جایی که دیگر قابل استفاده نباشد (به بیان دیگر این الگوریتم‌ها مقیاس‌پذیر نبودند). سرانجام در سال ۲۰۰۶، در [۹] اولین الگوریتم مقیاس‌پذیر برای حل حالت کلی MPC ارائه شد. این الگوریتم علیه مخرب‌های فعال امن بود و در محیط هماهنگ اجرا می‌شد.

اشکال اساسی‌ای که به الگوریتم‌هایی که در مقاله‌های قبل بررسی شده بودند وارد است این است که همگی برای محیط‌های هماهنگ هستند. محیط‌های واقعی مانند اینترنت معمولاً ناهماهنگ[4] هستند و این باعث می‌شود الگوریتم‌های گفته‌شده در موارد بسیاری غیر قابل پیاده‌سازی باشند. هم‌چنین اجرای الگوریتم‌های هماهنگ در عمل ممکن است بیشتر از الگوریتم‌های ناهماهنگ طول بکشد؛ زیرا مثلاً در شبکه‌ای که معمولاً سریع است اما گاهی بسیار کند عمل می‌کند، باید هر مرحله‌ی یک الگوریتم هماهنگ به اندازه‌ی بیشینه‌ی زمان رسیدن همه‌ی پیام‌ها طول بکشد، زیرا در غیر این صورت نمی‌توان بین یک شرکت‌کننده‌ی مخرب و کندی شبکه تمایز قائل شد. در صورتی که در یک الگوریتم ناهماهنگ، اعضا به محض دریافت اطلاعات کافی کارشان را ادامه می‌دهند.

در [۱۰] که در سال ۲۰۰۹ منتشر شد، یک پروتکل برای حل حالت کلی MPC در محیط‌های ناهماهنگ با مخرب‌های فعال (کمتر از $\frac{n}{۴}$ نفر) و با فرض وجود کانال‌های امن بین هر دو نفر ارائه شد که با دیگر الگوریتم‌های ناهماهنگ برای حل MPC دو تفاوت اساسی داشت:

(۱) وجود یک نقطه‌ی هماهنگی[5] فرض شده بود که کران بالایی برای زمان رسیدن پیام‌های فرستاده‌شده از طرف افراد درست‌کار در نظر گرفته بود (الگوریتم قبل و بعد از نقطه‌ی هماهنگی کاملاً ناهماهنگ است).

(۲) اتمام[6] الگوریتم فقط به شرط اتمام صحیح مرحله‌ی پیش‌پردازش تضمین شده بود. در حقیقت افراد مخرب می‌توانند باعث اخلال در مرحله‌ی پیش‌پردازش شوند اما اگر این اتفاق رخ ندهد، الگوریتم به اتمام می‌رسد و همه‌ی افراد درست‌کار خروجی را دریافت می‌کنند.

این مقاله سه نکته‌ی قابل توجه داشت که عبارت‌اند از:

(۱) پیچیدگی زمانی و محاسباتی بهترین الگوریتم‌های MPC آن زمان که t-private بودند (جلوی مخرب‌های منفعل را می‌گرفتند.) از $O(n^۲k|C|)$ بود که در آن $|C|$ سایز مدار محاسباتی مورد استفاده برای محاسبه‌ی f و k بیشینه‌ی طول هر ورودی است؛ در حالی که پیچیدگی این الگوریتم که جلوی مخرب‌های فعال را هم می‌گرفت همان $O(n^۲k|C|)$ بود.

(۲) در این مقاله روشی برای اجرای الگوریتم‌های هماهنگ در محیط‌های ناهماهنگ ارائه شده و از آن استفاده شده است.

(۳) نویسندگان برای نشان دادن عملی‌بودن الگوریتم، آن را به طور کامل پیاده‌سازی و ارزیابی کرده‌اند و در قالب نرم‌افزاری در اختیار عموم گذاشته‌اند.

حال الگوریتم را توضیح می‌دهیم. هدف نهایی در این الگوریتم محاسبه‌ی $f(x_۱,\ldots,x_n)=(y_۱,\ldots,y_n)$ است. در مرحله‌ی پیش‌پردازش، r_i عددی تصادفی در نظر گرفته‌شده که کمک می‌کند هر نفر ورودی خود را با آن جمع کند، تسهیم کند و سهم‌ها

[1] Benchmarking
[2] Beaver, Micali, and Rogaway
[3] Ben-Or, Goldwasser, and Wigderson
[4] Asynchronous
[5] Synchronization Point
[6] Termination

را بفرستد. برای شبیه‌سازی یک الگوریتم هماهنگ با R مرحله در محیط ناهماهنگ، الگوریتم زیر برای هر پردازه (مثلاً P_j) پیشنهاد شده است:

(۱) r که شماره‌ی راند است را ۱ قرار بده و برای هر i پیام $m_{j,i,۱}$ را که قرار است در اولین مرحله‌ی الگوریتم هماهنگ به P_i فرستاده شود، محاسبه کن.
(۲) $m_{j,۱}$ را که قرار است به همه فرستاده شود، محاسبه کن.
(۳) برای هر i، همه‌ی $m_{j,i,۱}$ ها و $m_{j,۱}$ را به P_i بفرست.
(۴) تا وقتی $r \leq R$:
۱.۴. صبر کن تا همه‌ی پیام‌های $m_{i,j,r}$ و $m_{i,r}$ از همه‌ی P_i ها دریافت شود.
۲.۴. به کمک پیام‌های دریافتی، پیام‌های خروجی که به شکل $m_{j,i,r+۱}$ و $m_{j,r+۱}$ هستند را محاسبه کن و در نهایت به r یکی اضافه کن.
(۵) متغیر g_j یکی از حالات S یا F را می‌گیرد که به ترتیب بیان‌گر به درستی اجراشدن یا به‌مشکل‌خوردن در مرحله‌ی پیش‌پردازش است. همچنین M_j شامل تمام پیام‌های به شکل $m_{i,r}$ برای $i \in \{۱, \cdots, n\}$ و $r \in \{۱, \cdots, R\}$ می‌باشد. پیام (check, g_j, M_j) را به همه بفرست.
(۶) صبر کن تا همه‌ی $n-۱$ نفر دیگر، پیام‌های (check, g_i, M_i) خود را بفرستند. اگر برای هر i، داشتیم $g_i = S$ و $M_i = M_j$، $s_i = x_i + r_i$ را به همه بفرست.
(۷) صبر کن تا s_j را از همه بگیری و سپس S_j را برابر $(s_۱, \cdots, s_n)$ قرار بده و به همه بفرست.
(۸) اگر همه‌ی $n-۱$ نفر دیگر S_i خودشان را قبل از timeout مشخص شده به تو فرستادند و برای هر i، S_i با S_j برابر بود، q_i را برابر S قرار بده. در غیر این صورت q_i را F قرار بده.
(۹) یک الگوریتم اجماع بیزانتین[1] روی q_i ها انجام بده تا همه روی یک مقدار مشترک q به توافق برسند. چون از اجماع بیزانتین استفاده کردیم، می‌توانیم مطمئن باشیم اگر همه‌ی افراد راستگو یک q_i داشته باشند، مقدار نهایی q هم برابر همان خواهد بود.

حال که مرحله‌ی پیش‌پردازش تمام شده است، وارد مرحله‌ی محاسبه می‌شویم. در این مرحله هر شخص در صورتی که مقدار نهایی q برابر S باشد، سهم‌های x_i را برای هر نفر دیگر از روی s_i و سهم‌های r_i محاسبه می‌کند. سپس الگوریتمی ارائه شده که با استفاده از آن، هر شخص سهم‌های y_i را با استفاده از سهم‌های x_i محاسبه می‌کند. در نهایت، سهم‌های y_i به P_i فرستاده می‌شوند تا وی بتواند y_i نهایی مخصوص خودش را محاسبه کند.

اما هنوز یک مشکل اساسی وجود داشت! الگوریتم ارائه‌شده با این که در محیط ناهماهنگ به خوبی قابل اجرا بود، مقیاس‌پذیر نبود. بدین ترتیب در سال ۲۰۱۷ در [۱۱] چند پروتکل برای حل مسأله‌ی MPC بین تعداد زیادی شرکت‌کننده معرفی شد که در هر دو محیط هماهنگ و ناهماهنگ قابل اجرا بودند. این الگوریتم‌ها هم‌چنین در برابر مخرب‌های فعال امن بودند، اما به این شرط که در محیط‌های هماهنگ کمتر از $\frac{۱}{۳}$ تعداد کل شرکت‌کننده‌ها را مخرب‌ها تشکیل دهند. این عدد برای محیط‌های ناهماهنگ به $\frac{۱}{۸}$ کاهش پیدا کرده است.

۴. پیشنهاد برای کارهای آتی

۱.۴. سیستم‌های داده خصوصی به عنوان سرویس. در گذشته سیستم‌های داده به عنوان سرویس[2] وجود داشتند که تحلیل‌گرانی که برای تحقیقات خود نیاز به داده‌های واقعی داشتند، می‌توانستند داده‌های مورد نیاز خود را از مجموعه‌ای از داده‌های به‌اشتراک‌گذاشته‌شده در آن سیستم‌ها به دست آورند. این سیستم‌ها مشکلاتی برای تحلیل‌گران به همراه داشتند که از جمله‌ی آن‌ها امکان به وجود آمدن مشکلات امنیتی برای داده‌های به‌اشتراک‌گذاشته‌شده با تحلیل‌گران و یا آشکار شدن داده‌های خصوصی افراد و نقض حریم خصوصی افراد به صورت ناخواسته بود. هم‌چنین ایجاد تغییراتی در داده‌ها به منظور حفظ حریم خصوصی صاحبان داده می‌تواند باعث کم‌شدن کیفیت و کاربرد داده شود.

[1] Byzantine Agreement
[2] Data as a Service

در سال‌های اخیر فعالیت‌هایی برای به وجود آمدن سیستم‌های «داده خصوصی به عنوان سرویس» آغاز شده اما آن‌ها در مراحل ابتدایی خود قرار دارند. برای مثال جانا[1] یکی از این سیستم‌ها است که صرفاً پرسمان[2] های ابتدایی بر روی آن پیاده‌سازی شده‌اند و برای کارهای آتی می‌توان امکانات جدیدی از جمله پرسمان‌های پیشرفته‌ی موجود در پایگاه‌داده‌های رابطه‌ای را به این سیستم‌ها افزود.

۲.۴. سیستم‌های تحلیل داده امن. سیستم‌های تحلیل داده امن، سیستم‌هایی مشابه سیستم‌های تجزیه و تحلیل داده‌های کسب و کار فعلی هستند با این تفاوت که داده‌های به صورت خاص رمزگذاری‌شده دریافت می‌کنند و تحلیل را با حفظ حریم خصوصی انجام می‌دهند و نتایج امن را به کاربر باز می‌گردانند و تنها کاربر قادر به بازگردانی داده‌ها و نتایج است. این سیستم‌ها نیز غالباً دارای پرسمان‌های ابتدایی هستند و نیازمند بهبود در توانایی تحلیل داده‌های ورودی هستند.

۳.۴. سیستم‌های معاملاتی جایگزین. افراد زیادی وجود دارند که تمایل دارند انواع معاملات و مبادلات را به‌صورت محرمانه انجام دهند. می‌توان با استفاده از MPC سیستم‌های مبادلاتی امنی برای این افراد ایجاد کرد.

تشکر و قدردانی

در پایان از استاد گران‌قدر دکتر صابر صالح کلیبر، که بدون رهنمون‌های ایشان نگارش این نوشته ناممکن بود، کمال سپاس‌گزاری را داریم.

مراجع

[1] Yao, A. Protocols for secure computations. *FOCS*. **82** pp. 160-164 (1982)

[2] Yao, A. How to generate and exchange secrets. *27th Annual Symposium On Foundations Of Computer Science (sfcs 1986)*. pp. 162-167 (1986)

[3] Goldreich, O., Micali, S. & Wigderson, A. How to play any mental game. *Proceedings Of The Nineteenth Annual ACM Symposium On Theory Of Computing*. pp. 218-229 (1987)

[4] Ben-Or, M., Goldwasser, S. & Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proceedings Of The Twentieth Annual ACM Symposium On Theory Of Computing*. pp. 1-10 (1988)

[5] Shamir, A. How to share a secret. *Communications Of The ACM*. **22**, 612-613 (1979)

[6] Chaum, D., Crépeau, C. & Damgard, I. Multiparty unconditionally secure protocols. *Proceedings Of The Twentieth Annual ACM Symposium On Theory Of Computing*. pp. 11-19 (1988)

[7] Beaver, D., Micali, S. & Rogaway, P. The round complexity of secure protocols. *STOC*. **90** pp. 503-513 (1990)

[8] Ben-David, A., Nisan, N. & Pinkas, B. FairplayMP: a system for secure multi-party computation. *Proceedings Of The 15th ACM Conference On Computer And Communications Security*. pp. 257-266 (2008)

[9] Damgård, I. & Ishai, Y. Scalable secure multiparty computation. *Annual International Cryptology Conference*. pp. 501-520 (2006)

[10] Damgård, I., Geisler, M., Krøigaard, M. & Nielsen, J. Asynchronous multiparty computation: Theory and implementation. *International Workshop On Public Key Cryptography*. pp. 160-179 (2009)

[11] Dani, V., King, V., Movahedi, M., Saia, J. & Zamani, M. Secure multi-party computation in large networks. *Distributed Computing*. **30**, 193-229 (2017)

[12] Bogetoft, P., Christensen, D., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J., Nielsen, J., Nielsen, K., Pagter, J. & Others Secure multiparty computation goes live. *International Conference On Financial Cryptography And Data Security*. pp. 325-343 (2009)

* دانشجوی علوم کامپیوتر، دانشگاه یورک
رایانامه: anadiri@yorku.ca

* دانشجوی علوم کامپیوتر، دانشگاه آلبرتا
رایانامه: talebira@ualberta.ca

[1] Jana
[2] Query