Kungliga Tekniska Högskolan
Examensarbete inom datalogi, grundnivå
DD143X

Handledare: Mikael Goldmann

KTH Computer Science
and Communication

Calcio

Ett protokoll för dold budgivning

David Strömberg
dstromb@kth.se
900319-5055

+46 70 63 33 648

Hagagatan 20
113 48 Stockholm

Simon Hansén
simonhan@kth.se
881025-2455

+46 76 20 00 977

Annedalsvägen 51
168 70 Bromma

2012-05-21

Redogörelse av arbetsfördelning

Samtliga delar av projektet har kontrollerats och diskuterats av både David Strömberg och Simon Hansén. Dock har en person haft huvudansvar för varje del. Vi har använt oss av *Google Docs* vid producerandet av rapporten. Detta gör att vi båda har skapat det mesta innehållet gemensamt. Huvudansvaret för varje del följer av punkterna nedan.

- Sammanfattning och abstract
  Simon Hansén

- Inledning och problemformulering
  David Strömberg och Simon Hansén

- Bakgrund om SMPC
  Simon Hansén

- Bakgrund om fotboll
  David Strömberg

- SMPC – Hur fungerar det?
  David Strömberg

- Shamir's secret sharing
  Simon Hansén

- Implementation av prototyp (skriftlig del)
  David Strömberg och Simon Hansén

- Implementation av prototyp (programmering)
  David Strömberg och Simon Hansén

- Slutsats
  David Strömberg och Simon Hansén

- Animationer
  David Strömberg

Abstract

SMPC (Secure multi-party computation) is a way to perform computations in which each party's input is kept secret and only the final result is public to all parties. This report examines why SMPC is not more widespread than it is today. We also try to apply SMPC to a new area of use which is sealed biddings for co-owned football players in the Italian Serie A. To implement this, we used Sepia which is a Java library that implements SMPC and Shamir's Secret Sharing.

In the end we concluded that the SMPC would work well if there existed a user friendly interface. The options available today are far too complicated for the average user and there is a need for a simpler solution. Although our implementation is very basic, it would work if it was extended with a user-friendly interface. In Serie A, an SMPC application would definitely be useful to restore the trust which was lost during the *Calciopoli* and the other corruption scandals that occurred in italian football. As long as we manage to clarify and explain how new scandals can be prevented by using the SMPC, we concluded that this would be a good area of use.

Sammanfattning

SMPC (Secure multi-party computation) är ett sätt att utföra beräkningar där varje enskild parts indata hemlighålls och endast resultatet blir tillgängligt för alla parter. I denna rapport undersöks varför SMPC inte är mer utbrett än det är idag. Vi försöker även att applicera det på ett nytt användningsområde vilket är dolda budgivningar för delägda fotbollsspelare i italienska Serie A. För att implementera detta använder vi oss av *Sepia* som är ett javabibliotek för att utveckla SMPC. Sepia använder *Shamir's Secret Sharing* vilket är en algoritm för delning av en hemlighet.

I slutändan kom vi fram till att SMPC skulle fungera bra om det fanns ett enklare gränssnitt för användarna. Det som finns på marknaden i dagsläget är allt för avancerat för den genomsnittliga användaren och det finns absolut ett behov av en enklare lösning. Även om vår implementation är väldigt grundläggande skulle den i praktiken fungera om den skulle utökas med ett användarvänligt gränssnitt. I Serie A skulle det absolut behövas för att återskapa det förtroende som förlorats vid *Calciopoli* och övriga korruptionsskandaler som förekommit inom fotbollen. Så länge det går att tydliggöra och förklara hur nya skandaler skulle kunna förebyggas med hjälp av SMPC kom vi fram till att detta skulle vara ett bra användningsområde.

Innehållsförteckning

Inledning

Secure multi-party computation (SMPC) är en del av kryptografin med syfte att utföra beräkningar över indata från olika användare samtidigt som varje användares indata hålls hemlig för övriga användare. Antalet användare och indata per användare kan variera.[1]

SMPC introducerades 1982 av Andrew Yao. För att på ett enkelt sätt kunna presentera detta skapade han miljonärsproblemet.[2] Yao tänkte sig två miljonärer som pratade med varandra, till slut så börjar de dividera om vem av dem som var rikast. Problemet uppstår när ingen av dem vill berätta sin förmögenhet för den andre. För att ta reda på vem som egentligen är rikast måste de skapa en funktion som kan avgöra om a > b där a och b är de två förmögenheterna. Vidare måste funktionen också vara säker, så att de i efterhand inte kan lista ut hur mycket den andre tjänar. Detta är något som kan lösas med SMPC.

Problemformulering

Syftet med rapporten är att undersöka om det finns något specifikt användningsområde för SMPC som idag är okänt eller där det inte är applicerat. Området som valts är dolda budgivningar för delägda fotbollsspelare i italienska Serie A, där endast det vinnande budet ska vara offentligt. Målet i rapporten är att undersöka följande:

- Skulle SMPC kunna förändra eller förbättra situationen jämfört med hur det ser ut idag?
- Finns det någon anledning till att SMPC inte är mer utbrett än det är idag?
- Hur skulle en implementation av SMPC kunna utformas för dolda budgivningar med avseende på säkerhet och manipulation av bud?

Bakgrund

Hittills har det varit svårt att hitta användningsområden för SMPC, tekniken har funnits länge men antalet implementationer är få enligt vad vi erfar. Vi har valt att ta upp de implementationer av SMPC som vi anser varit viktigast för utvecklingen. Vi har även valt att ta med bakgrundsfakta till hur fotbollens transfersystem är uppbyggt samt lite historia om förtroendet för fotbollen. För mer information om vilka regler som finns för övergångar inom fotboll hänvisar vi till FIFAs hemsida[3].

Secure Multiparty Computation Goes Live

Ett av de största SMPC-projekten genomfördes år 2008 i Danmark, sponsrat av det danska strategiska forskningsinstitutet. Projektet hade namnet *Secure Multiparty Computation Goes Live*. Det hade länge funnits problem med förhandlingar mellan de bönder som producerade sockerbetor och *Danica* som är den ledande inköparen av dessa. Principen var den att varje enskild bonde hade egna förhandlingar med Danica angående vilka säljrättigheter gällande pris och mängd som skulle finnas. Detta system gynnade Danica då de hade en ledande marknadsställning och själva fick all information från bönderna. Detta var inte bönderna alltför förtjusta i då Danica kunde få insikt i deras ekonomi och produktivitet genom att analysera buden. Ett system som självfallet missgynnade bönderna. Därav tyckte forskare att detta var ett utmärkt system att implementera SMPC på då all information om varje enskild part då kunde hållas hemlig för de andra. Efter att all information om buden hade erhållits kunde ett resultat på vilken mängd och till vilket pris varje bonde skulle sälja betorna för beräknas.[4]

Net voting

Ett av de mest intressanta implementationsområdena av SMPC är net voting vilket går ut på att underlätta röstningsprocedurer genom att göra det möjligt att rösta på nätet. En del forskning läggs ner på att försöka förbättra dagens röstningssystem och att göra en säker implementation av detta.[5] Eftersom varje persons röster måste hållas hemliga för att bevara integriteten och det endast är resultatet av alla röster som är viktigt har detta blivit ett intressant forskningsområde för SMPC. För närvarande drivs ett forskningsprojekt på Kungliga Tekniska Högskolan (KTH) vid namn Verificatum som behandlar detta. Verificatum använder sig av Secure Mix-Net[6] där varje användare krypterar sin röst. Sedan skickas den krypterade rösten till en betrodd tredje part som utför beräkningen och sedan får ett resultat av röstfördelningen som utdata.[7]

Det måste även finnas någon form av verifiering för att avgöra att den som röstar är berättigad till detta. I ett perfekt system går det alltså inte att på något sätt kunna spåra en enskild användares röst eller på något sätt kunna manipulera utdata. En utomstående part ska även i efterhand kunna verifiera att resultatet är korrekt.[8]

Transfersystemet inom fotbollen

Reglerna för fotbollsspelares kontrakt och övergångar stadgas av det internationella fotbollsorganet *Fédération Internationale de Football Association* (FIFA).[9] De nationella förbunden lyder under FIFA men kan även ha ytterligare tillägg till FIFAs regler och rekommendationer.
Alla spelare som har ett kontrakt med en fotbollsklubb (A) kan endast bryta sitt kontrakt eller flytta till en annan klubb (B) om en överenskommelse sker mellan spelaren och klubbarna. För att en övergång skall kunna ske kräver oftast A att B betalar en övergångssumma som kompensation. Om de två klubbarna kommer överens med varandra respektive spelaren kan en försäljning till B ske. Övergångar kan endast ske under ett transferfönster, som är en eller flera perioder under året där det nationella fotbollsförbundet bestämt att det är tillåtet att köpa och sälja spelare. Om den köpande klubben spelar i en annan fotbollsliga än klubben som säljer är det den köpande klubbens transferfönster som måste vara öppet.[10]

En spelare kan även lånas ut från A till B under en säsong, detta kan vara gratis eller så kan en kompensation ges till A beroende på lånets syfte. Ofta lånas unga spelare från större klubbar till en klubb i en lägre division med syfte att ge spelare mer speltid och därmed öka spelarens chanser att slå igenom. Lån mellan klubbar i samma liga kan endast ske under ett transferfönster men en utlåning till klubbar i lägre divisioner kan ibland ske mitt under säsongen beroende på hur den nationella ligans regler ser ut.

I vissa fotbollsligor, bland annat i Italien, finns det ett system för delägande av fotbollsspelare mellan två klubbar. Under ett transferfönster kan en klubb välja att sälja en andel av en spelare till en annan klubb. Syftet med detta är att fördela den ekonomiska risken; om spelaren inte når upp till förväntningarna förlorar det säljande laget inte lika mycket pengar. Om spelaren däremot överträffar förväntningarna gör den köpande klubben troligtvis en vinst vid en senare försäljning. Inför varje ny säsong måste de två delägande klubbarna komma överens om var han ska spela då han endast kan representera en klubb i taget.[11] Spelaren kan även lånas ut till en tredje klubb. Om en överenskommelse inte kan ske mellan klubbarna sker en dold budgivning där klubbarna får lägga varsitt bud på spelaren och lämna in dessa bud till en domare som sedan avslöjar vem som gav högst bud. Vinnaren av budgivningen äger sedan spelaren till fullo.

Förtroendet för fotbollen

Genom fotbollens historia har det förekommit ett flertal skandaler. Allt ifrån riggade matcher och mutade spelare till dopning. Detta är något som har förekommit i länder över hela världen ända sedan fotbollens födelse. Till exempel blev Torino i italienska Serie A av med sin ligatitel 1927 på grund av en utredning som visade att mutor till motståndarspelarna utbetalats av Torino. 1980 degraderades Lazio och A.C. Milan, även de i Serie A, för att ha riggat matcher på förhand. Detta är som nämnt tidigare absolut inte är unikt för Italien utan har förekommit i bland annat Tyskland, Belgien, Tjeckien och Grekland. [12]

Under våren 2006 uppdagades en av fotbollshistoriens största skandaler i Italien, *Calciopoli*. Skandalen inkluderade bland annat uppgjorda matcher i Serie A och Serie B. Skandalen uppdagades via

avlyssnade telefonsamtal där det framgick att högt uppsatta personer inom ett flertal fotbollsklubbar hade haft kontakt med domarbasen.[13] Mellan dessa kontakter utbyttes sportbilar mot fördelaktiga domare i nyckelmatcher. Under samma period förekom det även situationer där klubbar förde olagliga samtal med spelare som stod under kontrakt med andra klubbar.[14] Dessa samtal handlade bland annat om hur spelare skulle göra sig omöjliga i sin klubb för att den andra klubben skulle kunna köpa den spelaren för ett billigare pris.[15]

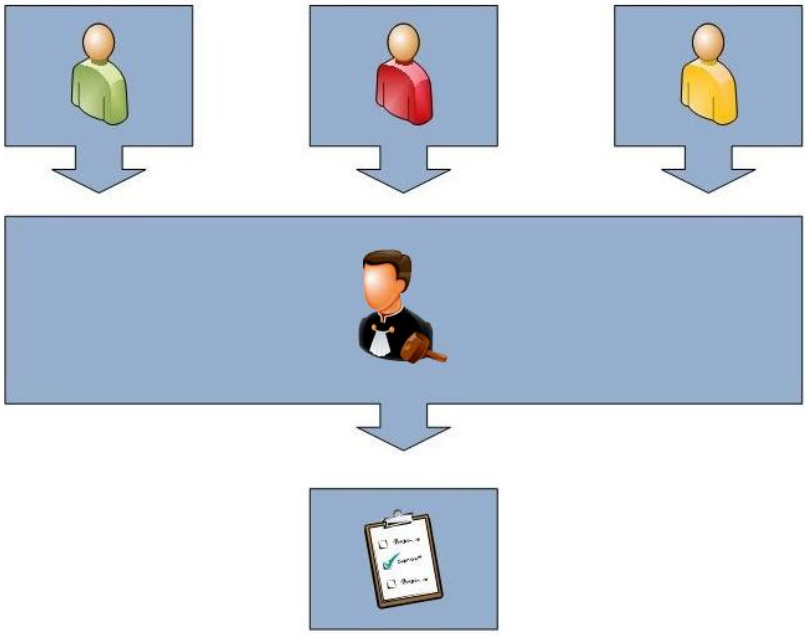
Än idag är Calciopoli högaktuellt i Italien då några av de inblandade klubbarna överklagat påföljderna som de anser varit orättvisa då andra klubbar med liknade affärer fått betydligt lindrigare straff. Detta till stor del på grund av att avlyssningsband försvunnit. Utöver fotbollsskandalen från 2006 har det under 2011 uppkommit nya anklagelser i Italien om spelare som medvetet gjort självmål efter att i samförstånd med maffian spelat på att deras egna lag skulle förlora matchen.[16, 17] Idag menar många att fotbollen håller på att tappa sin trovärdighet. En av dem är den gamle storspelaren Marco Tardelli som bland annat säger följande om fotbollens trovärdighet *"Problemet är att fotbollen inte längre uppfattas som ärlig. Ingen i italiensk fotboll vill ta itu med frågan på allvar. Även domarna har förlorat trovärdighet på grund av deras inblandning i skandalerna"*[18]. Tardelli är inte ensam om sina åsikter, även Gianfranco Zola instämmer: *"Allvarligt talat skadade skandalen den italienska fotbollen oerhört. Vi har förlorat vår trovärdighet och vårt rykte har skadats"*[19].

Secure multi-party computation

Det finns idag flera bibliotek och protokoll för SMPC och därmed ett flertal olika lösningar. Den mest förekommande strukturen är att bygga ett decentraliserat nätverk. Detta för att se till att det inte finns någon part som har tillgång till all information om indata samtidigt.

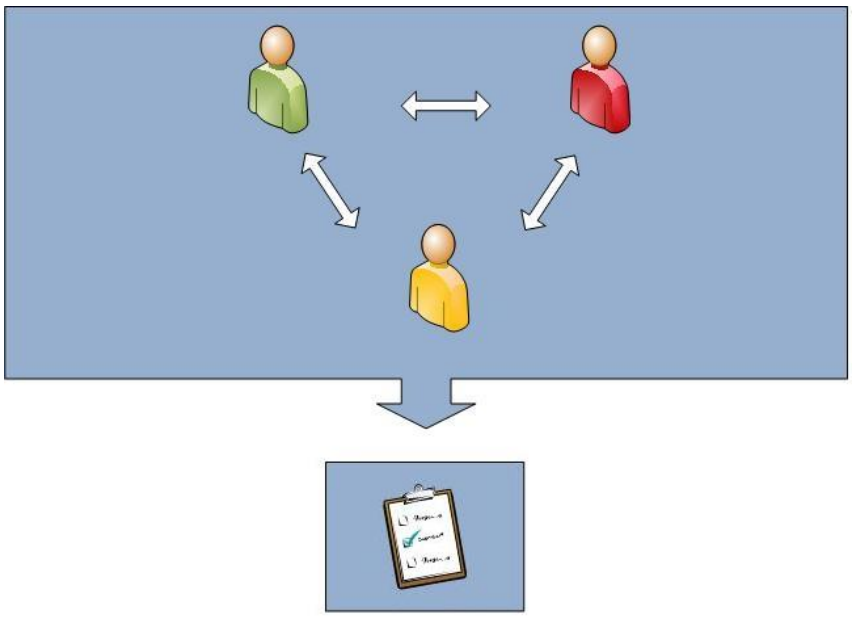
Idag agerar domaren på samma sätt som en central server som tillhandahåller båda buden samtidigt. Detta gör att det är enkelt att manipulera buden eftersom det lägsta budet hålls hemligt. Det är då enkelt för en mutad domare att helt enkelt lägga den ena klubbens bud en bit över den andras och på så sätt garantera den klubben en vinst i budgivningen samt att ett överpris inte läggs. Detta illustreras av figuren nedan, i en budgivning med tre klubbar.[20]

Figur 1 - Dagens system för dolda budgivningar



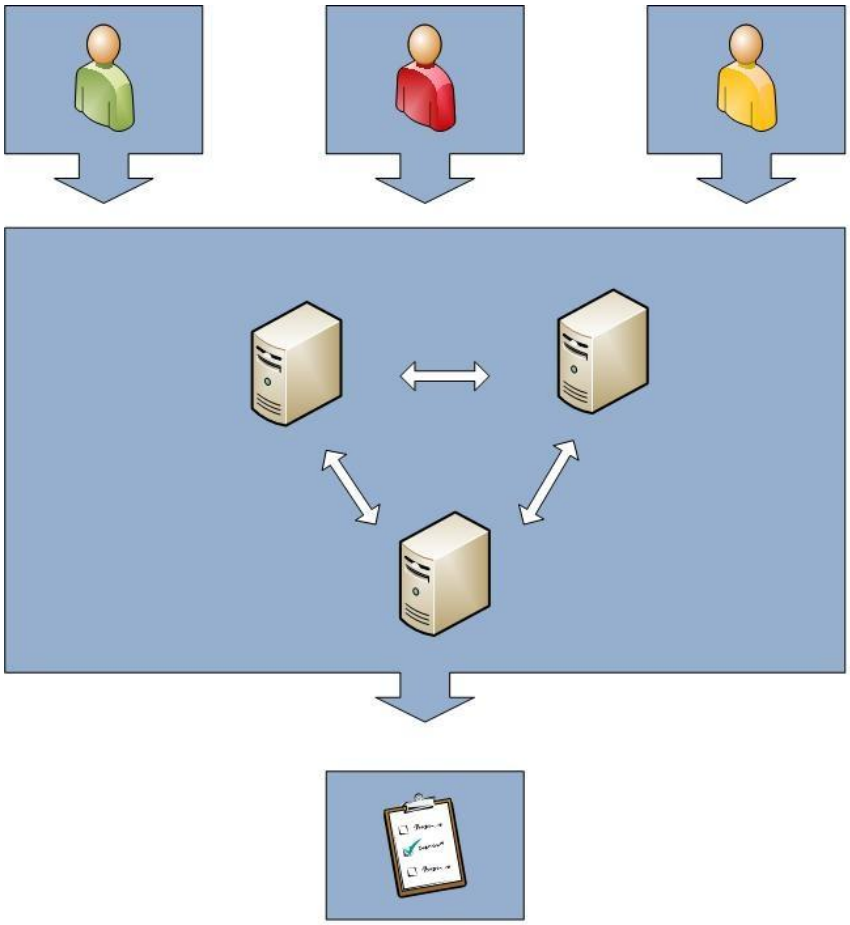
Genom att varje *användare (eng. peer)* manipulerar sin indata och skickar fragment av dem till de övriga deltagarna blir det omöjligt att veta vem som har vilken indata. Fragment av indatan skickas sedan mellan användarna. När alla beräkningar gjorts skickas svaret tillbaka till varje användare. Figuren nedan beskriver ett SMPC-system med tre användare.

Figur 2 - Exempel på ett SMPC-system



För att öka säkerheten kan även ett nätverk där alla deltagande klienter skapar en ny så kallad *privat användare (eng: privacy peer)* implementeras. Dessa privata användare får varsin unik del av varje användares data, på så sätt har ingen all information samtidigt. Sedan utförs beräkningarna på dessa datafragment bitvis och skickas vidare. När beräkningarna är klara skickas de tillbaka till användarna. Dessa privata användare måste vara minst lika många som antalet användare, vanligast är att varje användare skapar en privat användare på sin egen dator. Genom att öka antalet privata användare ökar säkerheten, till exempel kan en privat användare tilldelas en tredje part som alla deltagare litar på för att öka säkerheten.

Figur 3 - Exempel på ett SMPC-system med tre privata användare



Vanligast baseras ett SMPC-bibliotek på antingen *Shamir's secret sharing* (SSS) eller *Yao's Garbled Circuits*, där SSS är något mer förekommande. Vi har valt att lägga fokus på SSS eftersom det ger en enklare bild av hur beräkningar utförs samt att det är mer förekommande bland de olika biblioteken för SMPC.[21] SMPC-bibliotek använder SSS för att dela upp data till de olika parterna när beräkningarna skall göras. Detta för att säkerställa att all indata hålls hemlig.

Shamir's secret sharing

SSS är en algoritm tillhörande området hemlig delning inom kryptografin. En hemlighet S kan krypteras och genererar då n stycken unika nycklar $(s_1, s_2, \ldots, s_n)$ tillhörande m antal personer, där $m \leq n$. Det är alltså fullt möjligt att en person har fler än en nyckel om en hierarkisk struktur ska tillämpas på nyckelfördelningen. S representeras som ett tal i den ändliga kroppen Z_p där p är ett primtal samt att $p > S$. En ensam nyckel säger egentligen ingenting om S utan för att kunna återskapa S behövs minst k nycklar, k-1 nycklar genererar ingen nyttig information om S. Genom att sätta $n = 2k$ -1 krävs det $\frac{n+1}{2}$ nycklar för att komma åt hemligheten S.[22] För att utföra en sådan kryptering krävs ett unikt polynom $f(x)$, av grad n-1. Koefficienterna $(a_1,a_2,\ldots,a_{n-1})$ i $f(x)$ väljs slumpmässigt inom Z_p, a_0 sätts till S. Detta ger följande:

$$f(x) = S + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$$

Nästa steg blir att definiera nycklarna till S $(s_1,s_2, \ldots,s_n)$ som ges av $\{1,f(1); 2,f(2); \ldots; n,f(n)\}$. Vi får då n antal punkter på $f(x)$, där varje punkt är en nyckel. Om polynomets grad skulle väljas som mindre än k-1, resulterar det i att vi får en nyckelfördelning där $k < n - 1$.[23] För att sedan återskapa hemligheten används *Lagranges Interpolation* över Z_p, punkterna som behövs är då till exempel $\{(x_1,y_1),(x_2,y_3)\ldots(x_k,y_k)\}$, men det går självklart bra med vilka k punkter som helst. Det är alltså enkelt att interpolera ett polynom av grad k-1 om det finns k antal punkter. Färre än k punkter gör att sannolikheten för alla värden är lika stora, och därför är det omöjligt att lösa ut *S*. Fördelen med att använda denna typ av system gör det väldigt säkert mot inkräktare eftersom en majoritet av användarna behövs för att kunna återskapa hemligheten.

Om det t.ex. finns fem personer innebär det att ifall någons punkt skulle läcka ut skulle det inte vara tillräckligt mycket information för att finna hemligheten. De fyra kvarvarande personerna kan dock fortfarande lösa ut hemligheten. En fördel med SSS är flexibiliteten eftersom det går att med jämna mellanrum byta ut nycklar genom att ändra koefficienterna framför termerna i polynomet. Dock kan de gamla nycklarna fortfarande användas, vilket ur ett säkerhetsperspektiv kan vara negativt.[24]

Lagranges interpolation

$$L(x) = \sum_{j=1}^{k+1} y_j l_j(x)$$

$$l_j(x) = \prod_{\substack{1 \leq m \leq k+1 \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \left(\frac{x - x_1}{x_j - x_1}\right) \ldots \left(\frac{x - x_{j-1}}{x_j - x_{j-1}}\right) \cdot \left(\frac{x - x_{j+1}}{x_j - x_{j+1}}\right) \ldots \left(\frac{x - x_{k+1}}{x_j - x_{k+1}}\right)$$

Exempel: Bankfacket

Ett exempel kan vara att vi har en styrelse med 5 personer som ska ha tillgång till ett bankfack. För att kunna få tillgång till bankfacket behövs en majoritet av nycklarna. Låt oss säga att hemligheten S är 11. Därefter väljs ett primtal som är större än 11, exempelvis talet 17, vilket gör att alla beräkningar utförs i är Z_{17}. Eftersom vi vill att majoriteten av personerna ska behövas för att ta reda på hemligheten är alltså $n = 2k\text{-}1$. Då $n = 5$ ger detta att $k = 3$. Eftersom det ska konstrueras ett $(k\text{-}1)$-gradspolynom ger detta ett andragradspolynom i Z_{17}.

$$f(x) = S + a_1 x + a_2 x^2$$

Nästa steg blir att definiera koefficienterna till $f(x)$. a_1 och a_2 ska väljas slumpmässigt i Z_{17}. I detta fall valdes a_1 och a_2 till 3 respektive 7.[25] Med dessa koefficienter får vi följande andragradspolynom:

$$f(x) = 11 + 3x + 7x^2$$

När vi sedan vill konstruera varje persons nyckel $(p_1, p_2, \ldots, p_n)$ sätter vi $(x_1, x_2, \ldots, x_n)$ till $(1,2, \ldots,n)$ och skapar sedan alla nycklar med hjälp av $f(x)$ likt figuren nedan.

$$P_x = \big(x, f(x)\big) \equiv_p (x, 11 + 3x + 7x^2) \equiv_p (x, y)$$

$$P_1 = (1, f(1)) \equiv_{17} (1, 11 + 3 + 7) \equiv_{17} (1,4)$$
$$P_2 = (2, f(2)) \equiv_{17} (2, 11 + 3 \cdot 2 + 7 \cdot 4) \equiv_{17} (2,11)$$
$$P_3 = (3, f(3)) \equiv_{17} (3, 11 + 3 \cdot 3 + 7 \cdot 9) \equiv_{17} (3,15)$$
$$P_4 = (4, f(4)) \equiv_{17} (4, 11 + 3 \cdot 4 + 7 \cdot 16) \equiv_{17} (4,16)$$
$$P_5 = (5, f(5)) \equiv_{17} (5, 11 + 3 \cdot 5 + 7 \cdot 25) \equiv_{17} (5,10)$$

Om vi vill återskapa hemligheten S behöver vi tillgång till minst tre personers nycklar. Det spelar ingen roll vilka nycklar vi använder. Låt oss säga att vi har möjlighet att utnyttja nycklarna $p_1=(1,4)$, $p_2=(2,11)$ och $p_3=(3,15)$. Vi använder Lagranges Interpolationsformel där vi först löser ut $l_j(x)$ för de olika nycklarna (p_1, p_2, p_3), därefter sätts polynomen samman enligt figuren nedan.

$$l_j(1) = \frac{x-x_2}{x_1-x_2}\cdot\frac{x-x_3}{x_1-x_3} \equiv_{17} \frac{x-2}{1-2}\cdot\frac{x-3}{1-3} \equiv_{17} \frac{x^2}{2} - \frac{5x}{2} + 3$$

$$l_j(2) = \frac{x-x_1}{x_2-x_1}\cdot\frac{x-x_3}{x_2-x_3} \equiv_{17} \frac{x-1}{2-1}\cdot\frac{x-3}{2-3} \equiv_{17} x^2 + 4x - 3$$

$$l_j(3) = \frac{x-x_1}{x_3-x_1}\cdot\frac{x-x_2}{x_3-x_2} \equiv_{17} \frac{x-1}{3-1}\cdot\frac{x-2}{3-2} \equiv_{17} \frac{x^2}{2} - \frac{3x}{2} + 1$$

$$l_j(1)\cdot y_1 + l_j(2)\cdot y_2 + l_j(3)\cdot y_3 \equiv_{17} \frac{-3x^2}{2} + \frac{23x}{2} - 6 \equiv_{17} 7x^2 + 3x + 11 \rightarrow S = 11$$

Implementation av prototyp

Genom att använda ett befintligt bibliotek för SMPC förenklas processen med att ta fram en prototyp för de dolda budgivningarna. I implementationen kommer alla bud vara hemliga utom det vinnande. I början av detta projekt satte vi upp följande arbetsordning:

- Hitta bra bibliotek för SMPC där den grundläggande kommunikationen finns färdig.
- Undersök dokumentation och kod. Är de tydliga? Om inte, sortera bort.
- Ta reda på vilka matematiska funktioner som finns inbyggt. Finns det något som vi kan använda oss av? Om inte, går det att modifiera någon av de befintliga funktionerna?
- Konstruera ett eget protokoll och konfigurera det efter vårt syfte, det vill säga antalet tal att beräkna, antalet användare och så vidare.

Bibliotek

De bibliotek som undersöktes var *Virtual Ideal Functionallity Framework* (VIFF)[26], *Security through Private Information Aggregation* (Sepia)[27], *Sharemind*[28]och *Fair Play*[29]. VIFF och Sepia valdes ut ifrån mängden eftersom de i ett tidigt stadium verkade passa våra ändamål samt att det fanns relativt mycket dokumentation tillgängligt. Statistik visade även att Sepia skulle vara betydligt snabbare att använda än de övriga biblioteken.[30]Sepia och VIFF bygger båda på SSS[31] medan Fair Play och Sharemind bygger på Yao's Garbled Circuits. [32, 33]

VIFF

VIFF är ett danskt SMPC-projekt vid Århus Universitet och är skrivet i Python. Tidigt framkom det problem med installationen på grund av att installationsguiden inte var särskilt utförlig. Det fanns stora problem med att få själva strukturen att fungera. T.ex. för att få ett fungerande bibliotek var vi tvungna att installera ett flertal olika delar. Det systemet med minst antal komponenter att installera det på var Ubuntu, där det behövdes fyra stycken. I både Windows och Mac OS X behövdes fem komponenter.[34]
De grundläggande delarna som behövdes för att VIFF skulle fungera var följande:

- Python - Programmeringsspråk för att skriva VIFF
- Twisted - Ett bibliotek till Python för nätverkskommunikation
- OpenSSL – SSL-bibliotek för säker kommunikation
- PyOpenSSL – Ett Pythontillägg för SSL
- GMPY - Ett bibliotek till Python för snabb aritmetik av stora tal

Ett stort problem med att få allting att fungera var att den senaste versionen av VIFF släpptes i december 2009, vilket gjorde den beroende av gamla versioner av komponenterna listade ovanför. Det fanns inget som helst stöd för de versioner av t.ex. Python, Twisted och GMPY som nu finns tillgängliga. Det var stora svårigheter att få tag på de versioner som VIFF hade stöd för då de inte själva tillhandahöll dessa. Svårigheterna med installationen gjorde att vi efter två dagars försök till slut avslutade våra planer med att få VIFF att fungera.

Sepia

Sepia är ett Javabibliotek utformat vid *Eidgenössische Technische Hochschule* (ETH) i Zürich. Inbyggt i Sepia finns det ett flertal olika paket, t.ex *Primitives* som innehåller ett flertal matematiska metoder. Beräkning av minimum och multiplikation är några av de inkluderade metoderna. Det finns även paket som hanterar säker kommunikation mellan användarna.[35] Sepia använder sig av ett system med användare (Peer) och privata användare (Privacy Peer) likt det som är beskrivet i kapitlet *Secure multi-party computation.* Sepia använder sig inte heller av någon central server, utan all kommunikation sker direkt mellan de privata användarna. Varje privat användare kommunicerar sedan med användarna.[36]

Inkluderat i Sepia finns följande protokoll:

- *Addition* - För addition av heltal där den totala summan av alla användares indata ges
- *Distinct Count* - Totala antalet förekomster av användarnas indata
- *Entropy* - Beräknar sannolikhetsdistributionen av någon händelse (Tsallis distribution[37])
- *Event Correlation* - Rapporterar förutbestämda händelser som förekommer i en vald frekvens.

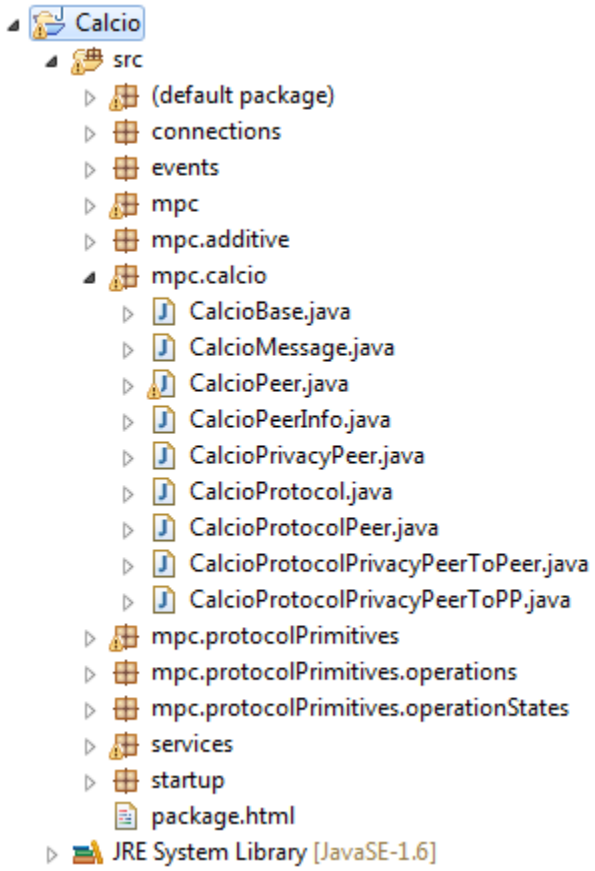
Calcio

För att kunna använda Sepias bibliotek till dolda budgivningar behövdes ett speciellt protokoll. Vi döpte vårt protokoll till *Calcio*. Namnet Calcio kommer från italienskan och betyder fotboll, ordet används ofta för att beteckna det italienska seriesystemet. När vi skapade protokollet användes ett skelett från Sepia där basala delar så som kommunikation mellan användarna fanns inbyggt. Vi har även utnyttjat några av de paket som finns i Sepiabiblioteket. I denna prototyp kommer samtliga användare och privata användare köras på samma dator där de kommunicerar via localhost. I ett verkligt scenario hade varje användare haft tillgång till en egen dator.

Steg 1 - Installation och importering av bibliotek

Eftersom det inte är möjligt att kompilera Javafiler som innehåller kompileringsfel var vi tvungna att importera Sepias huvuddelar till Eclipse. Vårt protokoll Calcio ligger i paketet `sepia/mpc/calcio` och innehåller nio klasser, precis som det givna skelettet. Resterande paket var givna av Sepia. Vi utgick från de nio skelettklasserna som vi sedan byggde vidare på för att bygga vårt egna protokoll.

Figur 4 - Calcios filstruktur i Eclipse



Calcioprotokollet består av följande Javaklasser:

- CalcioBase.java — Innehåller metoder som används av både användare och privata användare.
- CalcioMessage.java — Håller data som ska skickas mellan användare och privata användare.
- CalcioPeer.java — Hanterar tillståndet för en användare och startar protokollet med privata användare.
- CalcioPeerInfo.java — Lagrar information om privata användare.

- CalcioPrivacyPeer.java — Hanterar tillståndet för en privat användare och startar protokollet med användare och övriga privata användare.
- CalcioProtocolPrivacyPeerToPeer.java — Hanterar kommunikationen från en privat användare till en användare.
- CalcioProtocolPrivacyPeerToPP.java — Hanterar kommunikationen mellan två olika privata användare.

Steg 2 - Programmering

Calcio kräver att varje budgivare/användare har en textfil, *bid.txt*, där budet ligger sparat lokalt. Om det skulle visa sig att det inte finns någon fil eller att datan i filen på något sätt är felaktigt, skickas budet 0 från just den budgivaren. Koden nedan beskriver hur läsning och tolkning av indata sker. Variabeln *numberOfItems* sätts vid konfigurationen av programmet och innehåller ett tal som anger hur många tal en användare skall tillåtas ha i sin indatafil. I vårt fall kommer `numberofItems = 1` då varje användare endast kommer att kunna lägga ett bud. Detta reglerar som sagt inte i koden, utan görs i konfigurationen. I detta avsnitt är endast nyckeldelar av Calcio medtaget för att underlätta läsningen. Hela källkoden finns att hämta på `http://www.csc.kth.se/~dstromb/mpc`.

CalcioPeer.java, readDataFromFile()

```
public boolean readDataFromFile(String inputFolderName) {
      Integer bid;
      try {
            FileInputStream fis = new FileInputStream("bid.txt");
            DataInputStream dis = new DataInputStream(fis);
            BufferedReader br = new BufferedReader(new
InputStreamReader(dis));
      } catch (Exception e) {
            bid = 0;
      }

      inputData = new long[numberOfItems];
      for(int inputIndex = 0; inputIndex < numberOfItems; inputIndex++) {
            bid = Integer.valueOf(br.readLine());
            inputData[inputIndex] = Integer.MAX_VALUE-(bid);
      }
      return true;
}
```

Eftersom den inbyggda klassen Primitives endast har en inbyggd metod för att generera ett minimumvärde av två eller flera tal måste vi manipulera indata och utdata. Detta sker genom att använda negation samt ta differensen med det maximala värdet för en integer. Nedan visas hur manipulationen av utdata sker samt hur vi använder klassen Primitives.

CalcioPeer.java, writeOuputToFile() (utdrag)

```
Integer tmp = Integer.valueOf(line.substring(0, line.length() - 2));
tmp = (Integer.MAX_VALUE-tmp);
String result = tmp.toString();
Services.writeFile(result, fileName);
```

CalcioPrivacyPeer.java, startComputation() (utdrag)

```
primitives.min(operationIndex, data, -1, true);
```

Funktionen `min(id, data, knowledge, fewRounds)` i klassen Primitives har följande parametetrar och returvärden:

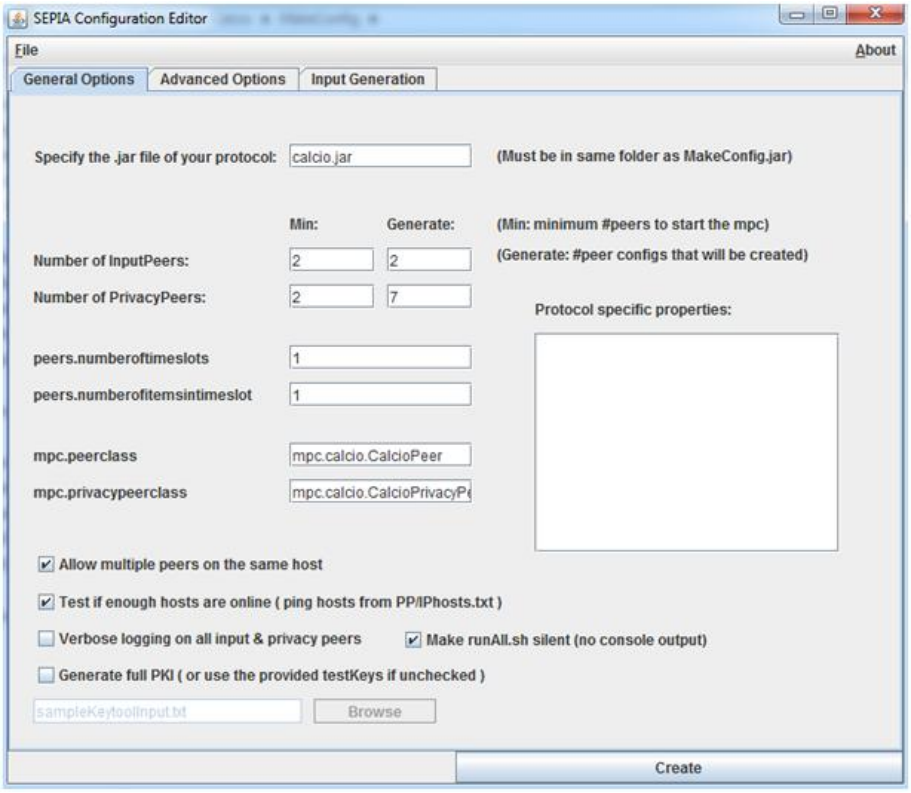
`@param`	`id`	Operationens id.
`@param`	`data`	De fragment av data som beräknas vid anropet
`@param`	`knowledge`	Ytterligare information om data. *knowledge* skall vara 1 om $data[i] \leq fieldSize/2$, 0 om $data[i] > fieldSize/2$, annars -1. Eftersom vi inte vet hur stor indata kommer att vara sätts knowledge = -1. fieldZize är storleken på den kropp som beräkningarna utförs i.
`@param`	`fewRounds`	Eftersom det endast är en beräkning som skall utföras behöver vi inte använda oss av fler rundor som är till för att fördela ut beräkningarna bättre över den anslutna användarna. Att fördela ut beräkningarna minskar belastningen på bland annat minnet.
`@return`	`true`	Om operationen utfördes korrekt.

Hela paketet med alla klasser som tillhör protokollet måste sedan göras om till en .jar-fil för att kunna användas vid körning av programmet. Detta görs smidigast genom att exportera mpc.calcio paketet i Eclipse till en .jar fil. Om UNIX-system används går det även att paketera det med hjälp av kommandot `jar -cf calcio.jar mpc` i terminalen.

Steg 3 - Konfiguration

Calcio kan likt de färdiga protokollen från Sepia köras både på Windows och UNIX/Mac då det är skrivet i Java. Vi valde att köra protokollet på Windows 7 samt att köra alla beräkningar på en dator. Eftersom all kommunikation mellan olika användare finns inbyggd i Sepia går det lika bra att köra det på flera olika datorer. Till Sepia hör ett konfigurationsprogram där det är möjligt att styra hur många rundor som skall köras, vilket protokoll som ska köras, antal användare, antal privata användare samt mängd indata. Här går det även att ställa in i vilken ändlig kropp alla beräkningar skall göras i, det vill säga Z_p. Det förinställda värdet på p som Sepia använder sig av är 2^{63}-25.[38] När konfigurationsprogrammet används skapas alla nödvändiga filer så som certifikat och keytools. Även scriptfiler som används för att köra igång varje klients kommunikation och beräkningar skapas. I dessa specificeras vilka certifikat samt vilket protokoll som ska användas. Alla nödvändiga filer kan sedan på ett enkelt sätt distribueras till varje användare genom scriptet *distributeConfigs.bat* som också skapas genom programmet.

Figur 5 - Sepias konfigurationsprogram



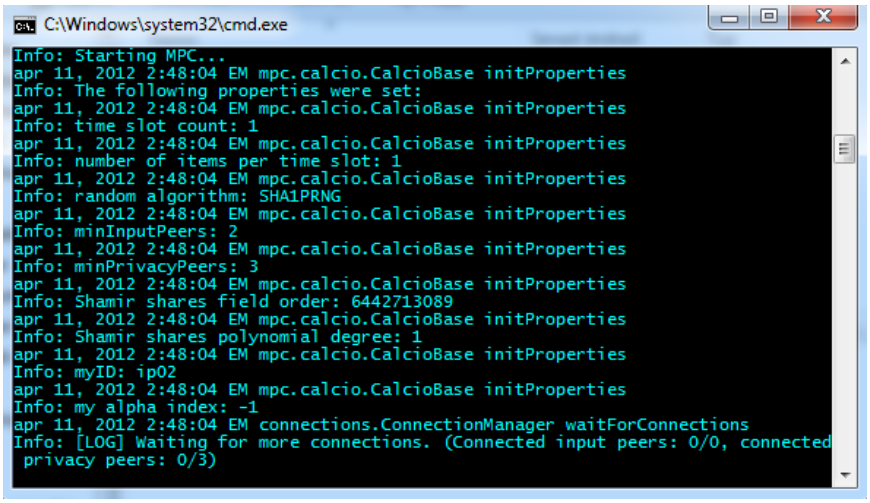
Steg 4 - Körning

Vid körningen ska användarna skriva in sina respektive bud i textfilen bid.txt som ligger i programmappen. Därefter kör varje användare igång sin *runPeer.bat*-fil vilket gör att programmet öppnas och ställer sig och väntar på att tillräckligt många personer blir anslutna. För att underlätta körning på en dator används scriptfilen *runCalcio.bat* som i sin tur startar upp alla användares klienter. Samma princip gällde för UNIX/Mac, men istället för .bat-filer användes shellscriptsfiler. När detta har skett skickas de krypterade delarna av buden mellan användarna och de privata användarna samtidigt som alla beräkningar görs. I slutändan skrivs det vinnande budet till en output-fil som även den ligger i programmappen hos varje användare. Programstrukturen fungerar på det sätt att endast det vinnande budet skrivs till output-filen.

runPeer.bat

```
java -cp "../../sepia.jar;../../calcio.jar" -
Djavax.net.ssl.trustStore=pKeyStore.jks MainCmd -v -p 0 -c
config.properties
pause
```

Figur 6 - En användare väntar på fler anslutningar



Slutsats

En av anledningarna till att SMPC inte är särskilt utbrett idag är att det inte finns någon bra applikation. SMPC är fortfarande ett forskningsprojekt och har varit det ända sedan 1980-talet. Allting måste köras via terminalkommandon, vilket förvisso inte är något problem för en person med bakgrund inom datalogi. Däremot finns det idag inga protokoll som riktar sig mot användare utan dessa förkunskaper. Med andra ord är de befintliga protokollen och biblioteken endast anpassade efter utvecklare, snarare än användare. De installationsguider som finns idag är enligt oss bristfälliga och biblioteken är ofta beroende av gamla versioner av annan mjukvara vilket leder till många problem med installationer.

Vi tror även att säkerhetsaspekten är en stor anledning till att det inte används. Det är svårt att förmedla att systemet verkligen går att lita på när det handlar om att dela med sig av känslig information. Det är viktigt att bygga upp ett förtroende kring denna typ av system, särskilt när det har varit så pass stora problem med korruption. Vi menar att det finns stora fördelar med att vara tydlig med hur allting fungerar.

Protokoll för hemliga budgivningar

Många pekar idag på att företroendet för organisationerna inom fotbollen är skadat och då särskilt i Italien. Även om det inte finns några kända fall, vad vi erfar, där det har varit problem med de dolda budgivningarna, finns det ett flertal incidenter där klubbarna kringgått de regler som gäller vid övergångar. Vi menar att allt som kan bidra till fler felsäkra lägen inom de administrativa delarna av fotbollen kommer att reducera antalet skeptiker. Faktum är att samma personer som var inblandade i de skandaler som uppdagats nu fortfarande är aktiva inom fotbollen och i allra högsta grad involverade i spelarövergångar. Genom att applicera en lösning med SMPC på de dolda budgivningarna i Serie A tror vi att det skulle öka förtroendet för ligan eftersom det då finns ett mindre sätt att fuska på. SMPC skulle göra att det blev teoretiskt omöjligt att muta domaren för att få honom/henne att i hemlighet manipulera buden.

I ett verkligt scenario skulle representanter från varje klubb få hämta ut ett usb-minne från fotbollsförbundet där Calcio-protokollet finns installerat. Alternativet är att de utsända representanterna från klubbarna får sitta vid varsin dator i samma byggnad, hos fotbollsförbundet till exempel. För att öka säkerheten behövs fler privata användare än två. Genom att lägga privata användare hos en, av båda klubbarna, betrodd tredjepart ökar säkerheten.

Ett problem som inte går att komma ifrån med vår prototyp är det faktum att alla användare måste vara anslutna samtidigt. Detta göra att det i praktiken blir svårt att implementera prototypen på ett större antal användare än två. När det endast är två användare som måste vara anslutna blir svårigheterna mindre för parterna att komma överens om en tid att sätta igång systemet. Problematiken med att SMPC kräver direktanslutning mellan användarna bör inte vara några större problem i detta fall.

Rent tekniskt fungerar vår prototyp. Resultatet som fås ut är det vinnande budet samt att alla andra bud hålls hemliga. Det som skulle behöva läggas till i Calcio för att kunna vidare utveckla det för allmänheten är ett användargränssnitt framtaget med hjälp av användartester, ett tydligt sätt att installera programmet på samt att göra det lättillgängligt. Med lättillgänglighet menar vi att det idag är

väldigt svårt att hitta information om SMPC samt att det är få som är medvetna om denna lösning. Vi tänker oss att det endast ska finnas en enda fil som användaren ska behöva ladda ner för att installera Calcio. I denna fil kan användarna sedan konfigurera programmet och välja vilken beräkning som skall göras och med vilka.

För att det skulle fungera i verkligheten skulle det även behövas en kombination av dagens protokoll med en funktion för att hämta ut det vinnande budets användare och skriva detta till output-filen. Dock är kodstrukturen inte anpassad för detta i nuläget. Ett alternativ vore att ha ett separat protokoll som beräknar miljonärsproblemet med samma information som ligger i bid.txt. Detta protokoll skulle kunna sättas igång samtidigt som Calcio med hjälp av att ändra i .bat-filen så att båda protokollen aktiveras, men i ett verkligt scenario skulle detta i så fall skötas av den huvudfil som användaren laddar hem.

Referenser

1. Wikipedia, Wikimedia Foundation. *Secure multi-party computation* [Hemsida på Internet]. 2011 [Hämtad 2012-03-08]. Tillgänglig på http://en.wikipedia.org/wiki/Secure_multi-party_computation

2. Department of Computer Science, Yale University. M.J. Fischer.Yale. *Lecture Notes 21 – The Millionaire's problem* [Hemsida på Internet]. 2009 [Hämtad 2012-03-12]. Tillgänglig på http://zoo.cs.yale.edu/classes/cs461/2009/lectures/ln21.pdf

3. FIFA. *Regulations on the Status and Transfer of Players* [Hemsida på Internet]. 2007 [Hämtad 2012-03-20]. Tillgänglig på http://www.fifa.com/mm/document/affederation/administration/01/06/30/78/statusinhalt_en_122007.pdf

4. Department of Computer Science, Århus universitet. Institution of Food and Resources, Köpenhamns universitet. Department of Economics, The Alexandra Institute. Peter Bogetoft, Dan Lund, Ivan Damgård, Janus Dam Nielsen, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Jesper Buus Nielsen, Jakob Pagter, Michael Schwartzbach och Thomas Toft. *Secure Multiparty Goes Live* [Hemsida på Internet]. 2008 [Hämtad 2012-03-12]. Tillgänglig på http://eprint.iacr.org/2008/068.pdf

5. School of Computer Science and Communication, Kungliga Tekniska Högskolan. Douglas Wikström [Hemsida på Internet]. 2012 [Hämtad: 2012-03-09]. Tillgänglig på http://www.verificatum.org/verificatum/index.html

6. Computer Science and Artificial Intelligence, MIT. Ron Rivest. *Lecture 18: Mix-net voting systems* [Hemsida på Internet]. 2004 [Hämtad 2012-03-09]. Tillgänglig på http://courses.csail.mit.edu/6.897/spring04/L18.pdf

7. Se referens nr. 5.

8. SAS, Isaac Newton Institute. Douglas Wikström [Hemsida på Internet]. 2012 [Hämtad 2012-04-02]. Tillgänglig på http://www.newton.ac.uk/programmes/SAS/seminars/020211451.html

9. Wikipedia, Wikimedia Foundation. FIFA [Hemsida på Internet]. 2011 [Hämtad 2012-03-20]. Tillgänglig på http://en.wikipedia.org/wiki/FIFA

10. Se referens nr. 3.

11. Wikipedia, Wikimedia Foundation. Co-ownership [Hemsida på Internet]. 2011 [Hämtad 2012-03-21]. Tillgänglig på http://en.wikipedia.org/wiki/Co-ownership_(football)

12. Italian Calcio Blog. *Calciopoli Scandal 2006* [Hemsida på Internet]. 2007 [Hämtad 2012-04-07]. Tillgänglig på http://calcioitaliascandal.blogspot.se/

13. MicroSport. *Calciopoli: Moggi Promises Incendiary disclosures* [Hemsida på Internet]. 2011 [Hämtade 2012-04-07]. Tillgänglig på http://www.mirosport.net/2011/soccer/36960/calciopoli-moggi-promises-incendiary-disclosures/

14. Sportbladet, Aftonbladet. Jennifer Wegerup. *Besvikelse och bestörtning efter Zlatans intervju* [Hemsida på Internet]. 2007 [Hämtad 2012-04-07]. Tillgänglig på http://mobil.aftonbladet.se/sportbladet/fotboll/landslagsfotboll/landslaget/article11075020.ab?partner=www

15. The World Biography. Zainal Arifin. *Zlatan Ibrahimovic life story* [Hemsida på Internet]. 2012 [Hämtad 2012-04-06]. Tillgänglig på http://theworldbiography.blogspot.se/2012/03/zlatan-ibrahimovic-life-story.html

16. Totalfotbollsbloggen. Dixie Eriksson. *M som i Monti och Moggi* [Hemsida på Internet]. 2011 [Hämtad 2012-04-08]. Tillgänglig på http://www.totalfotboll.nu/author/de/page/2/

17. Sportbladet, Aftonbladet. Kalle Karlsson. *Erkänner: "Jag fick betalt för att göra självmål"* [Hemsida på Internet]. 2012 [Hämtad 2012-04-08]. Tillgänglig på http://mobil.aftonbladet.se/sportbladet/fotboll/internationell/italien/article14632196.ab

18. Forza Italian Football. Enzo Misuraca. *Italy legend Marco Tardelli: I can't remeber anything about that 1982 goal* [Hemsida på Internet]. 2012 [Hämtad 2012-04-09]. Tillgänglig på http://forzaitalianfootball.com/2012/02/italy-legend-marco-tardelli-i-cant-remember-anyting-about-that-1982-goal/

19. FourFourTwo. *One-on-one: Gianfranco Zola* [Hemsida på Internet]. 2010 [Hämtad 2012-04-09]. Tillgänglig på http://fourfourtwo.com/interviews/one-on-one/93/article.aspx

20. Se referens nr. 1.

21. Department of Information Technology and Electrical Engineering,ETH Zürich. Martin Buckhart, Mario Strasser, Dilip Many och Xenofontas Dimitropoulos. *SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics* [Hemsida på Internet]. 2010 [Hämtad 2012-03-13]. Tillgänglig på http://sepia.ee.ethz.ch/publications/ssym2010-sepia.pdf

22. Laboratory of Computer Science, MIT. Adi Shamir. *How to Share a Secret* [Hemsida på Internet]. 1979 [Hämtad 2012-03-23]. Tillgänglig på http://www.christophedavid.org/w/c/files/ShareSecret/ShamirHowToShareASecret.pdf

23. CS, Berkeley. David Wagner. *Lecture 4.14.04* [Hemsida på Internet]. 2004 [Hämtad 2012-03-23]. http://www.cs.berkeley.edu/~daw/teaching/cs276-s04/22.pdf

24. Laboratory of Computer Science, MIT. Adi Shamir. *How to Share a Secret* [Hemsida på Internet]. 1979 [Hämtad 2012-03-23]. Tillgänglig på http://www.christophedavid.org/w/c/files/ShareSecret/ShamirHowToShareASecret.pdf

25. Se referens nr. 23.

26. VIFF. Martin Geisler, Tomas Toft, Mikkel Krøigaard, Thomas Pelle Jakobsen, Jakob Illeborg Pagter, Sigurd Meldgaard, Marcel Keller, Tord Reistad, Ivan Damgård och Janus Dam Nielsen [Hemsida på Internet]. 2008 [Hämtad 2012-03-12]. Tillgänglig på http://viff.dk/

27. Department of Information Technology and Electrical Engineering,ETH Zürich. Martin Burkhart [Hemsida på Internet]. 2011 [Hämtad 2012-03-21]. Tillgänglig på http://sepia.ee.ethz.ch/

28. Cybernetica och University of Tartu. Dan Bogdanov [Hemsida på Internet]. 2012 [Hämtad 2012-03-21]. Tillgänglig på http://sharemind.cyber.ee/

29. Computer Science and Engineering, The Hebrew University of Jerusalem. Dahlia Malkhi, Noam Nisam och Benny Pinkas [Hemsida på Internet]. 2004 [Hämtad 2012-03-21]. Tillgänglig på http://www.cs.huji.ac.il/project/Fairplay/home.html

30. Department of Information Technology and Electrical Engineering,ETH Zürich. Martin Burkhart och Xenofontas Dimitropoulos. *Privacy-Preserving Distributed Network Troubleshooting—Bridging the Gap between Theory and Practice* [Hemsida på Internet]. 2011 [Hämtad 2012-03-23]. Tillgänglig på http://delivery.acm.org/10.1145/2050000/2043632/a31-burkhart.pdf?ip=130.229.134.219&acc=ACTIVE%20SERVICE&CFID=74895815&CFTOKEN=85973308&__acm__=1333115201_1b463bef512663d73016773c860424d2

31. Se referens nr. 22.

32. Se referens nr. 30.

33. Faculty of Mathematics and Computer Science, University of Tartu. Oleg Selajev. *The Use of Circuit Evaluation Techniques for Secure Computation* [Hemsida på Internet]. 2011 [Hämtad 2012-03-23]. Tillgänglig på http://sharemind.cyber.ee/files/papers/sharemind_circuits_shelajev_2011.pdf

34. VIFF. Martin Geisler, Tomas Toft, Mikkel Krøigaard, Thomas Pelle Jakobsen, Jakob Illeborg Pagter, Sigurd Meldgaard, Marcel Keller, Tord Reistad, Ivan Damgård och Janus Dam Nielsen. *How to install* [Hemsida på Internet]. 2008 [Hämtad 2012-03-28]. Tillgänglig på http://viff.dk/doc/install.html

35. Department of Information Technology and Electrical Engineering,ETH Zürich. Martin Burkhart, Dilip Many och Manuel Widmer. *User Manual and Developer's Guide* [Hemsida på Internet]. 2011 [2012-04-01]. Tillgänglig på http://sepia.ee.ethz.ch/download/v0.9/UserManual.pdf

36. Se referens nr. 35.

37. Wikipedia, Wikimedia Foundation. *Tsalis Distribution* [Hemsida på Internet]. 2011 [Hämtad 2012-04-01]. Tillgänglig på http://en.wikipedia.org/wiki/Tsallis_distribution

38. Se referens nr. 35.