Datavetenskap

Daniel Geschwind och Simon Ogolla

Kursutvärderingssystem på internet

Examensarbete, C-nivå

2002:22

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Daniel Geschwind

Simon Ogolla

Godkänd, 2002-06-05

Handledare: Donald Ross

Examinator: Donald Ross

Sammanfattning

Den här uppsatsen handlar om skapandet av ett säkert system där man kan fylla i och lämna in kursutvärderingar på internet. Systemet skall också spara undan all data och kunna dra statistik ifrån det. En av anledningarna till att vi valde att göra detta var att vi såg en utmaning i att detta var någonting relativt nytt och kommande på marknaden. En annan anledning var att förenkla jobbet och minska tiden det tar för att framställa en sammanfattning. Idag görs kursutvärderings sammanfattningar för hand och på papper. Det tar flera timmar att göra en sammanfattning för en stor klass.

Syftet med vår uppsats har varit att skapa en prototyp på ett kursutvärderingssystem. Denna prototyp skall kunna ligga som grund till fortsatt arbete i skapandet av ett helt fungerande system. Vi har delat in systemet i två huvudkomponenter:

- Implementation och design
- Säkerhet

Vårt huvudsakliga resultat är att prototypens implementation och design är av tillfredsställande grad medan att prototypens säkerhet är mindre än vad man skulle kunna önska sig, men med tanke på ansatt tid så var ingenting annat att vänta sig.

Vår slutsats är att prototypen kan ligga som grund för kommande arbete, men att mycket arbete krävs att göra på säkerheten. Väljer kommande personer att skapa ett nytt system, kan vår prototyp fungera som ett hjälpmedel. Med hjälpmedel menar vi att man har tillgång till ett sätt att lösa de problem som uppstår.

Abstract

This bachelor's project is about the creation of a secure system where one fill in and return a course evaluation over the internet. The system shall also be able to store all data and retrieve statistics from the data. One of the reasons why we chose to do this was that we saw a challenge in this because it was relatively new and upcoming area within the computing market. Another reason was to simplify the work, and reduce the time it takes, to produce a course evaluation summary. Today a course evaluation summary is performed by hand and written on paper and it takes several hours to create a summary for a large class.

The goal of our bachelor's project was to create a prototype for a course evaluation system. This prototype will act as a base for any future development of such a system. We have divided the system into two major components:

- Implementation and design
- Security

Our main result is that the implementation and design of the prototype is of a satisfactory level. The security of the prototype is far less than one could wish for, but considering the time restrictions placed on the project, this was perhaps to be expected. Our conclusion is that the prototype can act as a base for future work but there is still considerable work remaining on the security of the system. If the future developers choose to create a new system, then our prototype can act as a starting point. By that we mean that the future developers have access to one method for solving any potential problems they may encounter.

Tack

Under vårt arbete, i skapandet av en prototyp på ett kursutvärderingssytem, har vi stött på olika typer av problem. Ett stort tack vill vi rikta till Mari Göransson som tog sig tid att hjälpa oss på vägen med olika implementations problem.

Vi vill också tacka Simone Fisher Hubner för att finnas där för frågor om säkerhet.

Vår handledare Donald Ross har varit en pådrivande faktor vilket har varit bra för oss. Vi vill tacka honom för det.

Innehållsförteckning

1 Inledning 1

2 Bakgrund 5

2.1 Diskussion av projektet 5
2.1.1 Befintliga system 7
2.1.2 Prototyp 8
2.1.3 Säkerhet 9

2.2 Sammanfattning 9

3 Prototypens design och implementation 11

3.1 Eventuella lösningar 11
3.1.1 HTML 12
3.1.2 PHP 12
3.1.3 ASP 14
3.1.4 CGI – Common Gateway Interface 15
3.1.5 MySQL 15

3.2 Val av verktyg för detta projekt 17

3.3 Prototyp beskrivning 17
3.3.1 Formulär 18
3.3.2 Inloggningssida 19
3.3.3 Huvudsidan 22
3.3.4 Ändra i kursutvärdering 24
3.3.5 Statistiksidan 27
3.3.6 Verifieringssidan för studenterna 29
3.3.7 Formuläret för studenten 30

3.4 Databasbeskrivning 31

3.5 Sammanfattning 33

4 Säkerhetsaspekter 35

4.1 Symmetrisk och asymmetrisk kryptering 35

4.2 Krypteringsalgoritmer 37
4.2.1 RSA 37
4.2.2 MD5 37
4.2.3 DES 38

4.3 Digital Signatur 39

4.4 Blinda Signaturer 39
4.4.1 Rösta med blinda signaturer 41

4.5 Prototypbeskrivning av säkerhet 44

4.5.1 Lösenord/inloggning ... 44
4.5.2 Enkelt biljett system ... 44
4.6 Sammanfattning ... 45

5 Resultat och utvärdering ... 47

5.1 Prototyp ... 47
5.1.1 Språk ... 47
5.1.2 Användargränssnitt ... 48
5.1.3 Databas ... 48
5.1.4 Dynamik ... 49
5.2 Säkerhetsaspekter ... 50
5.2.1 Röstning ... 50
5.2.2 Inloggning/konto ... 51
5.2.3 Databas ... 52
5.3 Sammanfattning ... 53

6 Sammanfattning ... 55

6.1 Detta projekt ... 55
6.1.1 Prototyp ... 55
6.1.2 Säkerhet ... 56
6.2 Framtida Utvecklingsarbete ... 56
6.3 Andra Tillämpningar ... 57
6.3.1 Valsystem ... 57
6.3.2 Duggor/tentor ... 57
6.4 Allmän slutsats ... 57

Referenser ... 59

A Bilaga Systemöversikt ... 61

List of Figures

Figur 2.1: Prototyp modell 6
Figur 2.2: Äldre systemet 7
Figur 3.1: Webbsida på exempel 3.2 13
Figur 3.2: PHP - Exempel 14
Figur 3.3: Formulär 19
Figur 3.4: Inloggningssidan 20
Figur 3.5: Screenshot på webbsidan lägg till / ta bort användare 21
Figur 3.6: Kod för ta bort användare 21
Figur 3.7: Huvudsidan 22
Figur 3.8: Skapa ny Kursutvärdering sida 23
Figur 3.9: Ändra i utvärdering 25
Figur 3.10: Frågetabell 26
Figur 3.11: Svarstabell skapas 26
Figur 3.12: Statistik över flerval 27
Figur 3.13: Uppbyggnad av diagram 28
Figur 3.14: Lista med alla fritextsvar 29
Figur 3.15: studenternas verifieringssida 30
Figur 3.16: Studentens formulär 30
Figur 3.17: Svarstabell i databasen 31
Figur 3.18: Databasens fyra grundtabeller 32
Figur 3.19: Nya tabeller som skapas vid ny utvärdering 33
Figur 4.1: Asymmetrisk kryptering 36
Figur 4.2: Symmetrisk kryptering 36
Figur 4.3: DES-kryptering 38
Figur 4.4: Digitala signaturer 39
Figur 4.5: Röstning med blinda signaturer 43
Figur 4.6: Enkelt biljettsystem 45

Figur 5.1: Verifiering på varje sida 51
Figur 5.2: Verifiering endast på loginsidan..... 52

1 Inledning

Denna uppsats handlar om ett kursutvärderingssystem och säkerheten som behövs för ett röstningssystem. Vårt system konstruerades från grunden trots att det fanns ett liknande system på skolan. I kapitel 2 går vi igenom varför vi valde att göra ett nytt system från grunden och varför ett system som detta kan behövas. Det som krävs för kursutvärderingssystem, som skall finnas på internet är att det är enkelt och att folk inte kan fuska. Bakom användargränssnittet finns en databas som lagrar all information om hur kursutvärderingar ser ut och vad studenterna svarat i sina utvärderingar.

När en prototyp för ett kursutvärderingssystem skall skapas så måste man ha ett verktyg för att skapa sidor på Internet och en databas som lagrar data. Sidorna som skapas måste skapas dynamiskt och då räcker det inte med HTML som är det sidbeskrivningsspråk som används för att göra hemsidor. I kapitel 3 tar vi upp olika språk som kan användas för att skapa dynamiska sidor, vi tittar också på några databaser som finns tillgängliga i dagsläget. Till Prototypen valdes skriptspråket PHP för att skapa dynamiska sidor och en MySQL databas. I kapitel 3 går vi också igenom hur prototypen ser ut. I prototypen har lärare ett eget konto för att logga in på systemet. Väl inne i systemet så finns det några olika funktioner som läraren kan utföra. De basala funktionerna, för lärarna, som är implementerade i prototypen är ”Skapa ny utvärdering”, ”Ändra på befintlig utvärdering”, ”Ta bort utvärdering” och ”Se på statistik”.

När en kursutvärdering skall skapas får läraren välja om han vill konstruera en utvärdering från grunden eller om han vill använda en mall. Konstruktionen av en utvärdering sker dynamiskt, d.v.s. att det går att lägga till och ta bort frågor samtidigt som man får se hur utvärderingen kommer att se ut. De nya frågorna som skapas till utvärderingen kan väljas till att vara endera flervalsfrågor eller fritextfrågor. Flervalsfrågorna har fem alternativ från ”dåligt” till ”mycket bra”. Skulle inte läraren vara nöjd med sin utvärdering kan han gå in och ändra på den, om han inte har publicerat utvärderingen.

Om man väljer att ta bort en utvärdering så försvinner all information till den utvärderingen från databasen, alla frågor och alla svar. Detta är en riskfaktor då lärarna kan ta bort andra lärares utvärderingar.

När läraren vill se statistik, för en utvärdering, får han gå till statistiksidan. Där finns alla svar uppdelade i flervalssvar och fritextssvar, flervalssvaren visas i stapeldiagram och fritextsvaren visas i en tabell.

När studenterna skall utvärdera en kurs måste de verifiera sig och få ett lösenord, som de senare använder för att logga in på utvärderingen som de skall göra. När de har gjort det så fyller de i utvärderingen och trycker på en klar-knapp, om de har kryssat i alla nödvändiga frågor så sparas svaren till databasen. Annars kommer det fram en text som beskriver felet.
Databasen lagrar all information till systemet i olika tabeller. Grunden i databasen består av fyra tabeller som alltid finns där, utöver det så skapas det två nya tabeller för varje utvärdering, en för frågor och en för svar.
Till ett kursutvärderingssystem finns det vissa säkerhetsaspekter, dessa aspekter går vi igenom i kapitel 4. I ett röstningssystem, vilket i grunden liknar ett kursutvärderingssystem, finns det vissa krav för att förhindra fusk och skydda varje individs enskildhet. Det är följande sex krav:

1. **Endast behöriga röstare kan rösta.**
2. **Ingen kan rösta mer än en gång.**
3. **Ingen kan ta reda på vad någon annan har röstat.**
4. **Ingen kan duplicera någon annans röst.**
5. **Ingen kan ändra någon annans röst utan att bli upptäckt.**
6. **Varje röstare kan vara säker att hans röst har kommit med i sluträkningen.**

I prototypen finns ett enkelt biljettsystem implementerat för att uppfylla några av de krav som finns beskrivna ovan. Biljettsystemet genererar nya biljetter för varje röstare som identifierar sig mot systemet. Biljetten som genereras skall användas för att få lägga sin röst eller i detta fall sin utvärdering. Om man använt sin biljett en gång, går det inte att använda den igen. Om man vill att alla krav skall uppfyllas kan man istället använda sig av blinda signaturer. För att förstå sig på blinda signaturer måste man veta lite om krypteringsmetoder och digitala signaturer vilket vi går igenom mer i detalj i kapitel 4. Man måste även ha säkerhet för åtkomsten av lärarnas sidor. Man vill inte att någon skall komma runt systemet på något sätt. För att lösa detta tar varje sida emot användarnamn och lösenord, då måste man logga in med korrekt användarnamn med tillhörande lösenord. För att förhindra att någon kommer in i databasen och ser lösenorden så krypteras varje lösenord innan dom sparas där. Om någon kommer åt de krypterade lösenorden så kan de ändå inte logga in i systemet.
När ett system konstrueras från grunden kan man inte göra allt på det bästa sättet från början. I kapitel 5 diskuteras vad som har gjorts och vad som kan göras bättre, och i sådana fall hur. Språket som vi valde att implementera i visade sig vara ett enkelt språk att sätta sig in i. Prototypen skapades så att den fick ett enkelt användargränssnitt men databasen byggdes upp

lite ostrukturerat. Detta berodde på att vi löste ett problem så att vi var tvungna att skapa nya tabeller för varje utvärdering. Dynamiken gjorde det möjligt för lärarna att skapa utvärderingar med ett okänt antal frågor. Dynamiken gör även att man slipper sammanställa statistik för hand. När säkerheten till röstningen skulle implementeras var det inte mycket tid över, så alla sex krav för säkerhet blev inte uppfyllda. Om man istället hade implementerat röstning med blinda signaturer så hade alla sex krav uppfyllts.

Säkerheten kring röstningen och säkerheten för lärarnas sidor implementerades. Enda sättet att komma åt lärarnas sidor är att ha ett giltigt användarnamn med tillhörande okrypterat lösenord.

Databasen som all information är lagrad i, finns på ett konto i skolan. Det är alltså en annan root-användare till systemet. Vill man att databasen skall vara säker bör man bygga upp databasen själv så att ingen annan har rättigheter till den.

2 Bakgrund

De flesta kursutvärderingar idag på Karlstads universitet sker på pappersform. Efter att studenterna har fyllt i en kursutvärdering för en kurs, så är det någon som måste sammanställa alla svar och dra slutsatser från dem. Dessa sammanställningar och svar måste också sparas undan någonstans för framtida behov. Efter en tid kan det vara svårt att hålla koll på dessa mängder av papper som måste sparas undan. Ett kursutvärderingssystem på internet skulle förenkla jobbet avsevärt för den person som är tvungen att sammanställa alla utvärderingar. Det skulle även lösa problemet med lagring av tidigare gjorda sammanställningar. Man skulle även kunna jämföra utvärderingar från flera år bakåt i tiden, utan att behöva leta fram en massa papper. Vi tror också att det skulle vara minst lika enkelt för studenten, om inte enklare, att fylla i utvärderingen på internet. Studenten kan då göra utvärderingen när det passar honom/henne. Vi tror att de flesta studenter inom de tekniska programmen här på Karlstads universitet hellre gör sin utvärdering på internet än på papper. När vi bestämde oss för att göra detta projekt så fick vi reda på att det fanns ett system på skolan som hanterade kursutvärderingar. En del av vårt projekt gick ut på att analysera det tidigare systemet och se om det mötte våra krav om implementation och säkerhet. Om systemet mötte våra krav, skulle vi fortsätta på det existerande och utveckla det. Skulle det inte passa på något sätt, så var vår uppgift att skapa en ny prototyp för ett system som kan hantera kursutvärderingar. Detta system skulle också ha en viss säkerhet.

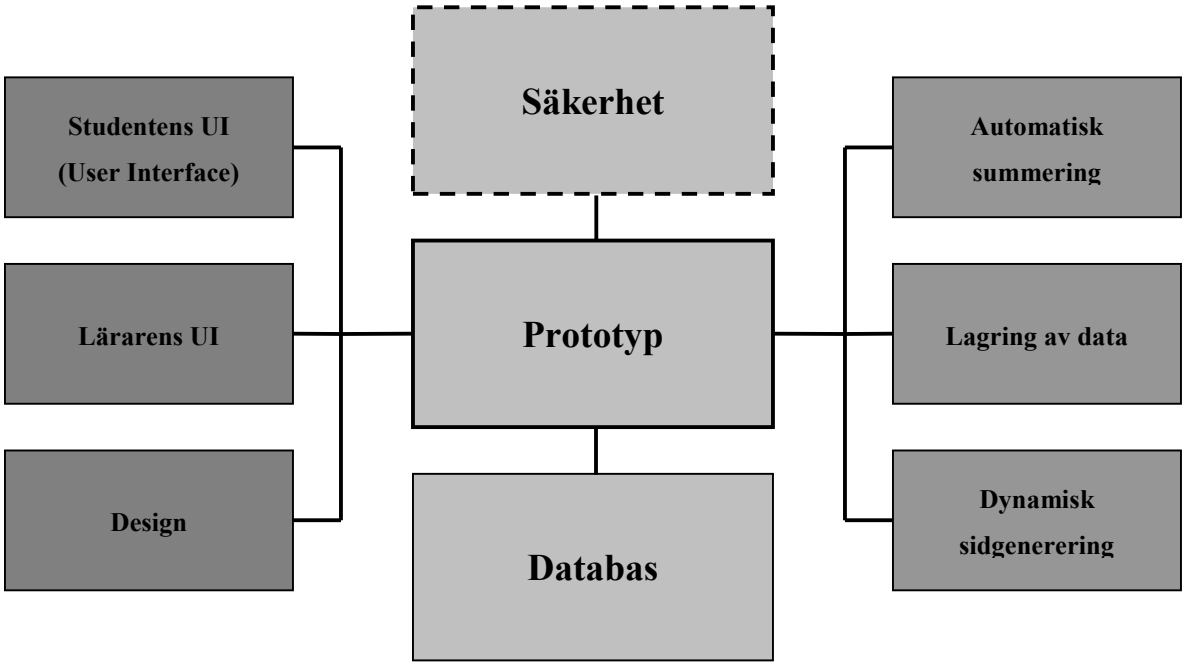
2.1 Diskussion av projektet

Projektet består till största del av två huvudkomponenter:

- **Prototyp**
- **Säkerhet**

Beroende på om det tidigare gjorda systemet är tillgängligt och fungerar på ett relativt bra sätt, skall vi fortsätta på det systemet och utveckla det. Skulle det tidigare gjorda systemet däremot inte vara tillgängligt eller inte passa på något sätt så skall en prototyp på ett nytt system skapas. Denna prototyp skall bland annat innehålla en sida där studenten kan fylla i sin kursutvärdering. Prototypen skall även kunna summera och producera statistik på de svar som lagras undan i databasen. Data från dessa utvärderingar skall lagras och kunna tas fram vid ett

senare tillfälle. Prototypen skall kunna ligga till grund för eventuell påbyggnad. För att det skall vara enkelt att lägga till nya funktioner och ändra i existerande system, skall prototypen vara uppdelad i funktioner. Användargränssnittet skall vara behagligt för så många som möjligt av användarna och systemet skall vara enkelt att förstå sig på.

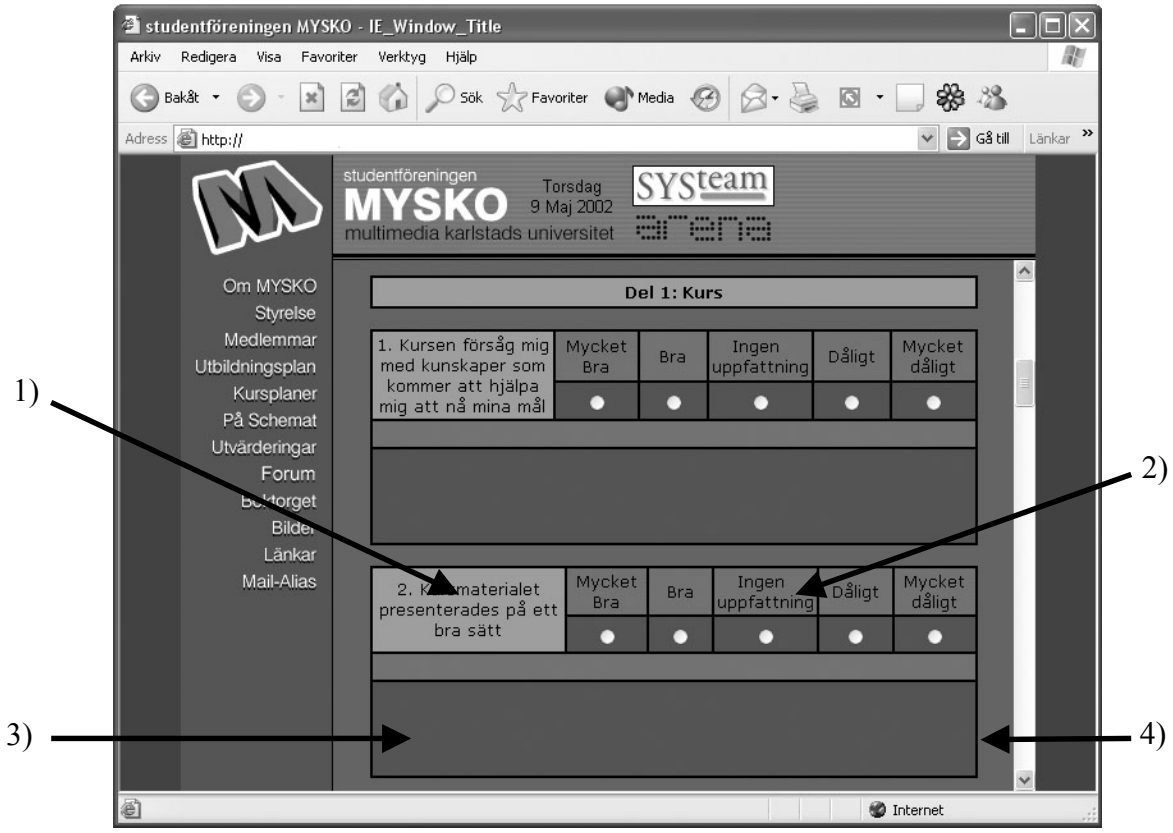


Figur 2.1: Prototyp modell

Figur 2.1 visar en bild över vår prototypmodell och hur de olika delarna hänger samman. Som vi också kan se från figur 2.1 så skall själva prototypen vara den största grundstenen till detta projekt. Till denna grundsten räknar vi också databasen. Den andra grundstenen är säkerheten, vilket också är en mycket viktig del. Prototypen skall använda sig av en databas för lagring av data. Databasen ger prototypen en ny dimension för att realisera dynamiken, som vi senare kommer att ge exempel på. Prototypen skall kunna ligga till grund för ett framtida system. Den ska också kunna visa hur man kan skapa ett sådant system genom dynamiska webbsidor. System där man kan göra utvärderingar, tentor eller val är det inte så stor skillnad mellan, då man talar om funktioner. Det som kan skilja är just säkerheten. Om man t.ex. vill göra ett system där man kan rösta till ett regeringsval på internet, så är det självklart att säkerheten är väldigt viktig. Säkerheten är viktig i andra fall också, men då man vill kunna hålla val över internet så krävs det att systemet är mycket säkert. Vi kommer att gå igenom olika säkerhetsbegrepp i kapitel 4 som kan hjälpa till för att göra ett system säkert. Vi ska även försöka realisera så att vårt system kommer att få en mindre grad av säkerhet.

2.1.1 Befintliga system

På Karlstads universitet fanns det redan ett system som hanterade kursutvärderingar. Det var Multimedia institutionen som hade köpt in det från en före detta student. Systemet var skrivet i CGI (Common Gateway Interface). Vår analys av detta system gjordes väldigt överskådligt eftersom det varken fanns kod eller dokumentation att tillgå.



Figur 2.2: Äldre systemet

Figur 2.2 visar en del av en kursutvärdering från det tidigare gjorda systemet. Figuren ovan är gråskalig, men i själva verket är webbsidan gjord i olika nyanser av blå. Det vi ser på bilden är två frågor i en kursutvärdering. Varje fråga består av själva frågan (1), svars alternativ (2) och en fritext ruta (3). Varje fråga omges också av en linje som skiljer frågorna åt (4). Själva systemet verkade fungera på ett bra sätt, men det fanns inte tillräckligt med säkerhet som man skulle vilja. Användargränssnittet och valet av färg tycker vi att det kunde ha varit bättre. Ett exempel på det är att varje fråga i utvärderingen har ett fält där man kan fylla i sina egna kommentarer i en fritext ruta (3). Dessa rutor har samma bakgrundsfärg som mycket annat i omgivningen och det finns ingenting som markerar att de finns. Vi tycker att det skulle ha varit tydligare med en vit textruta och en beskrivning till den. I största allmänhet verkar systemet vara bra men inte särskilt säkert, vi kan inte ta reda på så mycket mer eftersom vi saknar koden.

2.1.2 Prototyp

Vi valde att börja från början med ett nytt system. Detta berodde främst på att koden för det existerande systemet var otillgänglig och att det inte fanns någon dokumentation till det. Vi var även nyfikna på hur man gjorde ett nytt system.

Systemet skall göras så automatsikt som möjligt, så att hjälpen till den person, som ska sammanfatta utvärderingarna, blir så stor som möjligt. Kommentarer till frågor kommer fortsättningsvis att behöva utvärderas och sammanfattas. Denna sammanfattning kommer sedan att lagras med den övriga informationen till utvärderingen. Systemet skall göras så användarvänligt som möjligt, vilket bland annat betyder att mycket skall ske automatiskt.

2.1.2.1 Användargränssnitt

Som vi tidigare sagt så skall användaregränssnittet vara behagligt och enkelt. Lärare och studenter skall enkelt kunna se statistik över tidigare gjorda utvärderingar. Det skall inte heller vara några större svårigheter för lärare att producera nya kursutvärderingar för sina kurser. Läraren har också alternativ där han kan ta bort sina utvärderingar. Där det finns alternativ där man kan ta bort eller redigera informationen på databasen så får man en förfrågan om man verkligen är säker. Detta kan förhoppningsvis motverka att man genom misstag gör någonting som man inte vill.

2.1.2.2 Databas

Systemet ska presentera statistik från ifyllda kursutvärderingar. Denna statistik visar hur många personer som röstat vad på varje fråga. Statistiken hämtas från en databas genom olika frågor man ställer mot den, för att sedan presenteras som diagram på webbsidan. Databasen ska också innehålla all konto information till systemet, d.v.s. användarnamn och lösenord. Databasen ska även hjälpa till att utöka dynamiken hos systemet, vilket den gör genom att innehålla information som hämtas på ett dynamiskt sätt. Ett exempel på hur man kan hämta data på ett dynamiskt sätt, kan vara den sida som studenterna skall kunna fylla i sitt utvärderingsformulär på. För att den sidan skall vara dynamisk så kan man i förväg inte veta hur många frågor som skall produceras till formuläret. Dessa frågor kommer att ligga i databasen och läsas in till sidan i tur och ordning. Sidan kommer då bara att producera så många frågor som det finns i databasen utan att från början ha en aning om hur många frågor som fanns där. Det kommer även att behövas ett visst antal olika variabler beroende på hur många frågor det är, och det vet sidan inte heller om förrän den hämtar data från databasen.

2.1.3 Säkerhet

Säkerhet är en viktig del av alla system. Den är ännu viktigare för system som har tillgång till internet. Beroende på vad man har för system kan man välja olika nivåer av säkerhet. Om vi tar ett valsystemet som exempel så skriver Schneier [1] att: ”*Computerized voting will never be used for general elections unless there is a protocol that both maintains individual privacy and prevents cheating*”. Han tar också upp sex krav som är det minsta ett system måste uppfylla för att räknas som någorlunda säkert.

1. **Endast behöriga röstare får rösta.**
2. **Ingen kan rösta mer än en gång.**
3. **Ingen kan ta reda på vad någon annan har röstat.**
4. **Ingen kan duplicera någon annans röst.**
5. **Ingen kan ändra någon annans röst utan att bli upptäckt.**
6. **Varje röstare kan vara säker på att deras röst har blivit räknad.**

Säkerheten för prototypen skall i mån av tid göras så säker som det går. Ett ideal vore om alla krav som beskrivs ovan uppfylls. Om man går tillbaka till exemplet om regeringsvalet så är det oerhört viktigt med att alla dessa krav plus kanske några därtill uppfylls.
Om man har löst alla dessa sex punkter för att skapa ett säkert system, så betyder de inget om inte databasen är säker. Det säkraste är om man har satt upp en egen databas och inte använder sig av ett konto hos någon annan, för då kan root-användaren komma åt all information som databasen innehåller.

2.2 Sammanfattning

Vi valde att skapa ett helt nytt system, dels på grund av att det inte fanns någon dokumentation eller kod av det tidigare gjorda systemet och dels på grund av att vi ville lära oss hur man från grunden skapar ett sådant system.
Projektet består till huvudsak av två komponenter. Den ena komponenten är själva prototypen som skall fungera på ett tillfredsställande sätt. Mycket ska genereras automatiskt. Det skall vara ett enkelt användargränssnitt och det skall vara lätt att sätta sig in i hur systemet fungerar. Systemet skall även vara kopplat till en databas för att på ett enkelt sätt kunna lagra all data och ge systemet en ännu mer dynamisk förmåga. Den andra komponenten är säkerheten till prototypen. Prototypen skall göras så säker som det går att göra på ansatt tid.

3 Prototypens design och implementation

Det finns idag olika språk som man kan använda för att skapa webbsidor på Internet. Vill man även att webbsidan skall vara dynamisk, minskar alternativen av språk att välja mellan, men det finns fortfarande ett urval kvar bl.a. CGI (Common Gateway Interface), ASP (Active Server Pages) och PHP (PHP Hypertext Preprocessor). Vi kommer att gå igenom de mest använda språken, som man i dagsläget använder för att producera dynamiska webbsidor.
Vill man ge sin dynamiska webbsida ytterligare en dimension kan man koppla samman den med en databas, som vi tidigare har gått igenom. De exempel på språk som vi kommer att gå igenom stödjer de flesta olika typer av databaser. Tillsammans så utgör, något av dessa språk och en databas, ett bra verktyg för att skapa dynamiska webbsidor.
Det som ligger till grunden för att skapa dessa dynamiska sidor är språket HTML.
Designen är väldigt viktig i ett system. Har systemet en dålig design så kommer förmodligen systemet inte att användas i den utsträckning det skulle ha gjorts med en bra design. Det är viktigt att ett system har ett användarvänligt gränssnitt, så att användaren känner sig så behaglig som möjligt med miljön. Vi kommer att försöka få en enkel och funktionell design på vår prototyp. Med det menar vi att det inte skall finnas för mycket information på varje sida, det skall bara finnas så mycket funktioner som det behövs. Det skall vara lätt att ta bort och lägga till nya funktioner. Funktionerna skall endast göra det som det är tänkt att funktionen skall göra. Om man delar upp koden i olika funktioner är det lättare att sätta sig in i koden för en person som inte har gjort den.

3.1 Eventuella lösningar

I början fanns det flera eventuella lösningar till vad man skulle kunna använda sig av för språk och databas till vår prototyp. Vi försökte begränsa oss i valet av eventuella lösningar. Vi började med att minimera valet av språk till tre av de absolut vanligaste språken för att producera dynamiska webbsidor. Ett av språken som vi kunde välja var CGI. CGI är det språk som det tidigare gjorda systemet är skrivet i. Vi valde också mellan två språk som liknar varandra väldigt mycket, PHP och ASP. För att skapa riktigt dynamiska webbsidor krävs det också att man har en databas kopplad till systemet. Vi begränsade oss i valet av databas till två stycken, Access och MySQL. Vi begränsade oss ytterligare i valet av programspråk och

databas då vi undersökte vilken databas som kompletterade vilket språk bäst. Dessa undersökningar resulterade i att vi tillslut hade tre val att välja mellan. Vi valde mellan ASP kominerat med Access som båda kommer från microsoft, PHP kombinerat med MySQL och CGI kombinerat med MySQL. Notera att programspråken kan använda sig av nästan vilken databas som helst. Det vi gjorde var att vi kollade vilket språk och databas som kompletterade varandra bäst.

3.1.1 HTML

HTML står för Hypertext Markup Language. HTML är ett sidbeskrivningsspråk som används för att göra webbsidor. Tanken är att man ska kunna se HTML-dokument oavsett vilken browser man använder. När man gör en webbsida i HTML så strukturerar man upp sidan med hjälp av taggar (<kommando>). Det finns alltid en tillhörande slut-tag till en start-tag. Om programmeraren glömmer att skriva in en slut-tag så kommer den att genereras ändå. Vart i koden vet man inte. Det är beroende på hur programmet tolkas av HTML. Om man exempelvis vill att en text skall vara fet och skrivas ut på webbsidan, kan det se ut som exempel 3.1.

Exempel 3.1

```
<html><B>Hello World</B></html>
```

<*html*> är en start-tag som talar om att kommande text är HTML-kod. <*B*>-tagen säger att den text som kommer efter är i fet stil. Oftast vill man ha ett slut på den feta texten och då skriver man en slut-tag, <*/B*>. <*/html*>-tagen talar om att nu är det slut på HTML-kod.
Det går att skapa ett HTML-dokument med vilken text-editor som helst, bara man kan spara filen som ”.html” eller ”.htm”. Det är ingen skillnad att använda .html istället för .htm, .htm kommer från operativsystemet Dos oförmåga att hantera mer än tre stycken filändelser.

3.1.2 PHP

PHP är ett serverbaserat skriptspråk som används för att skapa dynamiska webbplatser. Dess användning och popularitet ökar kraftigt världen över. Språket är plattformsoberoende och kan användas på de flesta webbservrar. PHP är anpassad för Webbutveckling och kan bäddas in i HTML. PHP-baserade webbsidor behandlas precis som en vanlig HTML sida, man kan skapa och ändra i dem på samma sätt som i HTML dokument.
Notera att detta inte är som ett CGI skript. Filen behöver inte vara exekverbar eller speciell på något sätt. Tänk på det som om det vore en vanlig HTML sida som har specialtecken i sig som kan göra mycket intressanta saker.

Exempel 3.2

```
<html><head><title>PHP Test</title></head>
<body>
<? php echo "Hello World <p>"; ?>
</body></html>
```

Detta exempel är väldigt enkelt och man behöver inte att använda PHP för att skapa en sådan sida. Själva PHP koden börjar med ”**<?**” eller ”**<?php**” och slutar med ”**?>**”. Innanför dessa taggar så skrivs PHP-koden. Echo betyder att man skall skriva ut någonting på skärmen och vad som ska skrivas ut kommer efter. Exempel 3.2 kommer att se ut som figur 3.1 i webbläsaren

Figur 3.1: Webbsida på exempel 3.2

När en användare sedan besöker din webbsida som är programmerad i PHP, så är det inte koden du skrev som skickas tillbaka till användarens webbläsare. PHP-koden skickas först till en programtolk på webbservern som exekverar eller utför koden i samma ögonblick som användaren vill se webbsidan. Programtolken skriver sedan en basis av exekveringen en helt ny webbsida i traditionell HTML-kod utan PHP-kod, och det är denna som skickas tillbaka till användaren. Användaren kommer därför aldrig att få se någon PHP-kod. Den kod som användaren kommer att se från exempel 3.2, ser ut som exempel 3.3 nedan.

Exempel 3.3

```
<html><head><title>PHP Test</title></head>
<body>Hello World<p></body></html>
```

Användaren kommer alltså inte att se någon PHP-kod överhuvudtaget. För att se det lite tydligare kan man använda sig av en for-loop, enligt figur 3.2.

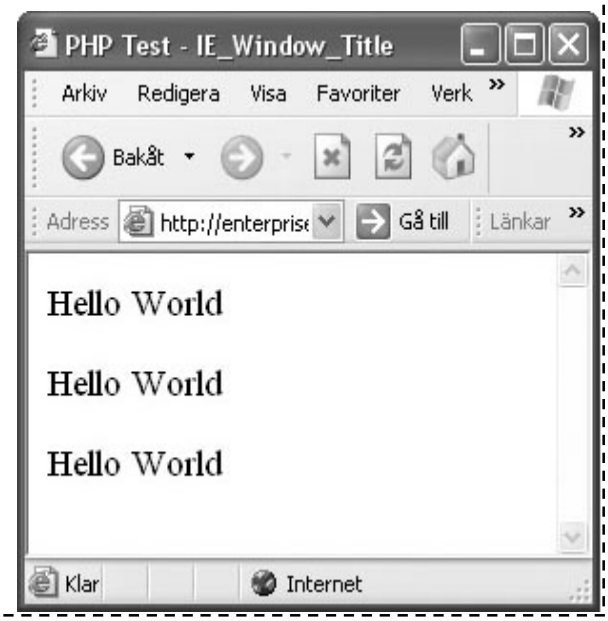
PHP-koden på server sidan

```
<html><head><title>PHP Test</title></head>
<body>
<? for($i=0; $i < 3; $i++)
{
echo "Hello World<p>";
}
?>
</body></html>
```

Koden som användaren ser

```
<html><head><title>PHP Test</title></head>
<body>
Hello World<p>Hello World<p>
Hello World<p>
</body></html>
```

Webbsidan för användaren



Figur 3.2: PHP - Exempel

En av de stora fördelarna med PHP är att det har stöd för de flesta olika typer av databaser. Genom att koppla en databas till PHP kan man få möjlighet att skapa riktigt dynamiska webbsidor. Man kan lagra bestående data och manipulera dessa via PHP-skript.

3.1.3 ASP

ASP, som står för Active Server Pages, är en teknologi nyligen utvecklad av Microsoft för att kunna baka in enklare programmeringskod, så kallad ASP-kod, i HTML koden. Poängen med detta är att göra webbsidan dynamisk, det vill säga när olika användare vid olika tidpunkter laddar ner webbsidan kommer olika resultat att visas. Det enklaste exemplet på en dynamisk webbsida är en webbsida med en klocka, som visar den aktuella tiden. En sådan sida går inte att skapa med vanlig HTML-kod. Om du skriver att klockan är 23.35 i HTML-koden, så kommer detta bara att stämma en gång per dygn. Med ASP kan du lägga in ett kommando i HTML-koden som skriver om HTML-koden i det ögonblick sidan laddas ner, så att den aktuella tiden skrivs in. På det sättet blir webbsidan dynamisk.

För att markera att det är ASP-kod inbakad i HTML-koden ska filen ha tillägget .asp istället för t.ex. .html. ASP fungerar på samma sätt som PHP, dvs användaren kommer bara att få se ren HTML-kod och ingen ASP-kod. För att deklarera att det är ASP-kod som man vill skriva så börjar man med att skriva ”<%” och när man vill sluta så skriver man ”%>”. Mellan dessa tagar skrivs själva ASP-koden.

3.1.4 CGI – Common Gateway Interface

CGI är en betydligt äldre, men för den skull inte sämre teknik än PHP och ASP. CGI är en standardiserad metod för att låta ett program på en webbserver kommunicera med en webbläsare i ren ASCII-kod. CGI är alltså inte ett programmeringsspråk. Så länge den överenskomna metoden följs, kan CGI-programmet vara skrivet i vilket språk som helst. Det i särklass vanligaste språket för CGI-program är Perl, men det finns även gott om CGI-program på nätet skrivna i C++, C, Java och Visual Basic. CGI används framförallt på **Unix-servrar**, även om funktionen finns på Windows NT och andra system.

Beroende på vad man använder för språk brukar man också se olika filändelser. Perl ger t.ex filändelserna **.pl** eller **.cgi**, C++ ger .cgi, .c eller .exe. Någon tydlig regel finns inte och ändelsen beror väldigt mycket på vad det är för typ av fil. En exe-fil är ju t.ex en körbar fil i Windows. Cgi-scripts på en hemsida brukar ha ändelserna .cgi eller .pl.

Exempel 3.4

```
$mystring = "Hello World";
print ($mystring);
```

Exempel 3.4 visar hur man i CGI kan skriva ut Hello World på skärmen. Det kommer att se ut på samma sätt som figur 3.1

3.1.5 MySQL

MySQL är ett system för att hantera den data som ligger sparat i en databas, dvs. en databas server. En databas är en strukturerad samling av data. Den data som finns i en databas kan vara mycket varierande, det kan vara allt från matchresultat i fotboll till ett helt skolsystems organisation.

MySQL är ett relationsbaserat system. Relationsbaserade system sparar data i separata tabeller hellre än att spara all data på ett ställe. På så sätt får man snabbare åtkomst och större flexibilitet. För att kombinera data från olika tabeller så används definierade relationer mellan tabellerna. SQL i MySQL står för ”Structured Query Language”, SQL är det vanligaste standardiserade språket för att göra förfrågningar mot en databas.

Programvaran MySQL är Open Source. Det betyder att det är möjligt för vem som helst att använda och modifiera. Man kan alltså läsa källkoden och ändra den så den passar ens behov.

Ett kommando i MySQL kan se ut på följande sätt:

mysql> **create table person (fnamn char(15), enamn char(30));**

Vad kommandot gör är att den skapar en tabell som heter person. I person-tabellen har det skapats två kolumner som heter fnamn och enamn. Fnamn kan bestå av max 15 tecken och enamn av max 30 tecken. Observera att mysql> är själva prompten och ingen del i kommandot. Oftast vill man ha data i sin tabell. Man kan lägga till data till tabellen genom att skriva:

mysql>**insert into person (fnamn, enamn) values ('Nils', 'Ekman');**

Om man skriver ut vad som nu finns i person-tabellen, ser det ut som exempel 3.5

Exempel 3.5

mysql> **select * from person;**

fnamn	enamn
Nils	Ekman

För att kunna arbeta mot MySQL, från t.ex. PHP, krävs det först att du ansluter dig till databasprodukten i PHP-skriptet. Funktionen som utför detta heter *mysql_connect()*. Som parametrar till funktionen anger man servernamn, användarnamn och lösenord. Dessa parametrar är frivilliga.

Exempel 3.6

```
$resultat = mysql_query("select * from person");
while($rad = mysql_fetch_array($resultat)) {
    echo "Namn: " . $rad[fnamn] . "<br>"; }
```

Exempel 3.6 visar hur man i PHP, enkelt kan hämta data från en tabell i MySQL. Den första raden ställer en fråga mot MySQL-databasen genom *mysql_query()* som är en funktion i PHP. I funktionen skriver man frågan på samma sätt som man skulle skriva i prompten för mysql, förutom att man måste skriva frågan inom citationstecken. Den andra raden läser in varje rad i tabellen till en array, och skriver sedan ut förnamnet på de personer som finns i tabellen. I vårt fall skulle det se ut enligt nedan:

Eftersom det bara fanns ett namn i tabellen skrivs det bara ut en gång. Om det en annan gång skulle finnas fler namn i tabellen, så skulle varje namn som fanns i tabellen skrivas ut en gång.
Det finns många olika databasprogram på marknaden, MySQL är ett. Några andra är Microsoft Access, dBASE och Oracle.

3.2 Val av verktyg för detta projekt

Valet av programmeringsspråk respektive databas grundade sig på flera olika anledningar. Som vi redan har tagit upp så var både kod och dokumentation för det tidigare systemet otillgängligt. Det gjorde att valet av verktyg var mer fritt då vi inte kunde fortsätta utveckla det system som redan fanns. Vi var intresserade av att se hur PHP fungerade, vi hade hört att det skulle vara mycket bra och lätt att arbeta i. Sedan fanns det redan en server på skolan som var dedikerad för PHP och MySQL. Det resulterade i att var att vi valde att programmera i PHP och använda en MySQL databas. Men för uppgiftens skull hade det inte spelat någon roll om vi hade valt någonting annat t.ex. ASP och Access eller CGI och MySQL.
Servern som vi använde oss av var en apache server. Den påstås ha bättre säkerhet än t.ex. Microsofts motsvarighet IIS, men mer om säkerheten på servrar kommer vi inte att gå igenom.

3.3 Prototyp beskrivning

Prototypen består av de mest basala delar som man vill att ett färdigt kursutvärderingssystem skall kunna göra. Prototypen består av ett enkelt användargränssnitt men är inte ett färdigt system, utan det krävs mer arbete. Mest arbete krävs det på säkerheten som vi kommer att gå igenom i kapitel 4.
I prototypen måste läraren identifiera sig mot systemet för att få tillgång till det. Han/hon kan göra sin egen kursutvärdering och kan också ta bort de utvärderingar som han inte vill ha kvar. Läraren kan även se och sammanställa statistik på alla kursutvärderingar innan de blir publicerade. Sammanställningen är till stor del automatisk, förutom sammanställningen för fritext frågor. Han kan bestämma när/om statistiken skall läggas ut på nätet. Systemet gör så

att utvärderingarna sker anonymt, d.v.s. läraren har ingen aning om hur någon röstat. Prototypen kan komma att ligga som grund för ett färdigt kursutvärderingssystem.

3.3.1 Formulär

Eftersom vi under de kommande delkapitlen kommer att gå igenom vår prototyp, så kan det vara bra att gå igenom lite grunder. Vår prototyp är uppbyggd till stor del av formulär. Detta ska vi försöka förklara lite närmare vad det är. Ett formulär skrivs i vanlig HTML-kod, dvs det är ingen PHP-kod. Men man kan komplettera formulären med PHP för att göra den dynamisk. För att kunna förklara det närmare kan man titta på exempel 3.7.

Exempel 3.7

```
<form action="<? echo $PHP_SELF; ?>" method="post">
<input type="radio" name=”variabelnamn” value="R">Radio
<input type="radio" name=”variabelnamn" value="T">Fritext<BR>
<input type="submit" name="submit" value="Lägg till frågan">
</form>
```

I exempel 3.7 syns koden för ett formulär som hanterar radio-knappar. Som man kanske kan se så börjar formuläret där <form>-*tagen* finns, och formuläret kommer att sluta där det finns en </form>-*tag*. Allra först efter <form>-*tagen* står det ***action=”<? echo $PHP_SELF; ?>”***, som talar om var någonstans formuläret ska skickas, dvs till vilken webbadress. I detta fall finns det en funktion i PHP som gör så att man lätt kan skicka formuläret till samma webbadress som man befinner sig på. Man kan välja mellan att låta variablerna i formuläret vara synliga i webbadressfönstret då man skickar formuläret. Det den gör nu är att den inte visar några variabler. Om man vill att variablerna skall synas skall man byta ut ***method=”post”*** mot ***method=”get”***. Sedan står det vilken typ av formulär det skall vara ***input type = ”radio”***, vilket betyder att det skall vara radio knappar. Knapparna sätts också att få ett variabelnamn. Vill man att det bara skall gå att ha en radioknapp markerad så sätter man alla knappar att få samma variabelnamn. När en radio-knapp är markerad så får variabeln, till den knapp som är markerad, ett värde. Det är i vårt fall antingen R eller T. Sist så har detta formulär en knapp som gör det möjligt att skicka iväg formuläret. ***input type="submit”*** talar om att det skall vara en knapp. Värdet som tilldelas knappen i exempel 3.7, är det namn som kommer att stå på knappen. Då man klickar på submit-knappen skickas formuläret med dess variabler till den angivna adressen. I figur 3.3 ser vi hur formuläret kommer att se ut. Det är det, inom den markerade rutan (1), som vi gått igenom koden för i exempel 3.7.

Figur 3.3: Formulär

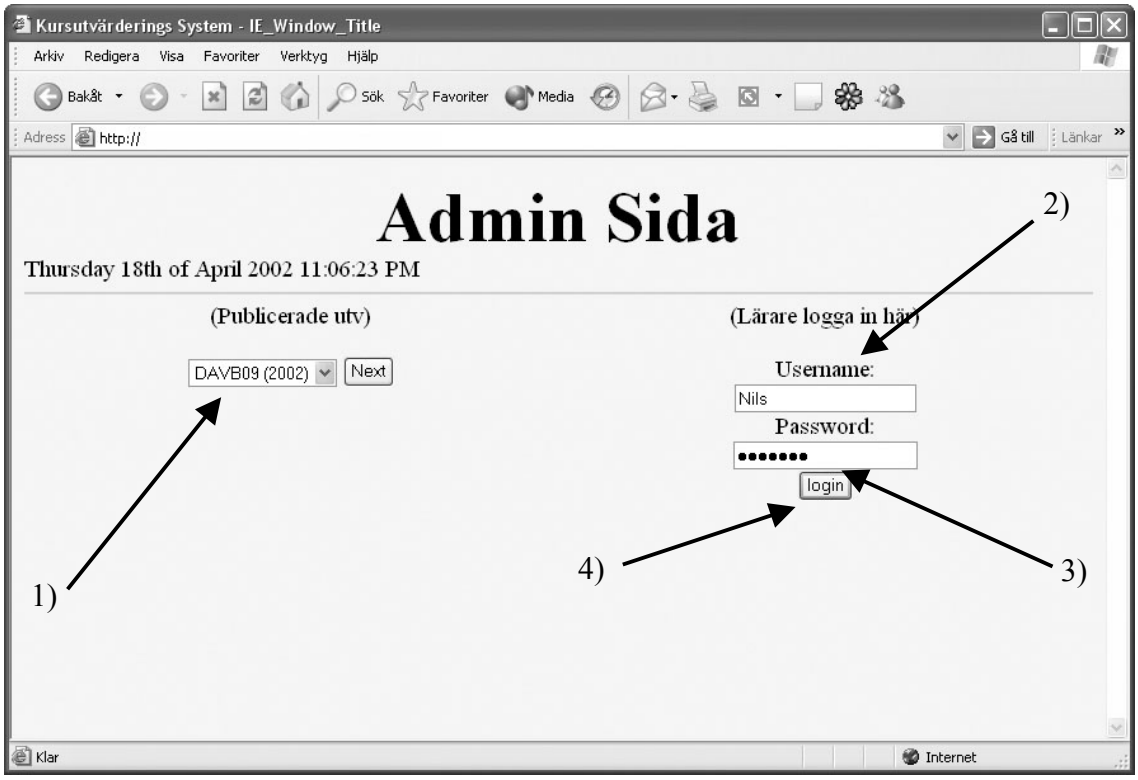
Det finns även andra formulär som vi använder oss av, bl.a. nedrullningslistor och textfält. För att få en nedrullningslista (2) istället, som också finns med i figur 3.3 kan koden se ut enligt exempel 3.8.

Exempel 3.8

```
<form action="<? echo $PHP_SELF; ?>" method="post">
<select name=”fraga”>
<option value=”Fråga1”>Fråga1</option>
</select>
<input type="submit" name="submit" value="Ta bort fråga">
</form>
```

3.3.2 Inloggningssida

Prototypen har en inloggningssida som studenten kan gå till för att se statistik på de publicerade utvärderingarna (1). Vem som helst kan se på vilken utvärdering som helst, bara den är publicerad. Läraren kan identifiera sig mot systemet (2) och det gör läraren genom att fylla i sitt användarnamn och lösenord i formuläret.



Figur 3.4: Inloggningssidan

För att skydda lösenordet, syns det bara asterixer i textfältet när man skriver in det (3). Detta löses med html-kod. Man inkluderar raden *type = ”password”* i formuläret för att åstadkomma detta. När läraren trycker på ”login”-knappen (4), så krypteras lösenordet med hjälp av en funktion i PHP som heter crypt(). Crypt()-funktionen returnerar en krypterad sträng, som är krypterad med en standard DES-baserad krypterings algoritm eller någon alternativ algoritm som är tillgänglig hos systemet, t.ex. MD5. Krypteringsalgoritmer kommer vi att ta upp i kapitel 4. Argumenten till funktionen är den sträng som skall krypteras och en frivillig salt-sträng som krypteringsalgoritmen kommer att baseras på. Några operativsystem stödjer mer än en typ av kryptering. Det finns tillfällen då den standard DES-baserade krypteringen är utbytt mot en MD5-baserad krypteringsalgoritm. Vilken typ av kryptering som skall användas beror på *salt*-argumentet. Vid installationstillfället bestämmer PHP vad krypteringsfunktionen skall vara kapabel till att göra och vilka *salter* som skall accepteras för att få tillgång till andra krypteringsalgoritmer. Om inget salt är medskickat, så kommer PHP att generera en, två tecken lång, slumpsträng. Om standard algoritmen är MD5, så kommer PHP istället att skapa ett slumpmässigt MD5-*salt* som är kompatibelt med funktionen. Det krypterade lösenordet jämför sedan med ett krypterat lösenord som är lagrat i databasen tillhörande användarnamnet. Är det fel lösenord eller om användaren inte finns, så kommer det fram en text som beskriver felet.

3.3.2.1 Konto information

Om inloggningen lyckas kommer användaren till en sida där han kan välja att byta sitt lösenord eller gå vidare till huvudsidan. Byte av lösenord går till så att man skriver in sitt gamla lösenord och sedan det nya lösenordet, plus en repetition av det nya lösenordet. Han behöver inte skriva in sitt användarnamn eftersom han redan är inloggad och systemet vet redan vem användaren är. Det gamla lösenordet jämförs med det i databasen samtidigt som det nya lösenordet krypteras och verifieras. Blir någonting fel så kommer det upp en text som beskriver felet.

3.3.2.2 Root användaren

När *root* användaren loggar in på systemet så kommer det fram ytterligare alternativ. Han kan även lägga till och ta bort användare på systemet. Han kan skapa nya konton och ta bort existerande konton. Om han vill ta bort en användare från systemet får han välja från en rullmeny vilken som han vill ta bort (1)

Figur 3.5: Screenshot på webbsidan lägg till / ta bort användare

```
    <form action="<? echo $PHP_SELF ?>" method="post">
    <select name="name">
2)  <? for($i = 0; $i < $antal_anvandare; $i++) {
        if($array[$i] != "root") { ?>
        <option value="<? echo $array[$i] ?>"><? echo $array[$i] ?></option>
```

Figur 3.6: Kod för ta bort användare

För att skydda systemet så kan root kan ta bort vem som helst utom sig själv från systemet. Detta har vi löst enligt ovan. Man hämtar alla användare från databasen och läser in dem till en rullmeny. En av användarna till systemet kommer att vara ”root”, men när det namnet kommer fram så lägger man inte till det i menyn (2).

Om man väljer att gå till huvudsidan så skickas man till en ny webbadress, programmerad i PHP. Skulle man skriva in den nya webbsidans adress direkt i webb-läsaren så kommer det att komma upp en feltext som säger att man inte har loggat in. För att få tillgång huvudsidan eller någon annan sida i systemet så måste man vara inloggad. Eftersom samma webbadress laddas om flera gånger, men med olikt utseende, uppstår det ett problem med *logout*-knappen. Mer om hur webbadressen laddas om och ändrar utseende kommer vi att ta upp senare i detta kapitel. Man skall inte kunna logga ut innan man har loggat in. Detta löste vi med en if()-sats som kollade om man redan var inloggad eller inte.

3.3.3 Huvudsidan

Själva huvudsidan består av ett antal alternativ i en rullmeny (1). Läraren väljer ett alternativ på menyn och klickar på *next* (2). Beroende på vilket alternativ läraren har valt, så sätts en variabel i formuläret till ett visst värde. Värdet skickas med då sidan laddas om.

Figur 3.7: Huvudsidan

Beroende på vilket värde variabeln har när sidan laddas, så anropas olika funktioner. PHP-koden för att anropa olika funktioner kan se ut som exemplet nedan:

Exempel 3.7

```
if(isset($knapp)){
   if($variabel == 1) { funktion1(); }
   if($variabel == 2) { funktion2(); }}
```

Först är det en koll om man har tryckt på en viss knapp. I vårt fall är det *next*-knappen vi talar om. Det är alltså Isset()-funktionen som kollar om man tryckt på knappen. Sedan har vi vår variabel, och beroende på vilket värde variabeln har så uppfyller den olika if-satser. Dessa if-satser gör olika saker. På detta sätt så kan man få samma webbadress att göra olika saker.
Man kan alltid logga ut från systemet om man är inloggad och därför är knappen *Log out* alltid synlig när man är inloggad (3).
Om man har valt det första alternativet på bilden ovan och klickat på *next*, så sätts en variabel till ett visst värde. Detta värde aktiverar en viss *if*-sats då webbsidan laddas om. Webbsidan kommer då att ändra utseende och se ut som Figur 3.7 istället. Det är alltså samma webbadress som till Figur 3.6 fast webbsidan gör en helt annan sak. I detta fall så har man valt att skapa en ny kursutvärdering. Man får skriva in namn på kursen och ange vilket format på frågeställningen man vill att kursutvärderingen skall ha. En *home*-knapp kommer fram för att det ska vara lättare att gå fram och tillbaka i systemet (4). Om man trycker på den så kommer man tillbaka till huvudsidan, vilket i detta fall är samma webbadress men det uppfyller en annan *if*-sats. *Home*-knappen försvinner då man är på huvudsidan, vilket vi har löst på liknande sätt som med *logout*-knappen vid loginsidan.

Figur 3.8: Skapa ny Kursutvärdering sida

För att läraren inte skall behöva skriva en helt egen utvärdering från början så finns det en klar utvärdering som läraren kan välja (5). Användaren kan välja mellan tre olika alternativ, default, default plus och egen utvärdering. Dessa val visualiseras av systemet med hjälp av radioknappar. Eftersom man bara vill att läraren skall kunna välja ett av alternativen så har vi löst det genom att sätta alla knappar till att ha samma namn. Exempel 3.8 visar hur vi löst det med knapparna.

Exempel 3.8

```
<form action="andra.php" method="post">
<input type="radio" name="eval_type" value="default" checked="checked">
<input type="radio" name="eval_type" value="defaultplus">
<input type="radio" name="eval_type" value="own">
</form>
```

Vill man även att ett av alternativen skall vara förvalt så skriver man dit *checked="checked"* efter det alternativet som man vill ha förvalt, vilket också visas i exempel 3.8. När användaren är klar med vad kursen skall heta och vilken frågeställning han vill att kursutvärderingen skall ha, så klickar han på *skapa* (6). Då anropas en annan webbadress och variabler skickas med till den sidan, beroende på vilket alternativ användaren har valt. Eftersom ett alternativ alltid kommer att vara förvalt så finns det ingen risk att inget alternativ kommer att vara valt.

3.3.4 Ändra i kursutvärdering

Om man har valt att Ändra i en kursutvärdering eller skapa en ny utvärdering från huvudsidan så hamnar man på denna sida. Här kan man lägga till (1) eller ta bort frågor (2). När man lägger till en fråga får man välja om man vill skapa en radioknappsfråga eller en fritextfråga. Om man väljer radioknappsfråga så kan frågan se ut som *Fråga 1* i figur 3.5. Väljer man istället en fritextfråga, visas en textruta istället för de fem radioknapparna. Under hela tiden som man redigerar får man se hur utvärderingen kommer att se ut, genom en förhandsgrandskning (3). Förhandsgranskningen är uppbyggd på liknande sätt som studenternas utvärderingssida. Det finns även en *klar*-knapp (4) som man ska trycka på när man är klar med utvärderingen.

Figur 3.9: Ändra i utvärdering

Sidan tar emot en variabel som talar om vilken webbsida som man kommer ifrån, ”skapa ny utvärdering” eller ”ändra i befintlig utvärdering” (*se Kapitel 3.3.3 huvudsidan*). Om man kommer från ”ändra i utvärdering” så hämtas den utvärdering man vill ändra på, sedan är det bara att lägga till eller ta bort frågor. Om man skapar en ny utvärdering så finns tre olika lägen som man kan skapa en utvärdering från, default , default plus eller egen utvärdering.

- *Default*

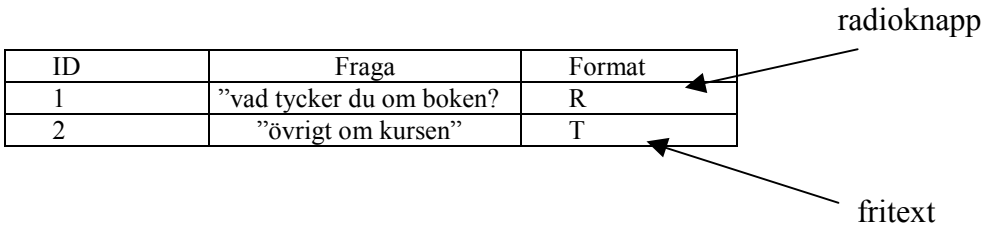
  I default så finns redan en klar utvärdering som ska används, man får då inte lägga till eller ta bort frågor. Endast förhandsgranskning och *klar*-knappen kommer att synas.
- *Default-plus*

  Här används också den färdiga utvärderingen, men här är det tillåtet att lägga till eller ta bort frågor.
- *Egen utvärdering*

  Här får man bygga upp sin utvärdering från början. Inga frågor finns i förväg när man valt detta alternativ.

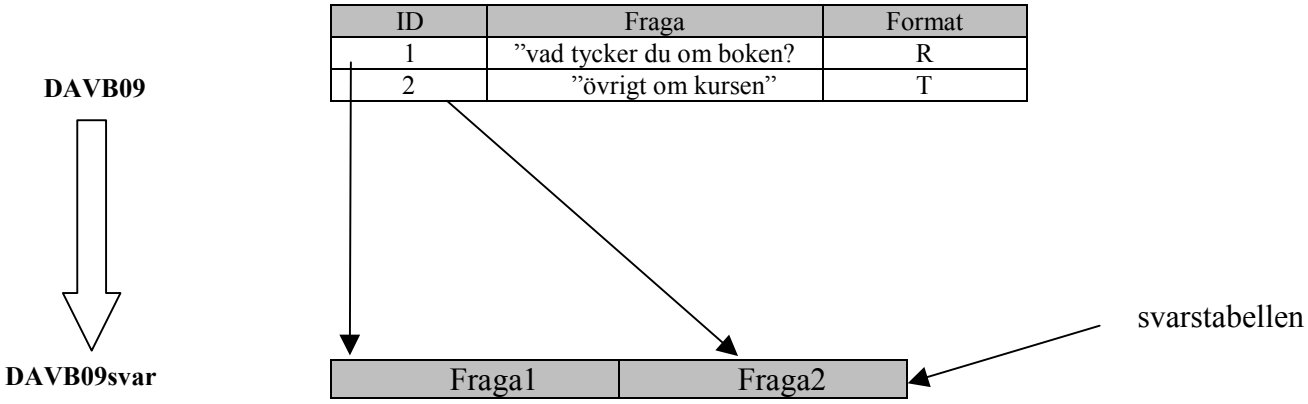
I alla tre fall så kommer en tabell med tre kolumner att skapas, Id, fråga och format. Format informerar om frågan är en fritextfråga eller en fråga med radioknappar. Om det är en fritextfråga så skrivs ett ”T” in i formatkolumnen, om det en fråga med radioknappar så skrivs ett ”R” in istället. Tabellens namn är den samma som kurskoden. Om man valt att skapa från

default eller default-plus så kommer alla defaultfrågor att kopieras till den nya tabellen, annars så skapas en tom tabell.



Figur 3.10: Frågetabell

När man är klar med utvärderingen och trycker på klar-knappen så skapas en svarstabell. I svarstabellen kommer alla svar från studenternas utvärdering att läggas in. Svarstabellen får kursutvärderingens namn med ordet ”svar” tillagt på slutet, om det t.ex. är till ”DAVB09” så kommer tabellen att få namnet ”DAVB09svar”. Tabellen kommer att innehålla lika många kolumner som det finns frågor så man kan lägga in alla svar. Figur 3.6 visar hur en svarstabell skapas när man trycker på klar-knappen.



Figur 3.11: Svarstabell skapas

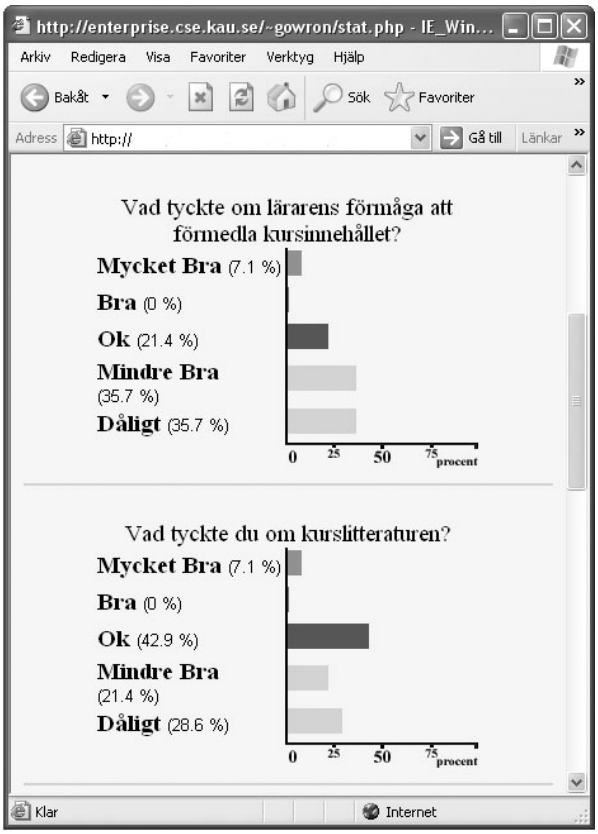
Exempel 3.9 visar ett kodstycke, ur en funktion, som skapar en ny svarstabell. Funktionen tar emot variabeln *tabell*, som i detta fall kommer att innehålla ”DAVB09”. På första raden ställer man en fråga mot databasen som returnerar ett resultat till variabeln *result*. *For*-loopen itererar så många varv som det finns rader i frågetabellen, dvs tabellen som ligger i variabeln *result*. Antalet rader får man fram med *mysql_num_rows()*. Loopen är till för att skapa en del till en *sql*-fråga. Om det t.ex. finns två rader i frågetabellen kommer delen att bli ”***fraga1 text, fraga2 text”***, där *text* är en datatyp i MySQL. Hela *sql*-frågan för att skapa en svarstabell, när frågetabellen ser ut som i figur 3.10, kommer att se ut som ”***create table DAVB09svar (fraga1 text, fraga2 text)***”

Exempel 3.9

```
$result = mysql_query(”select * from $tabell");
for($i = 1; $i < mysql_num_rows($result) + 1; $i++){
        if($i > 1){
                $text = $text . ",";}
        $text = $text . "fraga" . "$i" . " text";}
$sql = "create table $tabell" . "svar (" . $text . ")";
mysql_query($sql);
```

3.3.5 Statistiksidan

På statistiksidan finns diagram över flervalssvaren och en tabell med fritextsvaren. För att se på statistik så får man välja endera fritext eller flerval. Figuren 3.6 visar hur det ser ut om man har valt att kolla på statistik över flervalsfrågorna.



Figur 3.12: Statistik över flerval

3.3.5.1 Flerval

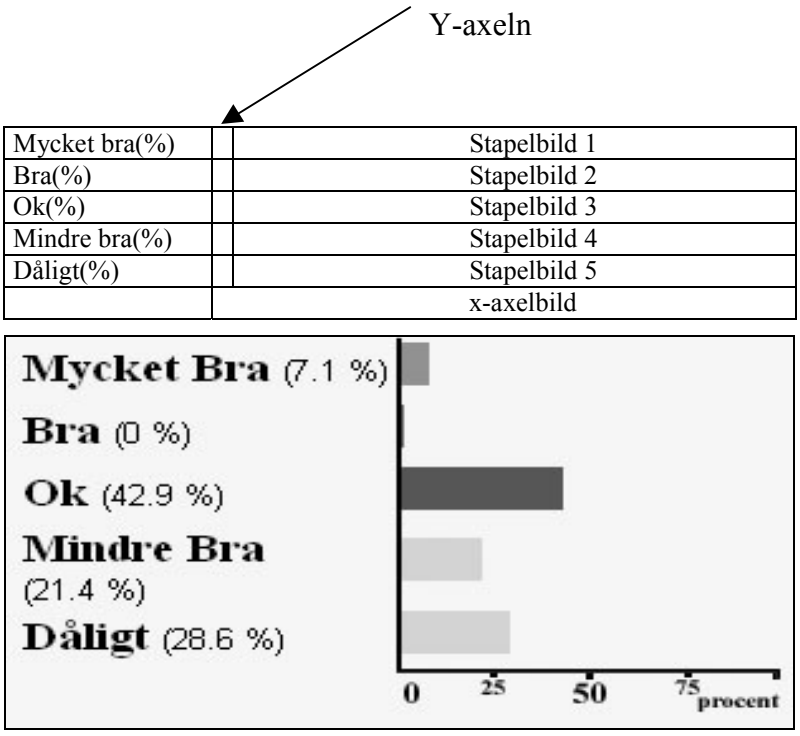
Alla svar räknas ut i procent för varje fråga. Procenten för varje svar läggs in i en array och skickas med till en utskriftsfunktion. Utskriftsfunktionen består av en tabell som innefattar text och bilder. Tabellen har sex rader, de fem första raderna har tre kolumner av olika storlek, den sista raden består av två kolumner. Den första kolumnen i de fem första raderna kommer

att innehålla namnet på svarsalternativet och hur stor del av svarsalternativet som studenterna röstat på. Den andra kolumnen i de fem första raderna kommer att innehålla endast en svart färg, den kommer att representera y-axeln i diagrammet. Den tredje kolumen kommer att innehålla bilder som blir vågräta staplar till varje svar. Hur stora staplarna kommer att bli bestäms av attributet width som får det värde som ligger i procentarrayen. Om värdet i arrayen är 50% så kommer bilden att spänna upp 50% av kolumnstorleken.

Exempel 3.9

```
<img src="stat.gif" height="20" width="<?echo $proc[1] . "%"?>"
hspace="0" vspace="0" border="0">
```

Den andra kolumnen på rad sex kommer att innehålla en bild som representerar x-axeln i diagrammet.



Figur 3.13: Uppbyggnad av diagram

3.3.5.2 Fritext

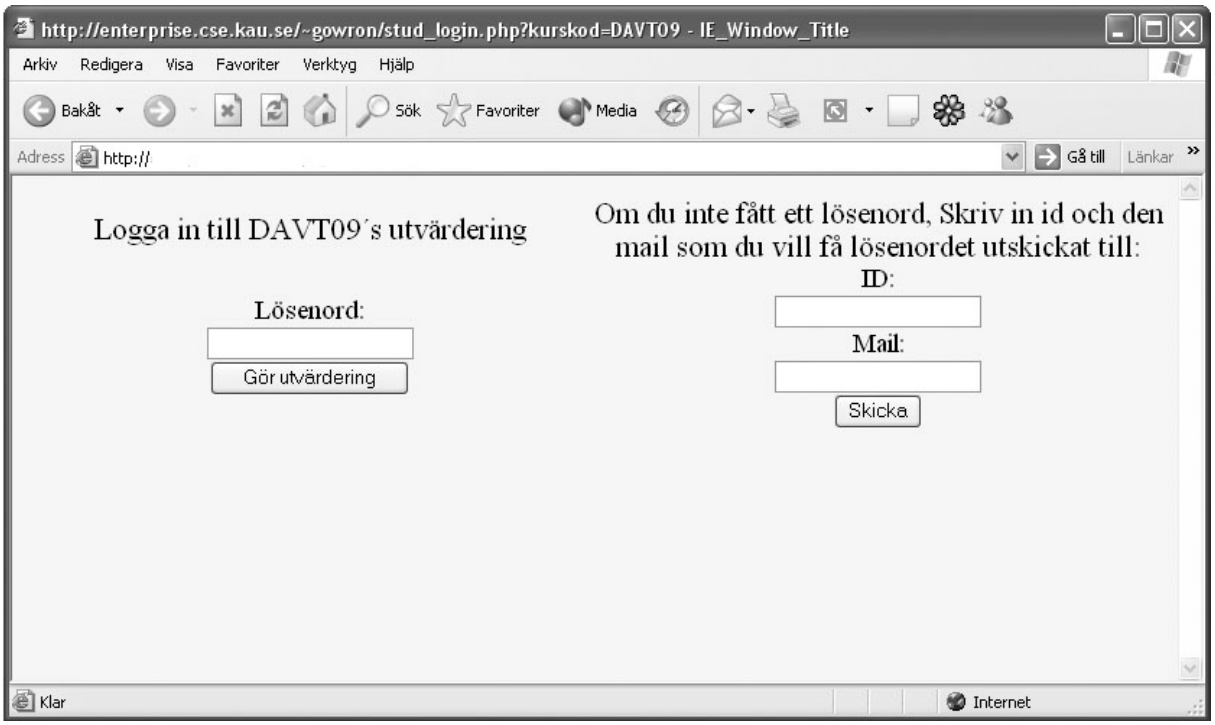
Alla fritextsfrågor visas här. Under varje fråga finns en tabell med studenternas svar till frågan. Längst ner på sidan finns en textruta. Rutan är till för att skriva in en sammanfattning från fritextsvaren, sammanfattningen kommer sedan att visas istället för alla studenternas

fritextsvar. Om det redan finns en sammanfattning sparad när man besöker denna sida så visas den sparade sammanfattningen i textrutan. Efter man har tryckt på ”spara sammanfattning” så sparas sammanfattningen, samtidigt som man får en förfrågan om man vill fortsätta att kolla på statistik eller gå tillbaka till huvudsidan.

Figur 3.14: Lista med alla fritextsvar

3.3.6 Verifieringssidan för studenterna

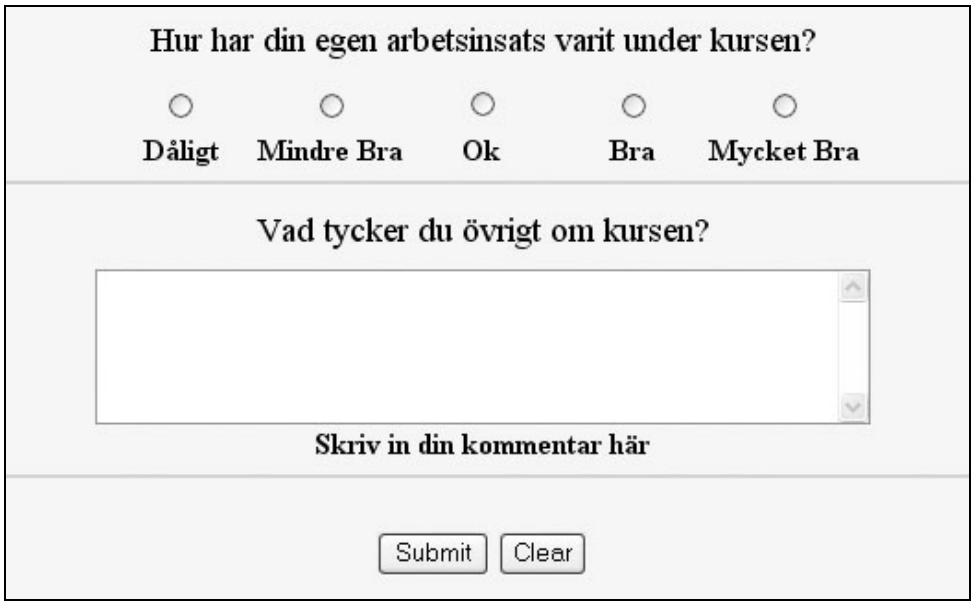
När studenterna ska göra en kursutvärdering så måste de verifiera sig. Endast de studenter som går kursen skall kunna utvärdera den. För att komma till denna sida så skall det finnas en länk från kursens hemsida. I länken så finns en variabel med kurskoden så att studenten kommer till verifieringssidan för rätt kurs. När studenten har kommit hit så skall de ange sitt lösenord, vilket de erhållit från biljettsystemet (se *Kapitel 4.8.2 enkelt biljettsystem*). Enda sättet att få ett giltigt lösenord är att skriva in sitt id, som man har blivit tilldelad under kursens gång. Om man skriver in rätt id så ska lösenordet skickas till den mail-adress som man anger. För att förhindra felstavning av mail så får studenten en förfrågan om mail-adressen stämmer innan lösenordet skickas iväg.



Figur 3.15: studenternas verifieringssida

3.3.7 Formuläret för studenten

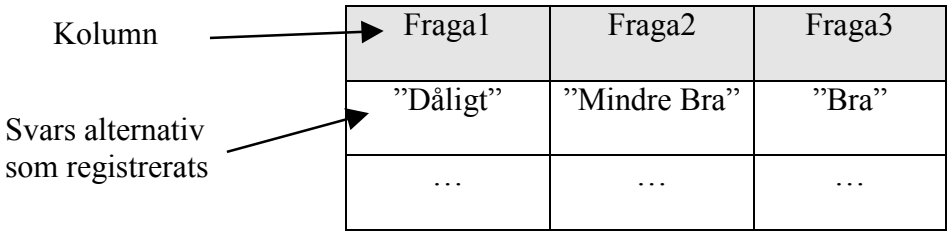
Själva sidan för utvärderingen är inte så konstig. Den består av frågor med antingen radio-knappar eller fritext ruta som svars alternativ.



Figur 3.16: Studentens formulär

Frågorna i formuläret kommer att variera efter hur många frågor som kursutvärderingen innehåller. Dessa frågor finns lagrade i en tabell i databasen och kommer att hämtas därifrån

och skrivas ut i formuläret. Hela sidan kommer att bli ett dynamiskt formulär. För att man bara ska kunna klicka i ett alternativ på varje fråga, så måste man ge alla radioknappar tillhörande en fråga, samma variabelnamn. Men för att det skall fungera att skriva ut flera frågor med svarsalternativ så måste variabelnamnet till radioknapparna ändras för varje ny fråga som skrivs ut. Detta måste ske dynamiskt eftersom man inte vet hur många frågor som det finns lagrade i databasen. Det har vi löst genom att läsa in alla kolumnnamn från svarstabellen i en array. Eftersom alla kolumnnamn är unika och är lika många till antalet som frågor, så passar det bra. Den första variabeln kommer alltså att få namnet ”Fraga1” och den andra ”Fraga2” osv.



Figur 3.17: Svarstabell i databasen

När man ska lägga till en ifylld utvärdering i databasen så uppstod det ett problem med att ta fram värdet på de variabler som finns i arrayen. Ett sätt att lösa det var att använda sig av HTTP_POST_VARS, som är en fördefinierad variabel i PHP. Genom att skriva *$HTTP_POST_VARS["$namnet_på_variabeln"]* så kan man få fram vilket värde som variabeln har. Och på så sätt lägga till det i svaret som man skickar till databasen. Det kontrolleras även att studenten har svarat på alla de frågor som krävs ett svar på. Har studenten inte fyllt i alla frågor och tryckt på submit-knappen så kommer det upp en feltext som beskriver felet. Det finns även en *clear*-knapp om man är missnöjd med vad man har fyllt i och vill göra om.

3.4 Databasbeskrivning

Databasens grund består av fyra tabeller, kursutv, passwd, student och ticket. Om en ny kursutvärdering skapas så läggs, kursens namn och när den skapades, till i kursutv-tabellen. Sammanfattningen som lärarna har summerat för kursen och om statistiken är publicerad sparas även i kursutv-tabellen. Passwd-tabellen innehåller alla användare och dess krypterade lösenord. Student-tabellen innehåller student-id till alla kurser och om dom har hämtat sin biljett. Student-id är det id de får under kursens gång för att kunna göra utvärderingen för

kursen. Ticket-tabellen innehåller alla giltiga biljetter till varje kurs (se *Kapitel 4.8.2 enkelt biljettsystem*).

kursutv

Course code	Published	Date	Summary
DAVB09	YES	2001	”Kursen ….”
DAVC09	NO	2002	

passwd

User	Password
Root	Zxvs54Tsk2
Alejandro	Oid3fkja2gs

student

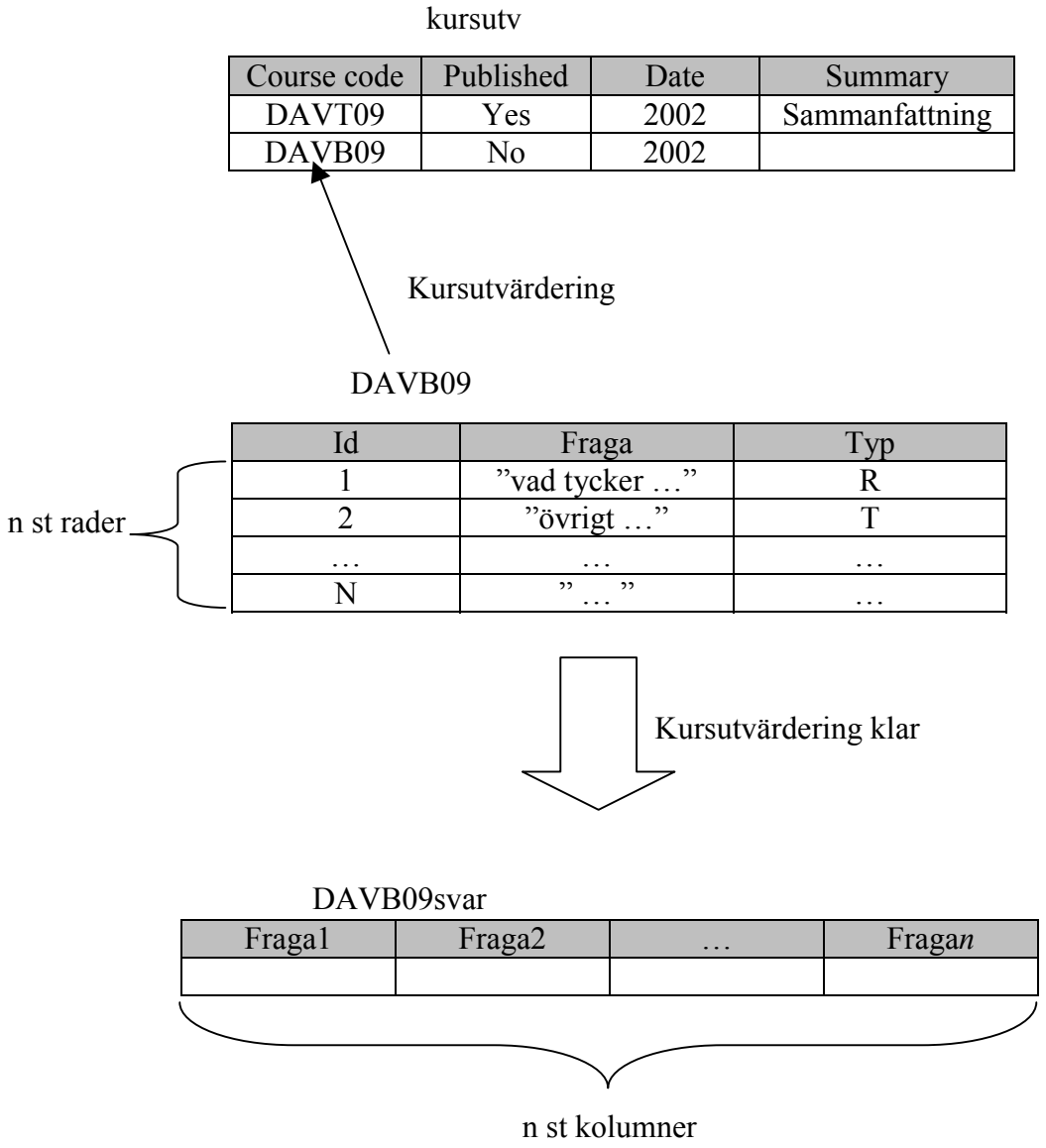
Id	Ticket	Course code
Di9simo	23523523562	DAVB09
Di9dang	63562727232	DAVB09

ticket

Ticket	Course code
23523523562	DAVB09
63562727232	DAVB09

Figur 3.18: Databasens fyra grundtabeller

För varje ny kursutvärdering som skapas så skapas också en ny tabell. Den nya tabellen får samma namn som kurskoden har. I den tabellen lagras alla frågor och vilken typ av svarsalternativ varje fråga skall ha, d.v.s. flerval eller fritext. När kursutvärderingen är klar för användning så skapas en tabell som alla svar ska ligga i. Den tabellen kommer att få kurskodens namn med ”svar” på slutet, se figur 3.19. Den nya tabellen kommer att innehålla lika många kolumner som det finns frågor så att alla svar kan sparas.



Figur 3.19: Nya tabeller som skapas vid ny utvärdering

3.5 Sammanfattning

Det finns ett antal språk på marknaden idag, som man kan använda för att producera dynamiska webbsidor med, bl.a. PHP och ASP. Det finns även ett antal olika databaser att välja mellan. Vi valde PHP som programspråk och MySQL som databas till vår prototyp. Valet visade sig vara ett bra val för att de bl.a. kompletterade varann på ett väldigt bra sätt genom att det var lätt att kommunicera mellan programspråket och databasen. PHP hade inbyggda funktioner för MySQL, vilket gjorde att kommunikationen mellan språket och databasen blev enkel och överskådlig. Dynamisk kod resulterar allt som oftast i att de antal rader man behöver skriva minskar drastiskt, än om man skulle skriva kod statiskt. Om man

programmerar en webbsida på ett statiskt sätt, t.ex. bara i HTML-kod, kommer sidan att se ut på samma sätt varje gång som sidan laddas. En dynamisk sida däremot, kan göra olika saker vid olika tillfällen, beroende på vilka värden som är satta etc.

Det vi ville visa med vår prototyp var att röstning, utvärdering, tentor, mm, kan på ett enkelt sätt göras över Internet. Vi har tagit fram ett enkelt användargränssnitt och visat att det är enkelt att producera grunden för sådana system. Vi har även visat att det är väldigt enkelt för användaren att fylla i utvärderingen.

Man kan t.ex. tänka sig ett system som tillåter människor att lägga sin röst till riksdagsvalet på Internet. I en artikel i [3] *Guardian Weekly* skrivs det, ”Britain will become the first country in the world to use internet for voting…”. För ett sådant system krävs det bl.a. att säkerheten är exceptionellt bra, vilket den inte har hunnits med att göra till vår prototyp. Vår prototyp visar hur enkelt det är att skapa dynamiska sidor som kan ligga till grunden för ett sådant valsystem, men då krävs det att man lägger på bättre säkerhet. Mer om säkerhet kommer i nästa kapitel.

4 Säkerhetsaspekter

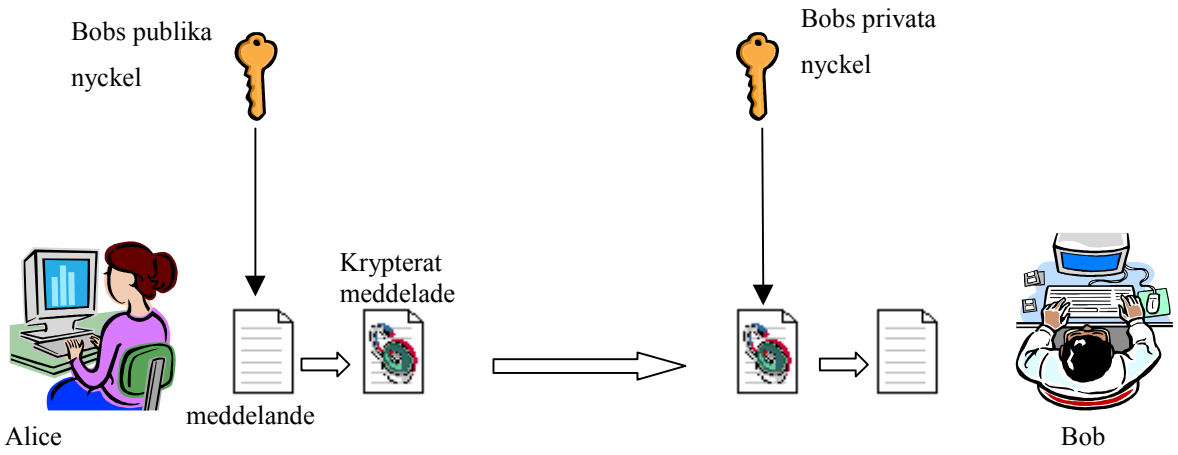
Online-röstning kommer aldrig att kunna införas om det inte finns något sätt att förhindra fusk och skydda varje individs enskildhet. Det finns vissa krav för att få detta att fungera:

1. **Endast behöriga röstare får rösta.**
2. **Ingen kan rösta mer än en gång.**
3. **Ingen kan ta reda på vad någon annan har röstat.**
4. **Ingen kan duplicera någon annans röst.**
5. **Ingen kan ändra någon annans röst utan att bli upptäckt.**
6. **Varje röstare kan vara säker på att deras röst har blivit räknad.**

I ett kursutvärderingssystem på en skola är inte säkerheten lika viktig som t.ex. i ett regeringsval men de har i stort sett samma struktur. I detta kapitel ska vi gå igenom vad som kan göras för att uppfylla dessa krav och om det går. Vi kommer att visa hur symmetrisk och asymmetrisk kryptering fungerar. Vi ska även se hur man använder någon av dessa för att skapa digitala och blinda signaturer. Vi kommer att gå igenom vilka krav som uppfylls med blinda signaturer och med ett enkelt biljett system.

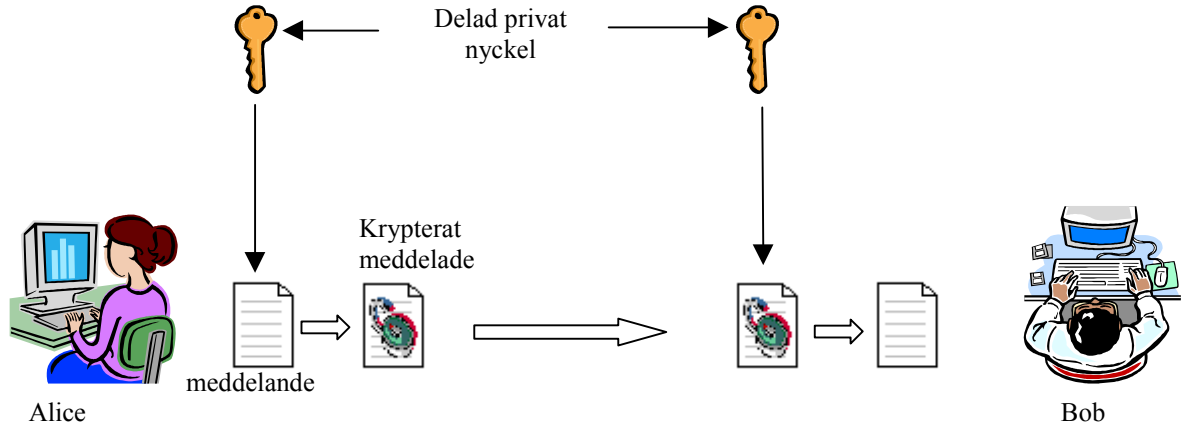
4.1 Symmetrisk och asymmetrisk kryptering

Det finns två former av nyckelbaserade algoritmer för kryptering. Asymmetrisk och symmetrisk. I asymmetrisk kryptering använder man sig av två nycklar, en publik och en privat. Den publika nyckeln för att kryptera ett meddelande och den privata för att dekryptera. Den publika nyckeln är tillgänglig för ”alla i världen” och den privata nyckeln är endast tillgänglig för mottagaren. I figur 4.1 krypterar Alice ett meddelande hon vill skicka till Bob med Bobs publika nyckel. Bob tar emot det krypterade meddelandet och dekrypterar det med hans privata nyckel.



Figur 4.1: Asymmetrisk kryptering

I symmetrisk kryptering måste avsändare och mottagare ha tillgång till samma nyckel. Symmetrisk kryptering är mycket snabbare än asymmetrisk kryptering, men det är inte bra att nyckeln måste göras tillgänglig för minst två parter. Om två parter skall dela nyckel måste det finnas ett sätt att bestämma vilken nyckel de skall använda. Om man skickar ett meddelande över internet kan folk ta reda på nyckeln om de avlyssnar trafiken. Det är alltså ett distributionsproblem. I figur 4.2 krypterar Alice det meddelande hon vill skicka med hennes och Bobs delade privata nyckel. Bob använder samma nyckel för att dekryptera meddelandet.



Figur 4.2: Symmetrisk kryptering

4.2 Krypteringsalgoritmer

För att kryptera används olika krypteringsalgoritemer, några av dessa går vi igenom här.

4.2.1 RSA

RSA har fått sitt namn från sina tre uppfinnare Ron Rivest, Adi Shamir och Leonard Adleman. RSA har sedan det skapats stått emot år av omfattande kryptoanalys. Trots att kryptoanalytikerna aldrig kunde bevisa RSAs säkerhet, så kan man säga att det är en pålitlig algoritm. RSAs säkerhet bygger på svårheten i att faktorisera stora tal. De publika och privata nycklarna är funktioner av ett par av stora primtal (100-200 siffror eller t.o.m. större).

För att RSA ska fungera behöver vi tre tal (e, d, n). Där e är krypteringsnyckeln (publika), d är dekrypteringsnyckeln (privata) och n är produkten av två stora primtal. För att kryptera en text använder man formeln:

$$\mathbf{C = m^e \bmod n}$$

och

$$\mathbf{m = c^d \bmod n}$$

för att dekryptera.

RSA kan också användas för att ge en digital signatur. Då använder man sin privata nyckel för att signera och den publika för att verifiera.

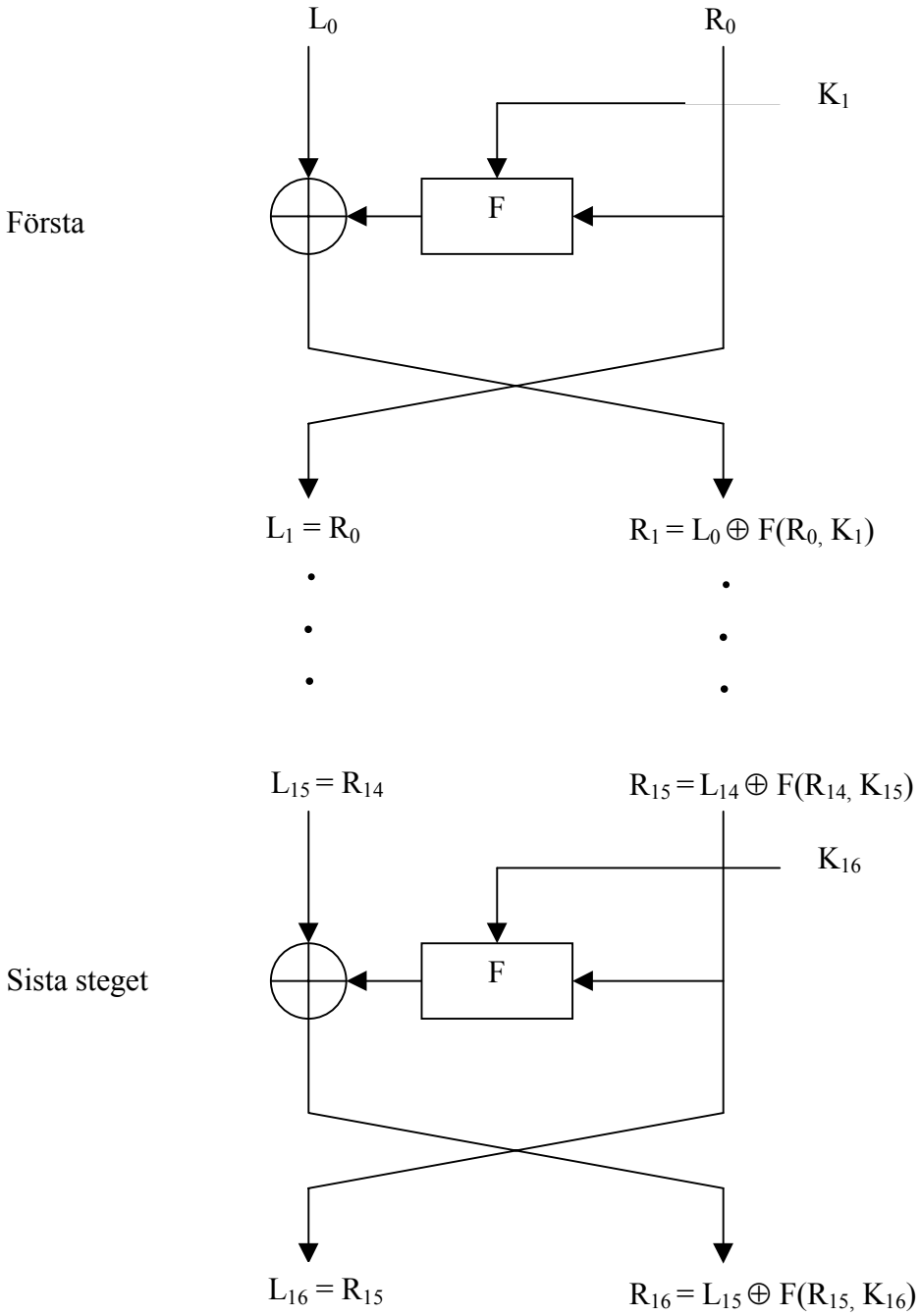
4.2.2 MD5

MD5 är en förbättrad version av MD4. MD4 är en envägs hashfunktion som är utvecklad av Ron Rivest. MD står för Message Digest, som betyder att man bryter ned ett meddelande till en mindre bit.

MD5-algoritmen tar ett meddelande av godtycklig längd som input och producerar ett 128-bitars ”fingeravtryck” eller ”message digest” som output. Algoritmen är avsedd för applikationer med digitala signaturer, där en stor fil måste bli ”komprimerad” på ett säkert sätt. Detta sker innan man krypterar med en privat nyckel i ett asymmetriskt krypto som t ex RSA

4.2.3 DES

DES, Data Encryption Standard, är ett block orienterat krypto som krypterar block av 64 bit. Krypteringen är kontrollerad av en nyckel på 56 bitar, som genererar 16 stycken 48-bitars delnycklar. Krypteringen sker i 16 steg, i varje steg används en delnyckel för att kryptera.

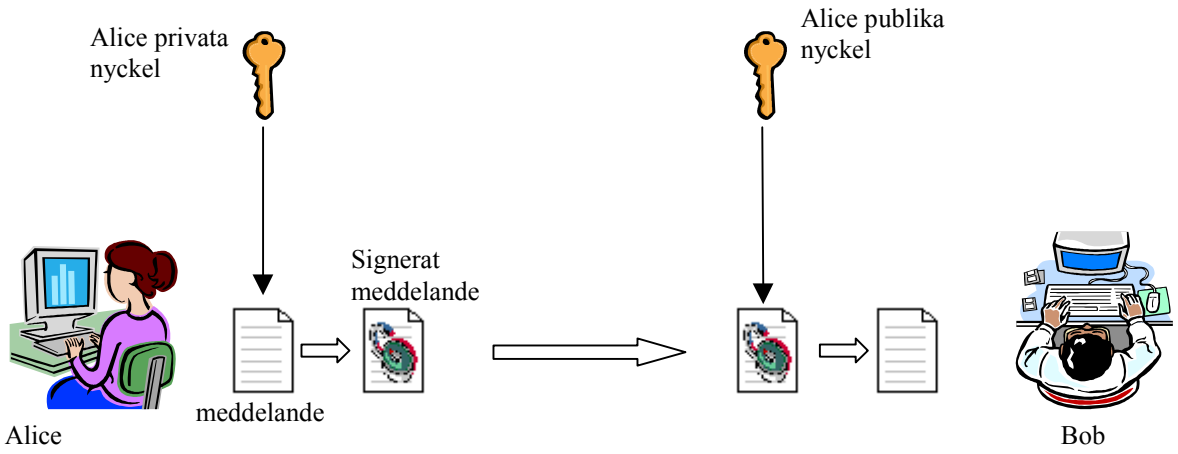


Figur 4.3: DES-kryptering

Figur 4.3 visar det första av de 16 stycken stegen för DES-kyptering, där L_0 och R_0 tillsammans utgör klartexten.

4.3 Digital Signatur

En digital signatur omvandlar meddelandet som är signerat så att vem som helst som läser det kan vara säker på vem det är som skrivit det. Dessa signaturer använder sig av en privat nyckel, som används för att signera meddelanden och en publik nyckel som används för att verifiera dem. Ett meddelande som är signerat med den privata nyckeln kan endast verifieras med den publika nyckeln. Alice vill skicka ett signerat meddelande, *m* till Bob, hon använder sin privata dekrypterings nyckel d_A, för att beräkna $d_A(m)$. Det kan verka konstigt att man använder sig av en dekrypteringsalgoritm på ett dokument som inte har blivit krypterat. Men en dekryptering är inget annat än en matematisk operation. Alice skickar det digitalt signerade dokumentet $d_A(m)$ till Bob. Bob använder sig av Alices publika nyckel, e_A, för att vara säker på att hon har skickat det. De bäst kända metoderna för att producera förfalskade signaturer skulle kräva många år, även om man skulle använda datorer som är mycket snabbare än dagens datorer. Figuren 4.3 visar hur Alice signerar ett meddelande och hur sedan Bob verifierar att signaturen kommer från Alice.



Figur 4.4: Digitala signaturer

4.4 Blinda Signaturer

Blinda signaturer är en utökning av digitala signaturer. Ibland vill man att det ska gå att signera ett dokument utan att veta exakt vad som står i det, men ändå ha en aning om vad det är som står i det. Det finns två varianter av blinda signaturer, den ena varianten går ut på att man inte har någon som helst aning om vad som står i det dokument som man signerar.

1) Alice tar sitt dokument och multiplicerar det med ett slumptal. Detta slumptal kallas för en ”blindande” faktor.
2) Alice skickar det ”blindande” dokumentet till Bob.
3) Bob signerar det ”blindande” dokumentet.
4) Alice dividerar ut den ”blindande” faktorn, och får original dokumentet signerat av Bob

Eftersom detta protokoll är helt ”blint” kan Alice ha fått Bob att signera vad som helst, t.ex. att Bob är skyldig Alice 2 miljoner kronor. Bob har ingen som helst aning om vad det är som han signerar, så detta protokoll är inte så användbart.
Den andra varianten är att Bob kan veta vad han signerar, men ändå inte. Han kan med största sannolikhet veta vad det är han signerar. Detta gå ut på att:

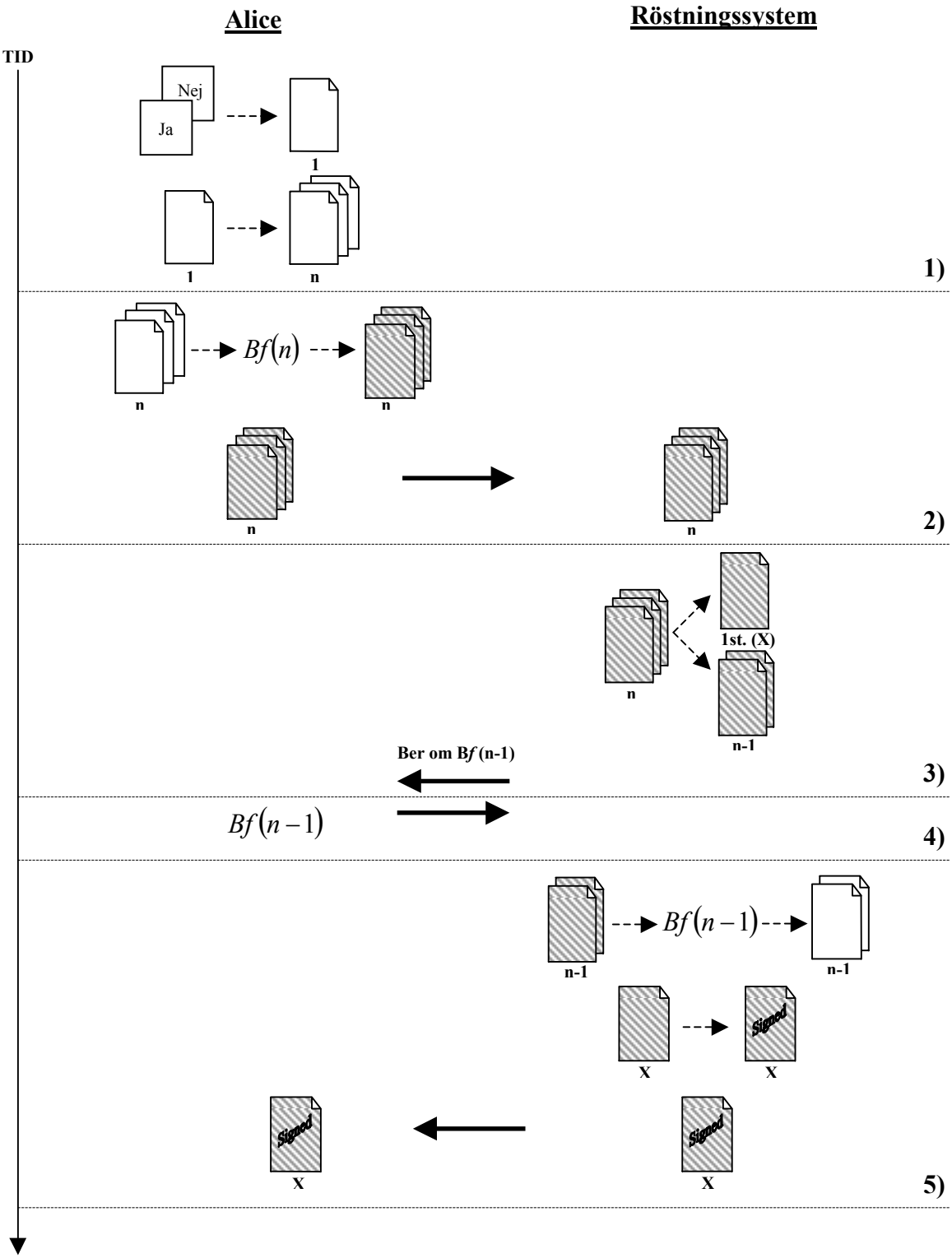
1) Alice förbereder *n* dokument
2) Alice ”blindar” varje dokument med olika ”blindande” faktorer.
3) Alice skickar de *n* ”blindande” dokumenten till Bob.
4) Bob väljer ut *n-1* dokument slumpmässigt och ber Alice att skicka de ”blindande” faktorerna för varje dokument.
5) Alice skickar de ”blindande” faktorerna.
6) Bob öppnar de *n-1* dokumenten och kollar att de är godtagbara och inte innehåller konstigheter.
7) Bob signerar det resterande dokumentet och skickar det till Alice.
8) Alice har nu fått sitt dokument signerat.

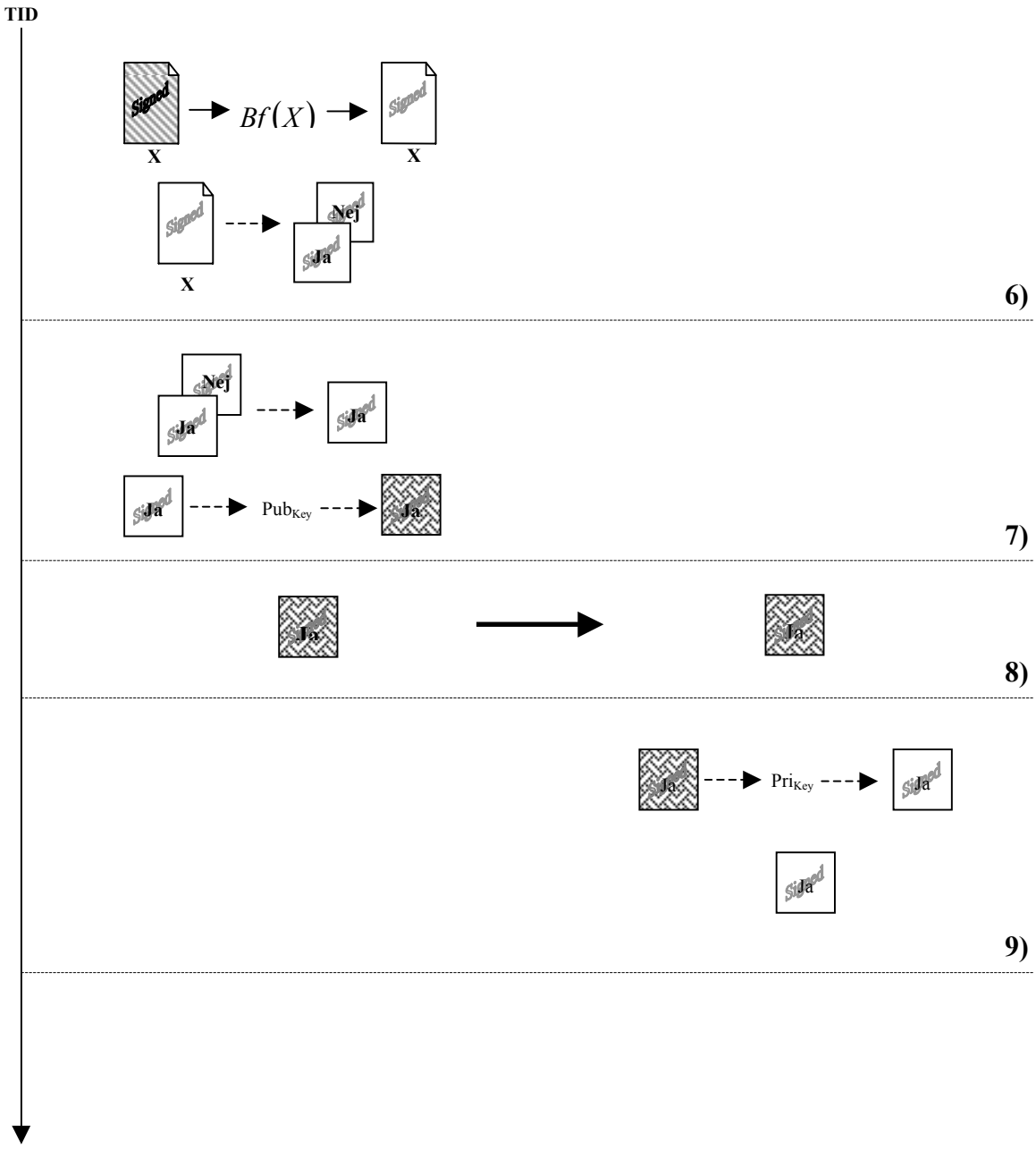
Detta protokoll är säkert gentemot att Alice försöker fuska. För att Alice skall kunna fuska krävs det att hon vet vilket dokument som Bob inte kommer att undersöka. Oddset för att det ska hända är väldigt litet. Bob vet detta och kan därför lugnt signera dokumentet. Det dokument som signeras erhåller alla egenskaper om anonymitet. Det finns ett trick som gör Alices chans att fuska än mindre. I steg (4) väljer Bob ut *n/2* dokument att kolla på, Alice skickar de tillhörande ”blindande” faktorerna i steg (5). I steg (7), kombinerar Bob ihop alla osignerade dokument till ett stort mega-dokument, och signerar mega-dokumentet. I steg (8), Alice tar bort alla ”bindande” faktorer. Bobs signatur är giltig om och endast om mega–dokumentet består av *n/2* identiska dokument. För att fuska behöver Alice gissa exakt vilken delmängd Bob kommer att pröva, oddsen är mycket mindre än det tidigare exemplet.

4.4.1 Rösta med blinda signaturer

Man måste skilja på den som röstar och själva rösten, för att tillhandahålla anonymitet. Samtidigt som man har verifiering av röstaren. Detta tar blinda signaturer hand om.

1) Varje röstare genererar *n* meddelande uppsättningar, varje uppsättning innehåller en giltig röst med varje möjligt svar. (Om rösten är ett ja eller nej svar så innehåller varje svar två röster, en för ”ja” och en för ”nej”). Varje meddelande innehåller också ett identifikationsnummer som är ett slumptal, stort nog att undvika dubbletter med andra röstare.
2) Varje röstare ”blindar” alla sina meddelanden och skickar iväg dom till röstningsystemet.
3) Röstningssystemet ber om de *n-1* ”blindande” faktorerna till de *n-1* meddelande uppsättningar de vill öppna.
4) Röstaren skickar de *n-1* faktorerna till röstningssystemet.
5) Röstningssystemet kollar sin databas för att försäkra sig om att röstaren inte har skickat in sina ”blindande” röster för signatur tidigare. Den öppnar alla *n-1* uppsättningar för att kolla att dom är rätt utformade. Sedan signeras varje uppsättning i det ”blinda” meddelandet, skickas tillbaka till röstaren och sparar röstarens namn i deras databas.
6) Röstaren gör så att uppsättningen inte är ”blind” längre, vilket leder till att det återstår en uppsättning av röster signerade av röstningssystemet. (Dessa röster är signerade men inte krypterade, så röstaren kan enkelt se vilka röster som är ”ja” och ”nej”.)
7) Röstaren väljer en av rösterna och krypterar den med röstningssystemets publika nyckel.
8) Röstaren skickar in sin röst.
9) Röstningssystemet dekrypterar rösten och kollar signaturen. Sedan kollar systemet om serienumret redan finns i databasen. Om systemet inte hittar serienumret i databasen så sparas serienumret och rösten. Systemet publicerar resultatet av röstningen och hur varje väljare har röstat, d.v.s. serienumret tillsammans med den tillhörande rösten. Man kommer fortfarande inte veta vem som röstat vad, eftersom ingen känner till någon annans serienummer.


Alice
Röstningssystem
TID
Nej
Ja
1
1
n
1)
n
Bf(n)
n
n
n
2)
1st. (X)
n
n-1
Ber om Bf (n-1)
3)
Bf(n−1)
4)
n-1
Bf(n−1)
n-1
Signed
X
X
Signed
X
X
5)



Figur 4.5: Röstning med blinda signaturer

4.5 Prototypbeskrivning av säkerhet

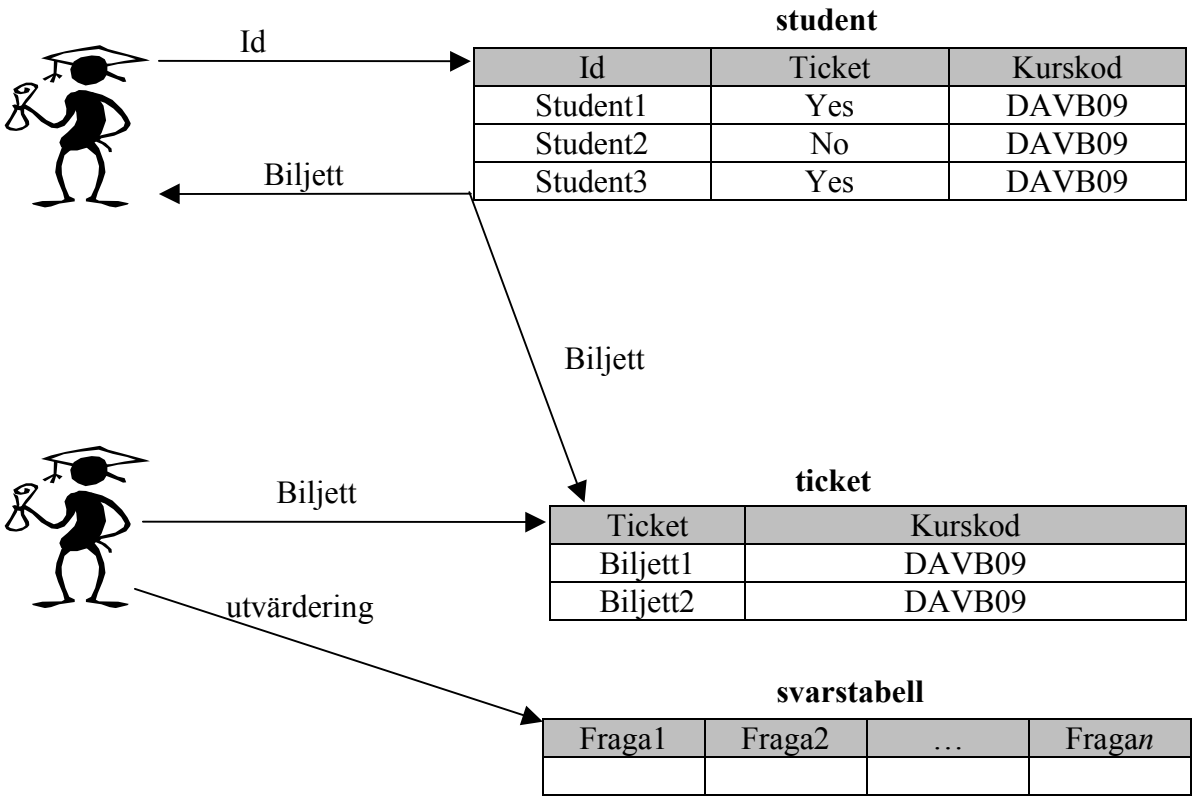
I det här delkapitlet kommer vi att se på den säkerhet som implementerades på prototypen.

4.5.1 Lösenord/inloggning

För att komma in i systemet krävs ett användarnamn och lösenord (se kapitel 3.3.2 *inloggningsida*). För att man inte skall kunna ta en genväg till en annan sida, förutom inloggningssidan, i systemet så tar varje sida emot användarnamn och lösenord. En jämförelse mot databasen sker varje gång en sida laddas. Är inte användarnamnet och tillhörande lösenord giltigt så kommer det ett felmeddelande.

4.5.2 Enkelt biljett system

När studenten ska göra sin utvärdering krävs det att studenten inte röstar mer än en gång och att röstningen är anonym. Man vill heller inte att någon annan än kursdeltagande skall kunna rösta. För att lösa detta så finns det ett enklare biljettsystem till utvärderingen. I biljettsystemet krävs det att studenten har ett unikt id som han kan verifiera sig med. När studenten har verifierat sig mot systemet returneras en biljett till studenten, den använder studenten för att göra en utvärdering. När biljetten är använd så raderas den från databasen så att det inte går att använda samma biljett igen. För att lösa detta så finns det två tabeller. Den första tabellen, som heter *student*, innehåller tre kolumner, en idkolumn, kontrollkolumn och kurskodskolumn. Idkolumnen innehåller alla studenters id som de verifierar sig med. Kontrollkolumnen säger om studenten har hämtat sin biljett. Kurskodskolumnen talar om för vilken kurs studentens id hör till. Den andra tabellen, som heter *ticket*, innehåller två kolumner, en med biljetter och en med kurskoder. För att studenten skall erhålla sin biljett krävs det att han skriver in en mail-adress när han verifierar sig. Vid lyckad verifiering så skapas biljetten, som är ett slumptal, och skickas iväg till den angivna mail-adressen. Biljetten sparas även undan i *ticket*-tabellen. För att studenten skall lyckas med verifieringen krävs det att han känner till sitt id till den tillhörande kursen och att han inte hämtat sin biljett tidigare. För att sedan få göra utvärderingen så anger man biljetten. När studenten är klar med utvärderingen så raderas biljetten från databasen. Försöker man med samma biljett igen så kommer ett felmeddelande att visas. Svaren till kursutvärderingarna är sparade i olika tabeller och visar inte vilken biljett som användes eller vilket id som användes.



Figur 4.6: Enkelt biljettsystem

Med det enkla biljettsystemet så uppfyller man vissa av de krav som finns för ett röstningssystem. Detta system hindrar att mer än antalet kursdeltagande utvärderar kursen. Detta beror på att man endast genererar nya biljetter vid en lyckad verifiering. Systemet gör att röstningen är anonym, ingen kan se hur någon har svarat i sin utvärdering. Det enda man kan se är om studenten hämtat sin biljett. Biljettsystemet hindrar inte att man kan ge bort sitt Id, d.v.s. sin utvärdering. Studenterna kan inte verifiera om deras utvärdering har lagts till i resultatet.

4.6 Sammanfattning

Ett kursutvärderingssystem liknar ett röstningssystem, fast man lämnar en utvärdering istället för en röst. För att uppfylla vissa krav så användes ett enkelt biljettsystem till kursutvärderingssystemet. Biljettsystemet såg till att det inte kunde komma utomstående och rösta, såvida dom inte hade fått en biljett av någon kursdeltagande. Om det däremot gäller ett röstningssystem så måste man hålla hårdare på att väljare inte kan ge bort sin röst och man

måste också vara säker på vilka som har röstat. För att kunna lösa detta så kan man använda blinda signaturer. Blinda signaturer ser även till att röstaren är anonym. Blinda signaturer är en utökning av digitala signaturer som gör att man inte kan se vad man signerar. För att veta vem som skickat ett meddelande så används en omvänd asymmetrisk kryptering i digitala signaturer. När man krypterar ett meddelande i asymmetrisk kryptering så använder man mottagarens publika nyckel för att kryptera meddelandet och när mottagaren tar emot det så dekrypterar han det med sin privata nyckel. För att göra en digital signatur så dekrypterar man ett meddelande med sin privata nyckel. För att kunna verifiera signaturen så används den publika nyckeln som tillhör signeraren.

5 Resultat och utvärdering

Det kursutvärderingssystem som har implementerats är ännu bara en prototyp. Vid projektets start valdes ett språk som vi tidigare aldrig programmerat i. I början när vi implementerade systemet hade vi väldigt lite kunskap om språket. Det var lätt att göra mindre smarta implementationsval. Prototypen kan alltså ha vissa svagheter som måste rättas till om man vill fortsätta att utveckla den till ett färdigt kursutvärderingssystem.
I detta kapitel skall vi se vad som gjorts och om man kan lösa det på ett bättre sätt. Vi skall värdera prototypen och dess säkerhetsaspekter. Värderingen av prototypen presenteras i fyra olika områden:

- Språk
- Användargränssnitt
- Databas
- Dynamik

Säkerhetsaspekterna som finns ligger i:

- Röstning
- Verifiering
- Databas

5.1 Prototyp

I prototypen har vi kollat på olika aspekter som vi nu kommer att gå igenom.

5.1.1 Språk

Prototypen är skriven i PHP och MySQL. Om man tidigare har programmerat i C/C++ och har lite kunskap om HTML så är det inte svårt att komma igång med PHP. När man börjar med PHP märker man snabbt likheter med C/C++. Trots likheterna behöver man inte deklarera sina variabler. Skriver man ”***$tal = 10;***” så kommer variabeln *tal* att bli en integer.
Om man vill kunna trycka på en knapp för att komma till en annan webbadress, så kan man skapa ett formulär för en knapp. I detta formulär anger man vilken adress som man vill att formuläret ska skickas till. I formuläret anger man även vilka variabler som skickas till den nya webbadressen. Den nya sidan tar emot alla variabler utan att ha någon kunskap om den

tidigare sidan, som variablerna kommer ifrån. När man sedan skall använda en variabel på den nya sidan, skriver man det namn som variabeln har. Skickar man med en variabel som heter *nummer,* så heter den även *nummer* på den nya sidan. Om en sida laddas om, d.v.s. att formuläret anger den befintliga webbadressen, så måste man även då skicka med de variabler man vill använda. De variabler som inte skickas med kommer att förlora sina värden. Formulären i prototypen kommer att innehålla flera rader bara för att skicka med variabler. Formulären i prototypen ligger i funktioner, då måste man även skicka med variablerna till de funktionerna, vilket kan göra det svårt att se strukturen i det hela. Om man t.ex. skickar med sex variabler till en funktion och fyra inte gör något, förutom att de skickas med för något annat ändamål, när sidan laddas om.

I PHP kan man enkelt bygga ihop strängar, det är bara att skriva ”.” mellan text och/eller variabler. Har man en sträng, ***$namn = ”Bob”***, kan man enkelt sätta ihop den med en annan sträng. Skriver man ***$sentence = ”Hi, my name is ” . $namn*** så kommer variabeln *sentence* att få värdet ***”Hi, my name is Bob”***. När man på ett sådant enkelt sätt kan konstruera olika meningar blir det också enkelt att konstruera SQL-frågor i PHP.

I MySQL ställer man vanliga SQL-frågor. Har man arbetat med databaser förut så är det inte svårt med MySQL.

5.1.2 Användargränssnitt

Gränssnitten mot användaren ska vara så enkelt som möjligt. Det ska inte vara svårt att använda vårt system första gången. Knapparnas placering och namn har betydelse för att man enkelt ska förstå sig på ett system. De flesta sidor i kursutvärderingssystemet har endast så många knappar som det behövs för att göra användargränssnittet så enkelt som möjligt. Den sida som man ändrar på en utvärdering, kan vid en första anblick verka lite rörig. Anledningen till det var att vi valde att läraren kunde tillverka sina egna kursutvärderingar, vilket sker dynamisk. Vill man inte ändra på någon utvärdering eller tycker det är för rörigt, så kan man välja att skapa en utvärdering i *default*-läge. I *default*-läget visas inte ”Lägg till fråga” och ”Ta bort fråga”. Statistiken över flervalssvar valde vi att visa i diagram med horisontella staplar, då vi inte hittade något sätt att manipulera diagram vertikalt på ett bra sätt.

5.1.3 Databas

Databasen kan nog byggas upp på ett smartare sätt. I kursutvärderingssystemet så skapas det nya tabeller för varje kursutvärdering. Om man skapar en ny kursutvärdering så skapas en tabell för alla frågor till den och en tabell för alla svar till utvärderingen. Istället för att skapa nya tabeller till varje utvärdering så kan man ha två tabeller. En med frågor till alla

utvärderingar och en med alla svar. För att komma åt rätt frågor så får man ha en extra kolumn med något id och sedan definiera relationer mellan tabellerna.

5.1.4 Dynamik

Ett bra exempel på hur dynamiken kan fungera på vår prototyp är då studenterna går till utvärderingssidan. Innan sidan genereras så vet programtolken inte hur många frågor som utvärderingen kommer att bestå av. Programtolken kommer att läsa in varje fråga från kursens frågetabell, i databasen, tills det inte finns några mer frågor att läsa in. Till varje fråga kommer det att skapas en ny variabel. Dessa variabler kommer att vara satta till de alternativ som studenten valt i utvärderingen, då studenten registrerar sin utvärdering. Antalet frågor kommer alltså vara lika med antalet variabler. För att lösa det använde man varje kolumnnamn i svarstabellen som en variabel. Dessa variabler bäddades sedan in i en HTTP_POST_VARS, för att få fram de värden som variablerna var satta till.

Vissa delar i vår prototyp har lösts statiskt. Läraren kan inte bestämma helt och hållet hur en utvärdering ska se ut. T.ex. så kan han inte bestämma hur många radioknappar som svaret skall innehålla. Det kommer bara att finnas fem radioknappar, från ”dåligt” till ”mycket bra”, att välja på. Man kan heller inte dela upp utvärderingen i olika delar, som t.ex. en litteraturdel och en laborationsdel. För att lösa så att dessa statiska delar blir dynamiska, så måste man utöka informationen i databasen. Ett exempel på det kan vara att lägga till en extra kolumn i frågetabellen som talar om hur många radioknappar det skall vara till varje fråga. Om man ändrar så att antalet svarsalternativ är varierande i utvärderingen, så krävs det även att man ändrar på statistiksidan. Den måste då även ta hänsyn till hur många radioknappar som finns när den genererar diagrammen.

I våra rullmenyer är det också dynamik. Alternativen läses in från databasen och på så sätt kommer de alltid att vara uppdaterade. Om man vill ta bort en utvärdering så får man välja från en rullmeny, vilken utvärdering som man vill ta bort. Efter man tagit bort en utvärdering så kommer rullmenyn att vara uppdaterad, eftersom man läser om från databasen efter att man tagit bort en utvärdering.

5.2 Säkerhetsaspekter

Säkerhetsaspekterna kring kursutvärderingssystemet och röstningsystem i allmänhet skall vi undersöka i de kommande delkapitlen.

5.2.1 Röstning

När man pratade om säkerhet till ett röstningssystem så fanns det vissa krav som man var tvungen att uppfylla, kraven var följande:

1. **Endast behöriga röstare får rösta.**
2. **Ingen kan rösta mer än en gång.**
3. **Ingen kan ta reda på vad någon annan har röstat.**
4. **Ingen kan duplicera någon annans röst.**
5. **Ingen kan ändra någon annans röst utan att bli upptäckt.**
6. **Varje röstare kan vara säker på att deras röst har blivit räknad.**

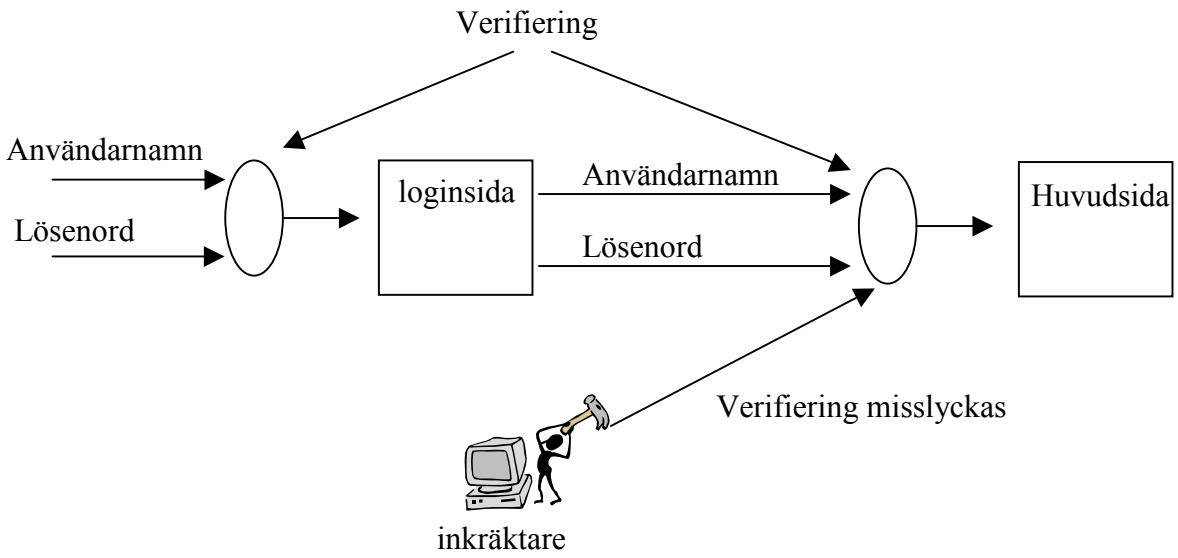
Den tid som var över efter implementationen av prototypen var otillräcklig för att göra vårt system riktigt säkert. Vi tog ändå reda på vad som behövs för ett säkert system. I prototypen så implementerades ett enkelt biljettsystem. Vår prototyp kollar att lika många deltagare som deltog i kursen får rösta. Om någon hämtat en biljett med sitt id så kan man inte hämta en ny biljett med samma id. Skulle någon lista ut vad en student har för id, så kan den hämta ut studentens giltiga biljett. När sedan studenten ska hämta ut sin biljett så kommer han att nekas till att få sin biljett. En student kan enkelt ge bort sin biljett till vem som helst, som kan göra utvärderingen i hans ställe. När utvärderingarna sparas så finns ingen information om hur någon har röstat. Detta enkla biljettsystem är ett val av mindre kvalité. Biljettsystemet valdes på grund av att tiden var kort. Vill man ha ett säkert röstningssystem kan man använda röstning med blinda signaturer. En röstare kan inte fuska med det här systemet. Blinda signaturer ser till att varje röst är unik. Om röstaren försöker skicka in sin röst två gånger så kommer röstningssystemet att se att serienumret redan finns och slänga den andra rösten. Ingen kan generera egna röster för att ingen vet systemets privata nyckel. Ingen kan heller avlyssna och ändra, andra personers röster av samma anledning. Ingen kan ändra sitt serienummer och på så sätt få en ny röst. Röstningssystemet kan inte se hur folk har röstat. Blinda signaturer hindrar röstningssystemet att se serienumret, på rösten, när systemet signerar rösten. Röstningssystemet kan inte länka ihop den signerade ”blinda” rösten med den som man använt för röstning. Serienumren publiceras tillsammans med tillhörande röst, så att de som röstat kan se att deras röst har räknats. Det finns ändå problem med blinda signaturer.

Om inskickningen av rösten inte sker anonymt, så kan röstningssystemet lista ut hur folk röstat.

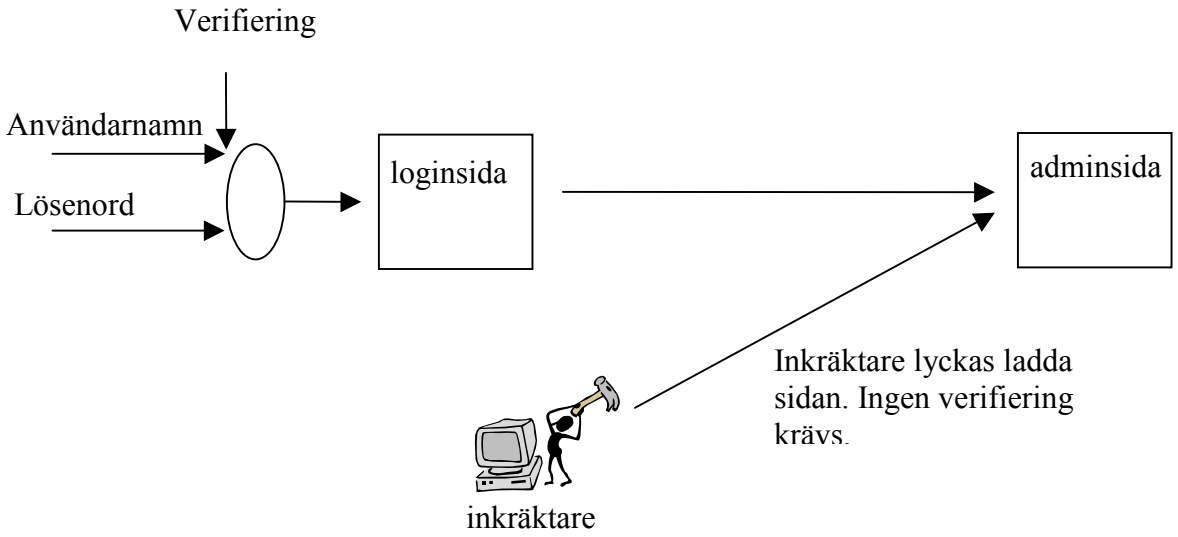
5.2.2 Inloggning/konto

I systemet har man olika användare. Användarna kan t.ex. vara lärare eller andra personer som har hand om kursutvärderingar. Användarna till systemet kan skapa nya utvärderingar, ta bort utvärderingar, ändra i utvärderingar och kolla på statistik på utvärderingar innan de blir publicerade. I prototypen kan användarna göra detta på alla utvärderingar. För att förhindra att man tar bort någon annans utvärdering borde det finnas restriktioner, d.v.s. endast de utvärderingar som hör till användaren skall vara tillgängliga att ta bort. Användare X skall inte kunna ta bort en utvärdering som användare Y skapat, om inte Y har tillåtit det. Det skall inte heller vara möjligt att få se statistik för någon annans utvärdering innan den är publicerad. Det skall heller inte gå att ändra på någon annans sammanfattning till utvärderingar.

Efter en användare har loggat in skickas användarnamn och lösenord med till varje sida som laddas. Hos varje sida måste man verifiera användarnamn och lösenord för att en användare skall få se sidan (se figur 5.1). Om man inte gör så, kan man gå runt inloggningssidan och komma in i systemet (se figur 5.2), om man vet vad en sida heter.



Figur 5.1: Verifiering på varje sida



Figur 5.2: Verifiering endast på loginsidan

Varje sida tar emot användarnamn och det krypterade lösenordet och jämför mot databasen. I databasen sparas användarnamn tillsammans med dess krypterade lösenord. Skulle någon inkräktare komma åt informationen om användare i databasen, skulle han kunna komma åt alla sidor i systemet. För att lösa detta är det bara att skicka med det okrypterade lösenordet istället för det krypterade. Nu när det krypterade lösenordet skickas med så kan inkräktaren, om han vet variabelnamnen, ha en möjlighet att skriva in dessa i adressfönstret. Om variabelnamnen är *user* och *passwd*, och man vill åt en annan sida i systemet så skriver man ***”sidansnamn?user=bob&&passwd=axegalw35xa”*** där ”bob” är en användare och hans krypterade lösenord är ”axegalw35xa”.

5.2.3 Databas

Databasen som systemet har måste även vara säker. Om man tänker sig säkerheten som ett skal, som skall omringa systemet så måste man även se till att databasen är innanför skalet. Om databasen skulle ligga utanför skalet, så får man bara en falsk känsla av att systemet är säkert. Man bör skapa databasen själv för att se till att man har säkerhet även runt den. Databasen som vi använder ligger på ett konto som vi blev tilldelade här på skolan. Det finns alltså en annan root-användare. Root-användaren har rättigheter till allt och kan komma åt all information i databasen.

5.3 Sammanfattning

Prototypen byggdes upp från grunden med ett skriptspråk vi aldrig testat. Språket var lätt att sätta sig in i, kanske på grund av att vi tidigare har programmerat i C/C++. Det var väldigt likt. För att kommunicera med databasen så användes MySQL, som också var lätt att förstå sig på eftersom vi i en tidigare kurs fått erfarenhet av databashantering. Databasen är lite ostrukturerad. Det skapas nya tabeller för varje kursutvärdering. Detta var för att kolumnnamnen används som variabelnamn. För att utöka dynamiken i prototypen, så att man kan skapa utvärderingar på ett mer dynamiskt sätt, får man utöka informationen i databasen.

Användargränssnittet är enkelt så att det inte skall vara svårt att använda systemet. Systemet måste vara mer än användarvänligt för att det skall komma i bruk. Det krävs även vissa säkerhetsmekanismer. Prototypen har ett enkelt biljettsystem implementerat som säkerhet. Biljettsystemet hindrar från att vem som helst kan utvärdera, men det uppfyller inte alla krav som finns för säker röstning över internet. Vill man att alla krav skall uppfyllas kan man använda sig av blinda signaturer.

I databasen lagras alla användare och deras lösenord. Lösenorden som sparas i databasen är krypterade. Om någon kommer på ett krypterat lösenord, så kan det användas på ett felaktigt sätt och ta sig in i systemet, men vi har tagit upp en enkel lösning på detta problem.

6 Sammanfattning

Ett webbaserat utvärderingssystem är inte så enkelt som det låter att producera. Ett sådant system består till huvudsak av två huvudkomponenter. Den första komponenten är ett fungerande system. Den andra komponenten är säkerheten. Dessa båda huvudfaktorer måste kombineras och samverka i harmoni för att skapa ett bra system. En stor del hänger på säkerheten då man vill skapa ett säkert system över internet. Särskilt då det gäller valsystem, där folk röstar för ett regeringsval. Man måste kunna garantera att man kan förhindra fusk och att användare är och förblir anonyma. Idealet för ett säkert system som vi tidigare har gått igenom, är att det klarar följande sex krav om inte mer.

1. **Endast behöriga röstare får rösta.**
2. **Ingen kan rösta mer än en gång.**
3. **Ingen kan ta reda på vad någon annan har röstat.**
4. **Ingen kan duplicera någon annans röst.**
5. **Ingen kan ändra någon annans röst utan att bli upptäckt.**
6. **Varje röstare kan vara säker på att deras röst har blivit räknad.**

Om minst alla av dessa krav uppfylls, kan man säga att systemet är så säkert att man kan använda det för t.ex. ett valsystem.

6.1 Detta projekt

Detta projekt gick ut på att skapa en prototyp på ett kursutvärderingssystem. I mån av tid skulle det läggas på någon typ av säkerhet på prototypen.

6.1.1 Prototyp

Prototypen innehåller basala funktioner såsom ”*skapa utvärdering*”, ”*ta bort utvärdering*”, ”*se statistik på utvärdering*” mm. Dessa funktioner skulle inte ha någon större betydelse om vi inte gjort en sida för studenten att fylla i, och registrera, sin utvärdering på. Det finns en del saker med vår prototyp som vi idag skulle vilja göra om. Desto mer vi har programmerat, desto mer har vi lärt oss om PHP och MySQL. Vi ser idag andra lösningar än de vi använde

oss av då vi började programmera. Prototypen är inte perfekt, men den fungerar. Det är väldigt enkelt att lägga till nya funktioner och ta bort gamla.
Språket som vi har valt att programmera i, är väldigt lätt att sätta sig in i och man kan göra mycket konstruktivt med det. Det har även speciella funktioner för att kommunicera med MySQL-databasen.

6.1.2 Säkerhet

Säkerheten till vår prototyp är inte den bästa. På grund av tidsbrist så finns det inte så mycket säkerhet som man kanske skulle önska. Det bästa vore om att man kunde applicera blinda signaturer på systemet, men det insåg vi redan tidigt att tiden inte räckte till. Vår prototyp innehåller en bråkdel av den säkerhet som skulle behövas för att prototypen skall räknas som säker. Vi uppfyller i bästa fall två av de sex punkter som vi listade upp tidigare i detta kapitel.

6.2 Framtida Utvecklingsarbete

Prototypen kan vara en grund för ett framtida utvecklingsarbete eller så kan den vara ett hjälpmedel. Med ett hjälpmedel menar vi att prototypen kan ge lösningar på problem som uppstår om man har valt att börja om från början med ett nytt system. Det som behövs ändras i vår prototyp är att göra om några funktioner från att vara statiska till att vara dynamiska, vilket vi gått igenom i Kapitel 5. Den största delen som det krävs förbättring är i säkerheten. Säkerheten i vår prototyp är otillräcklig. Det som krävs för att skapa en helt säker röstning eller utvärdering, är de sex punkter som vi gick igenom tidigare i detta kapitel. Dessa punkter uppfylls om man applicerar röstning med blinda signaturer. I kapitel 4 beskriver vi hur man kan använda blinda signaturer för att åstadkomma en säker röstning. Något mer än säkerhet och småändringar i vårt program är det förmodligen inte. Men vi förmodar att implementera blinda signaturer är ett helt kapitel för sig själv.

6.3 Andra Tillämpningar

Prototypen visar hur lätt det är att skapa ett system på internet. Prototypen bygger på olika funktioner och det visar att man lätt kan skapa olika typer av system t.ex. valsystem, utvärderingssystem eller ett system som kan användas för duggor eller tentor. Dessa system liknar varandra mycket, i alla fall om man talar om implementationen. Informationen i formulären kanske skiljer sig åt, men de kommer fortfarande att vara uppbyggda på ett liknande sätt. Säkerheten på alla dessa system är viktig, men i vissa fall kanske den är viktigare.

6.3.1 Valsystem

Exempel på ett annat system som man kan skapa är ett valsystem, t.ex. till ett regeringsval. På ett sådant system krävs det att det inte går att fuska. Minst alla av de sex punkter måste uppfyllas om det överhuvudtaget skall kunna räknas som säkert. Väljaren måste också kunna garanteras att vara anonym, så att t.ex. motståndarpartiet i valet inte kan påverka väljaren på något sätt eller trakassera väljaren.

6.3.2 Duggor/tentor

Ett annat exempel på system som kan göras och användas över internet, är system där man kan göra duggor eller tentor. Ett system där man kan göra tentor, kräver också en hög säkerhet liksom valsystemet. Man vill inte att folk ska kunna fuska här heller. Men här är inte anonymiteten något problem, eftersom man måste veta vem som har gjort tentan. Man måste veta att det är just den personen och ingen annan person som har gjort tentan.

6.4 Allmän slutsats

Man måste ha klart för sig redan från början att ett fungerande system med högsta tänkbara säkerhet inte tar en vecka att implementera, vare sig om man ska göra ett kursutvärderingssystem eller ett valsystem så tar det lång tid. I en artikel i [3] *Gardian Weekly* skrivs det, ”Britain will become the first country in the world to use internet for voting…”. Vidare står det att de vill att systemet skall vara uppe och fungera till nästa val, men de medger att det kommer att bli svårt att hinna. Det skulle ta dem flera år att producera ett färdigt valsystem. Eftersom vi var utan kod och dokumentation till det existerande kursutvärderings-systemet, var vi tvingade till att börja från början med ett nytt system. Vi

fick fram en fungerande prototyp men prototypen hade inte så mycket säkerhet som vi hade önskat. Detta beror främst på tidsbrist. Prototypen är relativt enkel gjord och lätt att förstå sig på. Om det kommer någon efter oss i utvecklingsarbetet av systemet, så är prototypen lätt att sätta sig in i. Väljer man ändå att inte använda prototypen så kan den vara bra som hjälpmedel då man gör ett nytt system. Den innehåller enkla lösningar till dynamiska problem, vilket man kan använda sig av.

Säkerheten till prototypen är vi mindre nöjda med. Det finns många problem med den. Skulle man besluta sig för att använda denna prototyp som grund till ett system, så måste man tänka om när det gäller säkerhet. Röstning med blinda signaturer som vi gått igenom i kapitel 4, skulle kunna vara en bra lösning för att få en säkerhet som uppfyller alla de sex krav som ett system minst måste uppfylla för att räknas som säkert.

Referenser

[1] Schneier, Bruce. (1996) *Applied Cryptography*. USA: John Wiley & Sons Inc.

[2] Ek, Jesper & Eriksson, Ulrika. (2001) *PHP4 programmering*. Sundbyberg: Pagina Förlags AB.

[3] Ronne, Erik.(1999) *ASP, Active Server Pages*. Stockholm: Docendo Sverige AB.

[4] The PHP group. (2002-04-25) www.php.net.

[5] INT Media Group, Incorporated. (2002-05-15) www.phpbuilder.net

[6] MySQL AB. (2002-03-15) www.mysql.com, www.mysql.com/products/what_is_mysql.html

[7] R. Rivest. (2002-05-10) www.globecom.net/ietf/rfc/rfc1321.html

[8] Fsdata. (2002-04-30) www.fsdata.se/manual/manual10.shtml

[9] Hägglund, Jonas; Granholm, Daniel & Akhavam, Mojgan. (2002-03-01)

[10] Lundström, Maria; Åsbergh, Anna & Norell, Elisa. (2002-03-02) hemsidor.torget.se/users/2/2273/html.html

[11] (2002-03-15) hoohoo.ncsa.uiuc.edu/cgi/overview.html

[12] The CGI Resource Index. (2002-03-15) cgi.resourceindex.com/

A Bilaga Systemöversikt