Algoritm för keystroke dynamics inspirerad av viktad sannolikhet och fuzzy logic

(HS-IKI-MD-04-303)

James Dickson (a00jamdi@student.his.se)

Institutionen för kommunikation och information
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på det datavetenskapliga programmet under vårterminen 2004.

Handledare: Henrik Engström

Lista över vanliga begrepp

Biometri (en. Biometrics) En säkerhetsmekanism vilken använder sig av fysiologiska eller beteendeattribut för att identifiera eller verifiera en användare (se identifiera och verifiera).

Duration Tiden en tangent hålls ned.

Error Rate Andelen felaktigt klassificerade input (se även FAR och FRR).

FAR False Acceptance Rate, den grad av felaktigt accepterade input i biometriska system.

FRR False Rejection Rate, den grad av felaktigt nekade input i biometriska system.

Identifiering En användare av ett system skall pekas ut (identifieras) i en databas av möjliga användare. Det vill säga att identifiera användaren är att avgöra vem användaren är.

Keystroke Dynamics En biometrisk metod vilken går ut på att verifiera eller identifiera användare utifrån dess unika sätt att använda tangentbordet.

Kontinuerlig verifiering Se verifiering

Latency Tiden mellan två tangentnedslag (se även duration).

Mönster-vektor En vektor med information om ett mönster (vanl. skrivsättsmönster)

Skrivsätts-mönster Det sätt som en användare brukar sitt tangentbord.

Statisk Verifiering Se verifiering

Tröskelvärde Den acceptansnivå ett biometriskt system har. Ett lågt tröskelvärde betyder att acceptansnivån är hög och tvärtom. Starkt sammankopplat med FAR och FRR.

Verifiering Med verifiering menas i rapporten vanligen en normal inloggningsprocedur med lösenord. Vid kontinuerlig verifiering sker själva verifieringen kontinuerligt då användaren är inloggad i systemet. Exempelvis var 20:e minut. Vid statisk verifiering; verifieras användaren endast en gång (vid inloggning).

Algoritm för keystroke dynamics inspirerad av viktad sannolikhet och fuzzy logic

Examensrapport inlämnad av James Dickson till Högskolan i Skövde, för Magisterexamen (M.Sc.) vid Institutionen för Kommunikation och Information.

2004-06-04

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Algorithm for keystroke dynamics inspired by weighted probability and fuzzy logic

Submitted by James Dickson to Högskolan Skövde as a dissertation for the degree of M.Sc., in the Department of Communication and Information.

2004-06-04

I certify that all material in this dissertation which is not my own work has been identified and that no material is included for which a degree has previously been conferred on me.

Signed: _____

Tack till

Henrik Engström för utmärkt handledning samt många goda förslag och tankar kring arbetet.

Academic AB (IT-gymnasiet), ***Partner IT-Utbildarna***, ***Tieto Enator*** och ***WM-Data*** som lånade ut sin tid till experimenten i denna rapport.

Alla försökspersoner dels från företagen (i huvudexperimentet) och de sju försökspersonerna i det initiala experimentet.

Ayken Guven för ingående förklaringar av den metod för keystroke dynamics som han och I. Sogukpinar introducerade i sin artikel *Understanding Users' keystroke patterns for computer access security* (2003).

Algoritm för keystroke dynamics inspirerad av viktad sannolikhet och fuzzy logic

James Dickson (a00jamdi@student.his.se)

Sammanfattning

Biometri är en relativt ny säkerhetsmetod för datorsystem. Biometri används ofta för att ersätta eller kombineras med användarnamn och lösenord. Detta görs genom att mäta ett fysiologiskt attribut eller beteendeattribut hos användaren. Keystroke dynamics är en biometrisk metod vilken registrerar användarens sätt att skriva på tangentbordet. En stor mängd försök med keystroke dynamics har gjorts i tidigare arbeten. Många av dessa har utgått ifrån metoder vilka använder ett högt antal stickprov från användarens beteende vid tangentbordet. Optimalt är dock en metod med hög säkerhet men samtidigt använder ett lågt antal stickprov. Denna rapport introducerar en ny algoritm för implementering av keystroke dynamics, vilken jämförs med två existerande algoritmer. Denna rapport visar att den nya algoritmen har högre prestanda än de övriga två i jämförelsen.

Nyckelord: keystroke dynamics, biometri, säkerhet, statisk verifiering

INNEHÅLLSFÖRTECKNING

1. Introduktion

I mitten av 90-talet ökade nätverksanvändningen vilket gjorde det lättare för företag att kommunicera (Meadows, 1997). Monrose & Rubin (2000) menar att detta även förenklade våra liv, men den ökade nätverksanvändningen var dock inte enbart positiv. Vi blir i och med datoriseringen mer och mer beroende av maskinerna, då de är en vital del i allt från flygkontrollsystem till banktransaktioner (Monrose & Rubin, 1997). En rad nya säkerhetshot skapas varje gång ny funktionalitet införs i verksamheten med hjälp av datorsystem (Meadows, 1997). Då fler och fler företag och organisationer datoriserar uppstår således nya hot mot verksamheten som måste mötas (Meadows, 1997). Ett exempel kan vara *hackers* som vill komma åt känslig information såsom kreditkortsnummer eller kopierar bankkort och PIN-koder (Furnell & Ord, 2000), (Kim, 1995). Förändring eller ej auktoriserad radering av känslig information är ett ytterligare hot som finns i alla datorsystem (Alves-Foss & Barbosa, 1995).

Datorsäkerhet innebär bland annat att veta vem som använder ett system och att ha möjlighet att styra vem som kan komma åt systemets olika funktioner (Chalmers, 1984). En viktig aspekt är således att *identifiera* och *verifiera* användarna. Med identifiering menas i detta fall att man har en databas över personer med möjlighet att använda en resurs. Databasen används av ett kontrollsystem för att ta reda på vem av de möjliga användarna som använder resursen. Med verifiering menas det som normalt kallas inloggning. Det vill säga, att försäkra sig om att personen är den som han/hon uppger sig för att vara (oftast genom användarnamn och lösenord) (Association For Biometrics, 2003).

Traditionellt sett har användarnamn och lösenord använts som metod för att verifiera personer som har tillgång till ett system (Chalmers, 1984, Monrose, Reiter & Wetzel, 1999). Lösenord har dock visat sig vara en svag metod för detta ändamål. Exempelvis tycks användare ha en tendens att välja lösenord som lätt kan gissas (Chalmers, 1984, Legget, Williams & Usnick, 1991, Monrose, Reiter & Wetzel, 1999). Downland, Furnell, Illingworth & Reynolds (2000) har gjort undersökningar som visar att så mycket som 21% av lösenorden som ingick i undersökningarna kunde gissas av en PC på en vecka. Om lösenord delas ut för att försvåra gissningar så tenderar användarna att skriva ned dessa på lappar vilka de har lätt tillgängliga nära sin datorstation (Chalmers, 1984, Monrose, Reiter & Wetzel, 1999). Legget, Williams & Usnick (1991) hävdar att stulna lösenord ofta kan användas en lång tid innan stölden upptäcks. Traditionella metoder såsom lösenord är därmed inte tillräckliga för att tillhandahålla den säkerhetsnivå som krävs (Monrose & Rubin, 1997). Nya säkerhetsmekanismer behövs således för att förhindra otillåten access av datorsystem (Monrose & Rubin, 1997).

Enligt Mahar, Napier, Wagner, Laverty, Henderson & Hiron (1995) är användarens sätt att använda tangentbordet en av de mest lovande metoderna för att identifiera en användare utifrån dess beteende. Denna rapport beskriver just denna säkerhetsmetod; *Keystroke Dynamics*.

Rapporten baseras på tidigare arbeten av Umphress & Williams (1985), Legget & Williams (1995), Joyce & Gupta (1990), Mahar et al. (1995), Monrose & Rubin (2000), Cantú, Gutiérrez, Lerma-Rascón & Salgado-Garza (2002). Dessa har undersökt olika

metoder och algoritmer inom området och nått lyckade resultat då det gäller att klassificera användare korrekt. Alla dessa förändrar dock den normala inloggningsproceduren (användarnamn och lösenord) på något sätt, vilket är önskvärt att försöka undvika. Antalet felklassificeringar är också fortfarande relativt högt vilket motiverar nya och förbättrade metoder (Guven & Sogukpinar, 2003). En ny algoritm för keystroke dynamics introduceras i denna rapport vilken höjer säkerhetsmetodens prestanda, då den används för verifiering med ett vanligt användarnamn och lösenord, i jämförelse med två tidigare vanligt använda algoritmer.

2. Bakgrund

Detta kapitel beskriver säkerhetsmetoden biometri övergripande. Keystroke dynamics, vilken är en speciell gren av biometri, beskrivs mer i detalj. Även två tidigare använda metoder för att implementera keystroke dynamics beskrivs övergripande.

De flesta typer av säkerhetsmekanismer går ut på att användaren först hävdar att han/hon är en specifik person (genom exempelvis användarnamn) för att sedan bevisa detta. I konventionella säkerhetsmetoder görs detta med ett lösenord (Rejman-Greene, 2001). Detta kallas *verifiering*.

I ett system som syftar till att identifiera användaren (*identifiering*) krävs det ej att användaren hävdar att han/hon är en användare som systemet känner igen. Systemet identifierar istället användaren med hjälp av annan information såsom beteendemönster eller fysiologiska attribut (Rejman-Greene, 2001).

Då användaren identifieras eller verifieras med hjälp av beteendemönster eller fysiologiska attribut kallas systemet för *biometriskt* (Mahar, et al., 1995).

2.1 Biometri

Biometri (*Biometrics*) definieras som:

> *"A measurable, physical characteristic or personal behavioral trait used to recognize the identity, or verify the claimed identity, of an enrollee"*
>
> Association for Biometrics (2003)

Biometri delas ofta in i de två kategorier som framgår av ovanstående definition: fysiska attribut och karaktären av användarens beteende (Bergadano, Gunetti & Picardi, 2002). Biometri är med andra ord att identifiera eller verifiera identiteten hos användarna med hjälp av dess beteende eller fysiologiska attribut (*biometriskt mått*).

Biometri har under senare år börjat användas i större utsträckning för att implementera säkerhetsmekanismer såsom identifiering och verifiering av användare (Rejman-Greene, 2001, Rejman-Greene, 2002, Guven & Sogukpinar, 2003).

En stor fördel med biometri som säkerhetsmetod gentemot vanliga lösenord är att biometriska signaturer inte kan tappas bort eller stjälas (Rejman-Greene, 2002). En följd av biometri är dock att endast en användare kan ha tillgång till en speciell användaridentitet eftersom fysiska attribut eller beteende inte kan lånas ut (Ilonen, 2003). Då man använder sig av biometri uppkommer en del frågor som rör användarnas integritet. Exempelvis kan en databas med fingeravtryck vara känsligt för vissa användare då databasen kan samköras med exempelvis brottsregister. Detta kan leda till låg användaracceptans (Gifford, McCartney & Seal). Fördelarna med biometri är dock enligt Rejman-Greene (2001) så stora så att de uppväger nackdelarna.

Gifford, McCartney & Seal (1999) pekar på fyra punkter som gör framtiden för biometri ljus.

- Mer spridd användning av nätverk och därmed även inloggningskoder (lösenord) gör det svårare att komma ihåg alla olika lösenord.
- Ökad e-handel kräver hög säkerhet då pengar skall hanteras över Internet.
- Billigare hårdvara för biometrisk identifiering.
- Datavetenskapen har på senare år fått fram bättre hård och mjukvara för ändamålet.

2.1.1 Existerande metoder för biometri

Det finns enligt Rejman-Greene (2002) ett stort antal metoder för biometri såsom: fingeravtryck, irisigenkänning, ansiktsigenkänning med flera. Korotkaya (2003) har analyserat möjligheten att använda lukt som biometrisk metod. Det finns flera kommersiella applikationer som använder sig av biometri för att förstärka säkerheten i systemet (Rejman-Greene, 2001). Lukt används inte ännu i något kommersiellt biometriskt system (Korotkaya, 2003).

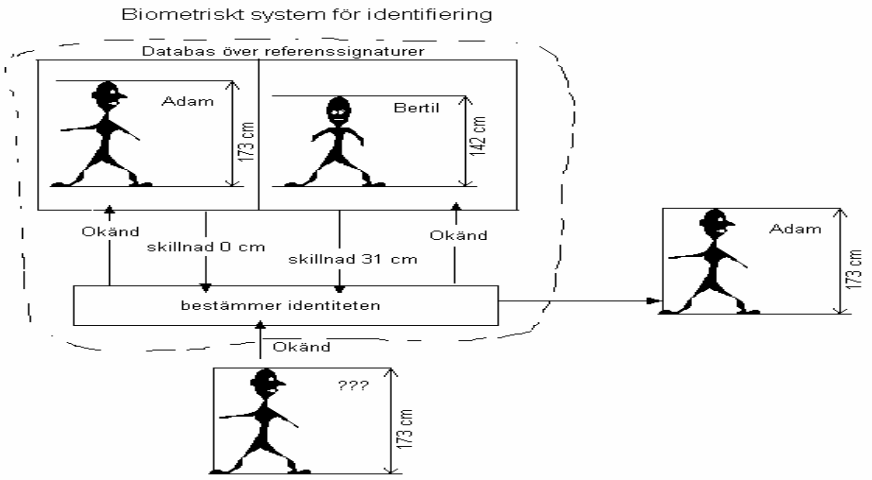
Rejman-Greene (2001) hävdar att de olika metoder som finns för biometriska system skiljer sig mycket från varandra i prestanda. Inte bara metoderna i sig utan även olika implementationer av samma metod ger olika prestanda (Rejman-Greene, 2001, Rejman-Greene, 2002). Det vill säga inte bara *vad* som mäts (fingeravtryck, iris, etc.) utan även *hur* det mäts (vilka specifika algoritmer som används).

2.1.2 Referenssignatur

För att det biometriska systemet skall ha möjlighet att verifiera eller identifiera en användare utifrån dess beteende eller fysiologiska attribut krävs att det finns en möjlighet att få ett mått på hur likt det som mäts är den typiska signaturen för användaren (Rejman-Greene, 2001). Denna typiska signatur kallas ofta *referenssignatur* vilket med andra ord är en mall som innehåller typiska exempel på mått tagna från användarens beteende eller fysiologiska attribut (Joyce & Gupta, 1990).

Exempel på användning av en referenssignatur för identifiering

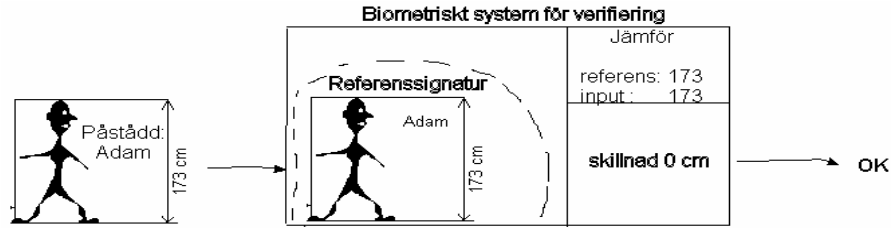
En referenssignatur för en användare skulle exempelvis kunna innehålla information om hans/hennes längd i centimeter. Vid identifiering så identifieras användaren till den användare vars referenssignatur ligger närmast hans/hennes längd (se figur 1).



Figur 1. – Biometriskt system för identifiering: användaren identifieras av systemet med hjälp av sin längd.

Exempel på användning av en referenssignatur för verifiering

Vid verifiering får användaren först hävda sin identitet. Därefter så måste han/hon bevisa (verifiera) att det verkligen stämmer (Rejman-Greene, 2001). I ett biometriskt system nekas eller accepteras en användare tillträde till ett system, utifrån hur väl dess input (i detta exempel användarens längd) överensstämmer med referenssignaturen för den specifika användare som han/hon uppger sig för att vara (Rejman-Greene, 2001).



Figur 2. – Biometriskt system för verifiering. Personen försöker bevisa (verifiera) sin identitet med hjälp av sin längd. Om längden överrensstämmer ges användaren tillträde till systemet.

För att acceptera vissa skillnader mellan referenssignatur och input används ett värde på hur mycket som får skilja (i figur 2; variationer i längd mellan referenssignatur och dess påstådde ägare) (Mahar et al., 1995). Detta värde kallas *tröskelvärde*.

2.1.3 Tröskelvärde

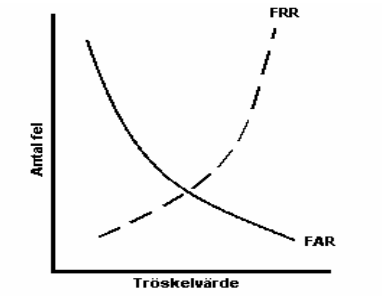
Alla biometriska system använder någon typ av tröskelvärden (Rejman-Greene, 2001). Med tröskelvärden menas den acceptansnivå som systemet har. För att öka tydligheten; om längd används för att verifiera användare så kan tröskelvärdet sättas till det antal centimeter det får skilja mellan användaren som skall verifieras och referenssignaturen. Tröskelvärdet har stor inverkan på systemets prestanda (Matayas & Stapleton, 2000).

2.1.4 Prestandamått

Enligt Matayas & Stapleton (2000) lider alla biometriska applikationer för verifiering av ett visst mått av felaktigt accepterad indata eller felaktigt nekad indata.

Effektiviteten i ett biometriskt system för verifiering kan mätas i två typer av fel; *False Acceptance Rate* (FAR) och *False Rejection Rat*e (FRR) (Mahar, et al., 1995, Monrose & Rubin 2000, Furnell & Ord, 2000,Guven & Sogukpinar, 2003).

Enligt Matayas & Stapleton (2000), (Rejman-Greene, 2001) är normalt FRR och FAR ömsesidigt uteslutande på så sätt att låg FRR medför hög FAR och tvärtom[1]. För att ha möjlighet att justera hur hög FRR som skall tillåtas används tröskelvärden. I figur 3 visas ett exempel på detta.



Figur 3. – Justering av tröskelvärdet (Rejman-Greene, 2001). Om tröskelvärdet höjs ökar FRR, om det sänks ökar i stället FAR.

Furnell & Ord (2000) använder i sin artikel ett konstant FRR på 30%. De anser nämligen att 30% misslyckade inloggningsförsök accepteras av användarna.

Då det gäller identifiering mäts prestanda i *Error Rate* vilket är ett mått på hur stor procent av användarnas identifieringsförsök som felaktigt nekas eller ges tillgång till systemet. Det vill säga; antalet felaktiga klassificeringar uttryckt i procent (Ilonen, 2003).

[1] Hög FAR behöver nödvändigtvis ej betyda låg FRR eller tvärtom. Det ömsesidiga uteslutandet kan ses som relativt. Det vill säga om; en metod sänker sitt tröskelvärde ökar normalt sett FAR medan FRR minskar och tvärtom.

2.1.5 Prestanda och användbarhet

Enligt Rejman-Greene (2001) så bör karaktärsdraget, som mäts av den biometriska metoden, vara så pass unikt för varje användare att det ökar säkerheten i systemet. Just att mätvärdet är unikt för varje användare ställer frågor om användarnas integritet (Rejman-Greene, 2001). Många metoder har alltför tidskrävande beräkningar för att vara användbara (Rejman-Greene, 2001). Mahar et al. (1995) menar att metoden *keystroke dynamics*, vilken går ut på att mäta användares unika skrivsättsmönster, är den mest lovande då det gäller att identifiera eller verifiera användare utifrån dess beteende. Keystroke dynamics är dessutom inte integritetskränkande (då sparade mätvärden inte kan samköras med andra register i stil med brottsregistret) och kräver inte långa beräkningstider (Monrose & Rubin, 2000).

En ytterligare fördel med Keystroke Dynamics, identifierad av Cantú, Gutiérrez, Lerma-Rascón & Salgado-Garza (2002) och även av Bergadano, Gunetti & Picardi (2002), är att ingen ytterligare hårdvara än standardutrustning behöver köpas in.

2.2 Keystroke Dynamics som biometrisk metod

Keystroke Dynamics som biometrisk metod bygger på att människor använder tangentbord på ett karaktäristiskt sätt. Skillnader mellan olika användares tangentbordsanvändning används som referenssignatur (Ilonen, 2003, Yu & Cho, 2003).

2.2.1 Identifiering eller verifiering inom keystroke dynamics

Keystroke dynamics kan användas för både verifiering och identifiering (Ilonen, 2003, Mahar, et al., 1995).

Identifiering

Med identifiering menar Ilonen (2003) att en stor mängd tangentbordstryckningar (exempelvis ett flertal meningar) samlas in och jämförs med tidigare insamlade referenssignaturer av alla användare, för att på så sätt kunna identifiera vem det är som sitter vid tangentbordet.

Verifiering

Ilonen (2003) delar, i kontexten keystroke dynamics, upp verifiering i två olika typer nämligen *statisk verifiering* och *kontinuerlig verifiering*.

Statisk verifiering

Med statisk verifiering menar Ilonen (2003) att identiteten av användaren verifieras då han/hon loggar in genom att skrivsättsmönstret på lösenordet jämförs med användarens referenssignatur (det vill säga användarens normala skrivsättsmönster för lösenordet). Med andra ord används statisk verifiering som en extra säkerhet då lösenord används som säkerhetsmekanism.

Kontinuerlig verifiering

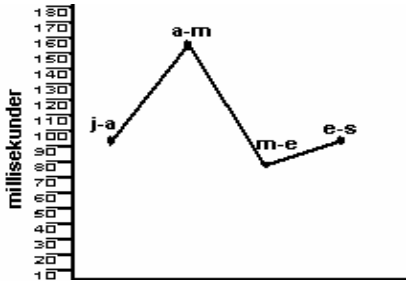
Med kontinuerlig verifiering menar Ilonen (2003) då en användare kontinuerligt verifieras medan han/hon arbetar framför datorn. Det vill säga istället för att endast utföra verifieringen vid inloggningstillfället så görs det exempelvis var 20:e inmatad mening (Ilonen, 2003).

2.2.2 Vad karaktäriserar ett skrivsättsmönster

Keystroke Dynamics fungerar på så sätt att en referenssignatur skapas baserat på hur en användare använder tangentbordet. Det som vanligtvis används som mätvärden är *duration* och *latencies*.

Duration och latencies

En referenssignatur innehåller det typiska användarexemplet för en specifik användare i fråga om exempelvis *duration:* hur länge han/hon trycker ned tangenter, *latency*: tiden mellan varje tangentnedslag (Ilonen, 2003). För statisk verifiering används normalt sett latencies för ett fåtal ord (exempelvis ett lösenord). Figur 4 visar ett diagram över latencies för lösenordet ”james”.



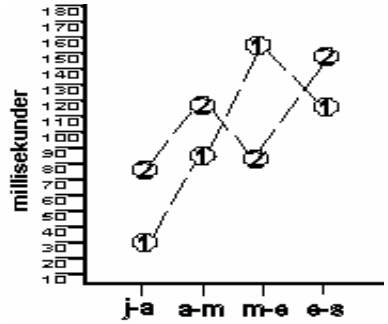
Figur 4. – Diagram över latencies för ordet: ”james”

Referenssignaturen jämförs vid varje verifierings eller identifieringstillfälle med indatan (exempelvis durationer uppmätta på lösenordet). Enligt Monrose & Rubin (2000) går det inte att säga utifrån tidigare arbeten om mätning av både duration och latency ger en bättre prestanda. Monrose & Rubin (2000) anser att antalet experiment som genomförts i tidigare undersökningar såsom Mahar et al. (1995) är för få för att kunna dra några slutsatser.

Stabila lutningar

Joyce & Gupta (1990) menar att varje användare har en *stabil lutning* mellan vissa latencies i referenssignaturen. Med detta menar de att vissa latencies alltid är högre än andra i samma inmatningssekvens (exempelvis för ett lösenord). I figur 5, skulle lutningen mellan j-a och a-m kunna vara ett sådant förhållande där alla latencies för j-a är lägre än de för a-m i samma inmatningssekvens.

Det som tydligt inte är en sådan stabil lutning i figur 5 är förhållandet mellan a-m och m-e, då latency för a-m vid första inmatningen är lägre än latency för m-e och högre vid andra inmatningen. Det vill säga lutningen mellan a-m och m-e är stigande första gången och avtagande den andra.



Figur 5. – Stabil lutning för latencies j-a och a-m i ordet james

Joyce & Gupta (1990) menar att i en optimal algoritm så måste stabila lutningar i skrivsättsmönstret tas till vara på. Detta föreslår författarna skall kallas *”slopes of the signature curve”*. I denna rapport kommer detta att kallas för *stabila lutningar*.

2.2.3 Prestandafrågor för keystroke dynamics

Enligt Bergadano, Gunetti & Picardi (2002) är keystroke dynamics en svårare biometrisk säkerhetsmekanism att implementera än de som tillhör kategorin av biometri som använder fysiska attribut för att identifiera eller verifiera användaren. Detta beroende på att det sätt som människor använder tangentbordet varierar mycket beroende på hur de mår fysiskt och psykiskt (Bergadano, Gunetti & Picardi, 2002).

Olika grupper av användare tros också ha en signatur som är mer eller mindre lämpad för keystroke dynamics. Exempelvis tros ovana datoranvändare ha en mindre lämpad signatur för keystroke dynamics (Monrose & Rubin, 1997).

2.2.4 Hur likhet mellan två skrivsättsmönster mäts

En rad olika sätt att mäta likhet mellan referenssignaturen och inmatning (klassificering) har använts för keystroke dynamics (Umphress & Williams, 1985, Monrose & Rubin, 2000, Joyce & Gupta, 1990, Ilonen, 2003).

Nedan beskrivs några av de mest använda modellerna för att få fram ett mått på likhet mellan referenssignaturen och inmatning i olika metoder för keystroke dynamics (Guven & Sogukpinar, 2003). Dessa kommer att användas i stora delar av rapporten.

Euklidisk distans

Euklidiskt distansmått mäter det diagonala avståndet mellan två punkter och beräknas.

$$D(R, U) = \left[\sum_{i=1}^{n} (R_i - U_i)^2 \right]^{1/2}$$

R: Referenssignaturen
U: Den vektor som är indata vilken skall identifieras eller verifieras.
n: Antalet element i vektorerna (exempelvis antalet latencies).

Monrose & Rubin (2000) hävdar att euklidiskt distansmått är användbart för att mäta likheten mellan referenssignaturen och en användares inmatning. Euklidiskt distansmått är dock svårt att finjustera (Guven & Sogukpinar, 2003). Därför menar Guven & Sogukpinar (2003) att man bör se sig om efter bättre modeller för att fånga mönstret hos användarnas tangentbordsanvändning.

Icke-viktad sannolikhet (Non-Weighted Probability)

Denna modell bygger på sannolikhetsteori och har enligt Monrose & Rubin (1997) en något högre prestanda än euklidiskt distansmått. Denna matematiska modell använder sig av ett beräkningsmått som Monrose & Rubin (1997) kallar Score. Detta beräknas som en matematisk funktion vilken summerar sannolikheterna för att varje enskilt element tillhör referenssignaturen vilket då används som ett mått på hur *lik* hela inmatningen är referenssignaturen[1]:

$$Score(R, U) = \sum_{i=1}^{n} \Pr ob\left(\frac{U_i - \mu_{R_i}}{\sigma_{R_i}}\right)$$

(Monrose & Rubin, 1997, Monrose & Rubin, 2000, Guven, Sogukpinar, 2003)

i : element i vektorn (exempelvis ett specifikt latency)
μ_{R_i} : Medelvärde för referenssignaturens element *i*
σ_{R_i} : Standardavvikelse för referenssignaturens element *i*
Prob(x): Sannolikheten för x enligt normalfördelning[2].

Vid identifiering så identifieras användaren till den vars referenssignatur ges högst värde då funktionen räknas ut (Monrose & Rubin, 2000). Vid verifiering kan modellen

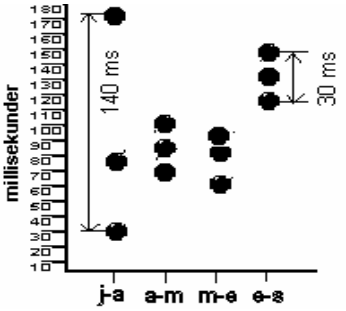
[1] Funktionen är för läsbarhetens skull förenklad och anpassad för att användas i det specifika fall som är aktuellt för detta arbete.
[2] Se bilaga A för detaljer kring de statistiska metoder som använts i detta arbete.

användas ihop med ett tröskelvärde som beskriver hur hög den summerade sannolikheten för en inmatning måste vara för att verifieras (Joyce & Gupta, 1990).

Viktad sannolikhet (Weighted Probability)

Enligt Guven & Sogukpinar (2003) är det en allmän uppfattning att vissa tangenttryckningar är mer unika än andra för olika användare. Exempelvis kan vissa användare ha ett karaktäristiskt sätt att trycka ned bokstavskombinationen ER där latency mellan E och R alltid är hög, medan andra latencies kan ha större variationer. De mer karaktäristiska mönstren kan på så sätt ges större vikt vid beräkning av hur lik en inmatning är referenssignaturen.

Figuren nedan visar hur latencies mellan olika inmatningar varierar i stabilitet. Latency mellan j-a har exempelvis stora variationer (140 ms) medan variationerna för latencies mellan e-s har väldigt små variationer (30 ms). De latencies som uppmäts för e-s kan således få en högre vikt än de för j-a.



Figur 6.- Latencies för tre olika inmatningar (representerade med fyllda cirklar) av ordet "james"

Precis som med icke-viktad sannolikhet beräknas ett värde på likheten mellan inmatningen *U* och referenssignaturen *R*. Det som skiljer modellerna är vikten *W* för varje latency eller duration[1].

$$Score(R, U) = \sum_{i=1}^{n} \Pr\ ob\ (\frac{U_i - \mu_{R_i}}{\sigma_{R_i}}) \times W_i$$

Där W_i är vikten för element *i* (exempelvis latency mellan j-a) (Monrose & Rubin, 2000, Guven & Sogukpinar, 2003).

Standardavvikelsen fyller redan samma funktion som vikten. Vikter är dock ett ytterligare sätt att ta till vara på karaktären i skrivsättsmönstret och kan sättas mer godtyckligt och efter andra förutsättningar än variationer för latencies eller durationer. Exempelvis kan det användas för att ge mer vikt åt ett specifikt latency som av någon anledning anses vara mer pålitlig (stabil) (Guven & Sogukpinar, 2003).

[1] Funktionen är förenklad för att passa arbetets specialfall och anpassad för högre läsbarhet.

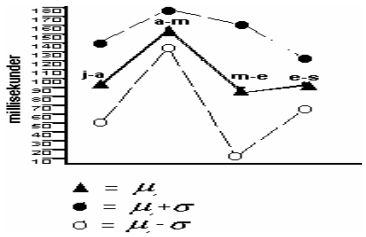
2.3 Tidigare implementeringar av Keystroke Dynamics

Ett stort antal metoder har använts för att implementera keystroke dynamics. Exempelvis har många varianter av neurala nätverk använts för att implementera keystroke dynamics (Bleha, Knopp & Obaidat, 1992). Neurala nätverk har för identifiering gett lyckade resultat på små databaser men ger extremt höga beräkningstider för varje ny användare som läggs till i systemet. Detta beroende på att hela nätverket måste tränas om då en ny användare läggs till i databasen (Monrose & Rubin, 2000). Neurala nätverk kräver också en stor träningsmängd (det vill säga många inmatningar för att skapa referenssignaturen) vilket kan vara svårt att motivera användare att ställa upp på (Yu & Cho 2003). Det är fullt möjligt att använda olika varianter av neurala nätverk för statisk verifiering vilket tidigare arbeten av exempelvis Yu & Cho (2003) visar. Generellt kan dock sägas att många metoder ej är direkt applicerbara för statisk verifiering.

Det finns inte utrymme att här gå igenom alla detaljer i alla olika implementationer av keystroke dynamics. De två typer av metoder som har mest relevans för denna rapport är statistiska metoder och fuzzy logic vilka beskrivs nedan.

2.3.1 Statistiska metoder

Olika varianter av viktad och icke-viktad sannolikhet används ofta som grund för statistiska metoder (Umphress & Williams, 1985, Joyce & Gupta, 1990, Monrose & Rubin, 2000). Umphress & Williams (1985) räknar i sin variant för kontinuerlig verifiering ut medelvärdet för latencies i referenssignaturen. μ (medelvärdet) + σ (standardavvikelsen) *1.5 sätter de som övre tillåten gräns för varje latency som skall verifieras. Dessutom används $\mu - \sigma * 1.5$ som undre gräns (Umphress & Williams, 1985). Ett uppmätt latency sägs klara gränserna om de är högre än den undre gränsen och lägre än den övre. I annat fall sägs att gränsen *missats*.



Figur 7. – Tillåtna övre och undre gränser för latencies som skall verifieras. Medelvärdet betecknas här med μ och standardavvikelsen med σ.

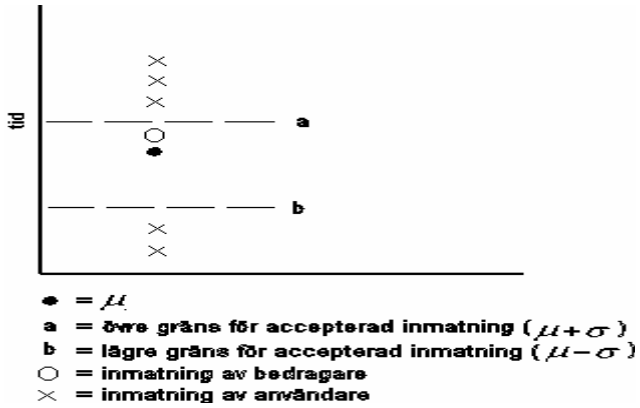
Vidare tillåts att 40% av alla latencies ej klarar dessa gränser. Detta kan ses som metodens variant på tröskelvärde (Umphress & Williams, 1985).

Joyce & Gupta (1990) använder en liknande statistisk metod för statisk verifiering. De använder förutom latencies från lösenordet även latencies från användarnamnet, som består av förnamn och efternamn. Med denna metod felklassificerades 17% av inloggningarna.

Statistiska metoder använder endast den specifika användarens tidigare input som referenssignatur vilket gör dem lämpliga att använda detta vid statisk verifiering (Cantú et al., 2002). Vid utveckling av nya metoder används ofta statistiska metoder som en referenspunkt (Joyce & Gupta, 1990).

Problem med statistiska metoder

Då ren statistik används så tas vissa karaktäristika ej hänsyn till vid klassificeringen. Om exempelvis en referenssignatur mäts med många extremt höga värden och många extremt låga värden för exempelvis en specifik latency så ger det upphov till en stor standardavvikelse, vilket i sin tur leder till en låg undre gräns och en hög övre gräns. En person vars latencies ligger nära medelvärdet kan då bli accepterad medan den korrekta användaren ej blir accepterad (Joyce & Gupta, 1990, Guven & Sogukpinar, 2003). I figur 8 illusteras problem förenklat för fem inmatningar (för ett enskilt latency) som utgör en användares referenssignatur.



Figur 8. – Problem med att använda statistik

Ytterligare en implikation av statistiska metoder är att om ett nytt mätvärde, vilket är högre än medelvärdet, läggs till så ändras både den övre och undre gränsen. Detta beror på att referenssignaturens medelvärde används för att avgöra dess grad av likhet med inmatningen som skall verifieras (Joyce & Gupta, 1990).

Angreppssättet med statistik gör att det går att beräkna sannolikheterna för att en inmatning felaktigt nekas tillgång till systemet (FRR) enligt vissa antaganden om fördelningen i datamängden (Umphress & Williams, 1985, Legget & Williams, 1988, Joyce & Gupta, 1990). Ofta går det dock inte att få korrekt kunskap om vilken fördelning en datamängd har. Detta speciellt då det gäller något så osäkert som olika användares beteende (Guven & Sogukpinar, 2003). Ett sätt att angripa detta problem som bättre kan acceptera denna typ av osäkerhet i datamängder är *fuzzy logic* (Zadeh, 1994).

2.3.2 Metoder med fuzzy logic

Statistiska metoder tillhandahåller ofta ingen adekvat modell för situationer där beslut måste tas utifrån approximativa datamängder (Yager & Zadeh, 1992). Där traditionella beräkningar använder sig av exakta, diskreta beslut utnyttjar fuzzy fogic icke-precisa mått, en slags suddighet för att tolerera ovisshet i datamängden (Zadeh, 1994). Fuzzy logic kan på så sätt ses som ett tillägg till klassisk logik för att bättre passa situationer där ett brus existerar i datamängden.

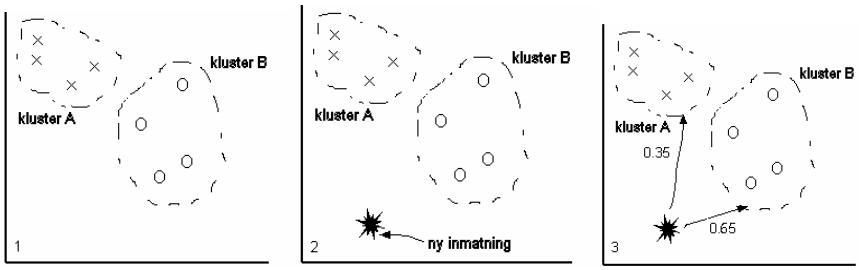
I vardagliga resonemang används ofta suddiga beskrivningar av omvärlden. Exempelvis: ”*det är en ganska bra fotbollsmatch*”. Med klassisk logik är det antingen en bra eller en icke-bra fotbollsmatch. Till skillnad från traditionell logik kan fuzzy logic ha olika grader av sanning i en proposition (Norvig & Russel, 1995). En fotbollsmatch kan således vara bra med grad 0.7. Fuzzy logic är på så vis bra för att representera vaga sanningar.

Fuzzy logic utnyttjar samma typ av ungefärliga värden vilka gör att vi människor kan förstå olika dialekter eller läsa olika typer av handstilar (Zadeh, 1994). Fuzzy logic har på grund av dessa fördelar använts mycket inom industrin och applicerats inom en rad olika områden (Zadeh, 1994).

Identifiering med hjälp av fuzzy logic

Bleha, Hussien & McLaren (1989) använder en utökning av mängdlära anpassad för fuzzy logic i sin keystroke dynamics applikation för identifiering. De delar upp olika inmatningar i kluster, där varje inmatning är medlem till en viss grad. Optimalt skall varje ny inmatning klassificeras till det kluster som tillhör den användare som gjort inmatningen.

I figuren nedan visas inmatningar för användare A och B (med varsitt kluster). Där X representerar inmatningar för A och O representerar inmatningar för B. När en ny inmatning skall identifieras jämförs medlemskapet för inmatningen i de olika klustren. Användaren identifieras till det kluster som har högst medlemskapsvärde för inmatningen.



Figur 9. – Identifiering av användares inmatning (att inmatningen är nära ett kluster ger den högt medlemskapsvärde)

För att kunna göra klassificeringar (identifieringar) måste medlemskapsvärdet som varje inmatning har i varje kluster först initieras (Bleha, Hussien & McLaren, 1989).

Initiering av medlemskapsvärden

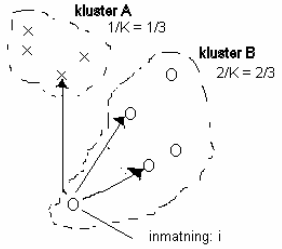
Bleha, Hussien & McLaren (1989) utvärderar *Fuzzy K-Nearest Neighbors* (Fuzzy-KNN) som initieringsalgoritm. Denna algoritm arbetar utifrån teorin att ett visst antal av de mest lika inmatningarna i referenssignaturen skall användas för att avgöra vilket kluster som varje ny inmatning tillhör. De mest lika inmatningarna i referenssignaturen till varje ny inmatning kallas för *grannar* (neighbors). Antalet grannar som skall användas måste anges i förväg och betecknas *K*. Därav namnet K-Nearest Neighbors (K- Närmsta Grannar).

Det initiala medlemskapsvärdet (ett reellt tal mellan 0.0 och 1.0) som en inmatning *i* har i kluster *j*, beräknas[1]:

$$\mu_j^{(i)} = \begin{cases} \max(\frac{m_j}{K}, 0.51), om\_inmatningen\_i\_tillhör\_klustret\_j \\ \min(\frac{m_j}{K}, 0.49), om\_inmatningen\_i\_ej\_tillhör\_klustret\_j \end{cases}$$

K : antalet grannar som skall användas i klassificeringen
m_j : antalet grannar (av de K närmsta grannarna till i) som tillhör kluster j

Varje kluster menas här vara en specifik användares alla inmatningar (vilket kan ses som referenssignaturen för användaren). Det vill säga om inmatningen är gjord av den användare som klustret tillhör kommer medlemskapsvärdet för inmatningen i det klustret att bli som minst 0.51. Om inmatningen tillhör en annan användare blir medlemskapsvärdet som mest 0.49 (Bleha, Hussien & McLaren, 1989).



Figur 10. – Medlemskapsvärdet initieras genom att finna de närmsta grannarna (ofta genom euklidiskt distansmått). I figuren är K = 3. Det vill säga de 3 närmsta grannarna till inmatning i används.

[1] Detta är en något annorlunda beskrivning av orginalmodellen. Detta för att öka läsbarheten i kontexten för detta arbete. Betydelsen av modellen är fortfarande samma som i orginalmodellen.

Klassificering

För att utföra klassificeringar i Fuzzy-KNN (det vill säga identifiera en inmatning till rätt kluster) utför Bleha, Hussien & McLaren (1989) följande beräkning för varje ny inmatning *U*:

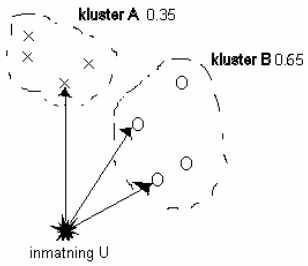
Medlemskapsvärde för inmatning *U* till kluster *j* = $\mu_j(U) = \dfrac{\sum_{i=1}^{K} \mu_j^{(i)} \left(\dfrac{1}{|U - R_i|^{2/(m-1)}} \right)}{\sum_{i=1}^{K} \left(\dfrac{1}{|U - R_i|^{2/(m-1)}} \right)}$

R : en vektor med de K närmsta grannarna till U. Det vill säga de inmatningar i referenssignaturen som är mest lika U (likheten mäts ofta med euklidiskt distansmått).

$\mu_j^{(i)}$: medlemskapsvärdet som den närmsta grannen i till inmatning U har för kluster j.

m: kallas för *fuzzifier* och är ett ofta ett tal mellan 1.0 och 2.0 vilket används för att göra gränserna för klassificeringar mer flytande. Det vill säga öka de medlemskapsvärden som är angränsande till två eller flera användares referenssignaturer (Bleha, Hussien & McLaren, 1989).

Användaren identifieras till det kluster där inmatningen ges högst medlemskapsvärde.



Figur 11. – De K närmsta grannarna till inmatning U hittas. I figuren är K=3

Med denna metod för klassificering lyckades Bleha, Hussien & McLaren (1989) uppnå en Error Rate så låg som 1 %.

Problem med fuzzy logic för verifiering

Vad som gör det svårt att överföra metoder som använder fuzzy logic (såsom Fuzzy-KNN) direkt till verifiering är att vid verifiering finns ingen direkt nytta med att jämföra inmatningen med alla andra användares referenssignaturer. Detta skulle kunna lösas exempelvis genom att låta varje användare ha ett gemensamt lösenord (eller en extra

gemensam textsträng vilket tidigare utnyttjats av bland annat Cantú et. al (2002)). Men det angreppssättet förändrar den vanliga inloggningsproceduren (användarnamn och lösenord). Dessutom kanske det endast existerar en användare i systemet vilket gör metoden direkt oanvändbar.

De suddiga gränser för vilka inmatningar som tillhör en användare bör dock kunna tas till vara i en modell för statisk verifiering utan att behöva använda ett gemensamt lösenord (eller extra textsträng).

3. Problembeskrivning

Som beskrivits i bakgrunden har de flesta metoder för keystroke dynamics utvecklats för identifiering eller kontinuerlig verifiering med mycket stort antal inmatningar som kan jämföras med referenssignaturer. Många tidigare metoder (med exempelvis neurala nätverk) har då de implementerats för statisk verifiering använt ett gemensamt lösenord för alla användare i systemet (Cantú, et al., 2002). Detta för att kunna använda referenssignaturerna från alla användare i systemet då en verifiering för en specifik användare skall utföras.

Keystroke dynamics är dock önskvärt att använda som en extra säkerhetsåtgärd utan att förändra den vanliga inloggningsproceduren (användarnamn och lösenord). Dessutom är det önskvärt att tiden det tar att skapa en referenssignatur för en användare är på en acceptabel nivå. Detta är ett problem för metoder som använder neurala nätverk eftersom dessa kräver stora träningsmängder (stort antal inmatningar för att skapa referenssignaturer).

Som påvisats i bakgrunden har statistiska metoder många nackdelar. Om den normala inloggningsproceduren skall behållas är dock alternativ som Fuzzy-KNN ej direkt applicerbara.

3.1 Mål

Målet med detta arbete är skapa en ny algoritm för statisk verifiering med keystroke dynamics som tar tillvara på de karaktäristika som användare har i sitt skrivsättsmönster (exempelvis möjliga stabila lutningar etc.). Detta genom att kombinera idéer från viktad sannolikhet och den klassificeringsmodell för fuzzy logic som används av Bleha, Hussien & McLaren (1989). Om algoritmen har högre prestanda i fråga om FAR och FRR än två vanliga sätt att implementera keystroke dynamics (statistisk metod och euklidiskt distansmått) så antas att den nya algoritmen är en lovande algoritm för keystroke dynamics.

3.1.1 Delmål 1

Ett första delmål är att kartlägga sådana karaktäristika i användares skrivsättsmönster som kan användas för utveckling av en ny algoritm för keystroke dynamics.

3.1.2 Delmål 2

Detta delmål är att utveckla en ny metod för statisk verifiering med keystroke dynamics som tar till vara på den information som kommer fram i analysen för delmål 1.

3.1.2 Delmål 3

Ett tredje delmål är att analysera prestandaskillnader mellan den nyutvecklade algoritmen och de metoder som använder ren statistik och de metoder som använder euklidiskt

distansmått, vilka är de mest grundläggande metoderna och därmed är lätta att implementera och förklara.

Om den nya metoden ej har högre prestanda än metoder som använder ren statistik och euklidiskt distansmått som klassificeringsmodell, skall man kunna dra slutsatsen att den nya metoden kan uteslutas från mängden av lovande metoder och framtida ansträngningar bör ligga på annat håll.

3.2 Avgränsning

Som tidigare nämnts gör Ilonen (2003) skillnad på två olika sätt att använda Keystroke dynamics; verifiering och identifiering. Detta projekt begränsar sig till att testa algoritmerna vid verifiering. Ofta används både lösenord och användarnamn som input till metoderna (Joyce & Gupta, 1990). Den begränsning som görs här är att endast ett ord (lösenordet) används som extra säkerhetsmått. Om endast lösenordet används blir det lättare att behålla existerande design på inloggningsprocedurer när man väl bestämmer sig för att implementera metoden som extra säkerhet. PIN-koder och liknande har dessutom ofta inget användarnamn. Även om användarnamnet skalas bort från säkerhetskontrollmekanismen görs antagandet att tendenser i prestandaskillnader ändå kommer att kunna utläsas från testresultaten. Tidigare tester med 6-siffriga PIN-koder har lyckats urskilja tydliga mönster i knappsatsanvändningen (Furnell & Ord, 2000). Detta bör tyda på att endast lösenordet är tillräckligt för att kunna göra tydliga klassificeringar hur olika användare använder knappsatsen. Men framförallt för att se tendenser i skillnader mellan olika metoder för verifieringsprocessen.

En ytterligare begränsning är att endast standardutrustning kommer användas för testning av de olika metoderna. Detta är en naturlig begränsning då stort testmaterial kommer att behöva samlas in. Att utveckla specialdesignade knappsatser är allt för tidskrävande. Med standardutrustning menas att testen utförs med ett på förhand valt operativsystem (Windows XP) samt att standardtangentbord används.

Standardutrustning implicerar ytterligare en begränsning. Monrose & Rubin (1999) skriver att de tre parametrar som är möjliga att användas som input för keystroke dynamics är avståndet i tid mellan knapptryckningar (latency), kraften som används för att trycka ned tangenterna, samt hur länge personen håller ned varje tangent (duration). Med standardutrustning är inte trycket på tangenterna mätbart och kommer därför ej användas som mätdata. Ingen av de metoder som undersöks i denna rapport har använt sig av mätning av trycket på tangenterna. Perrig, Song & Venable (1997) identifierar ytterligare möjliga mätdata till algoritmerna nämligen användandet av shift-tangenterna i förhållandet till bokstäverna. Perrig, Song & Venable (1997) använde i sina test även användandet av numeriska tangenterna som speciell mätdata. Denna ansats kommer inte beaktas i detta projekt och ej beaktas alls då algoritmerna implementeras.

Eftersom denna rapport endast utvärderar algoritmer för verifiering kommer endast input från den äkta användaren (det vill säga användaren som äger lösenordet) användas vid träning. Med andra ord kommer träning endast att ske på sann input.

3.3 Förväntat resultat

Det resultat som förväntas är att den nyutvecklade algoritmen har högre prestanda (i fråga om både FAR och FRR) gentemot de övriga algoritmer som skall implementeras. Detta beroende på att metoden, som inspireras av idéer från viktad sannolikhet och fuzzy logic, intuitivt tycks fånga en mer samlad bild av det ofta instabila användningsmönstret hos användaren.

4. Metod

Detta kapitel beskriver möjliga metoder, som kan användas för att uppnå delmålen och vilka av dessa metoder som valts.

För att få fram en ny metod för statisk verifiering med keystroke dynamics har olika tillvägagångssätt analyserats. Problemet ligger i att visa vilka unika skrivsättsmönster som olika människor har, för att sedan kunna utnyttja denna kunskap i utvecklingen av en ny metod.

4.1 Metod för delmål 1

För att uppnå delmål 1, det vill säga; *finna mätbara karaktäristika i användares skrivsättsmönster* (i latencies och durationer), är ett experiment som möjliggör observation av skrivsättsmönstret en bra utgångspunkt. Tidigare arbeten har redan utfört sådana experiment, vilket skulle ha varit väldigt behändigt att få ta del av. Mätvärden från tidigare arbeten visade sig dock vara väldigt svårt att få tag på vilket gör det nödvändigt att utföra egna experiment.

Experimenten kommer av resursskäl att utföras på ett snabbt och resurssnålt sätt. Endast personer i författarens omgivning kommer att kontaktas för att deltaga i experimenten. Detta är en brist för rapportens första delmål, vilket kan skapa förutsättningar för feltolkningar. Detta kommer dock att ge sig till känna i delmål 3 då experiment skall utföras med en mer representativ grupp. Om experimentet ej ger tillförlitliga resultat gör det att delmål 2, utvecklingen av algoritmen, ej kan ta tillvara på skrivsättsmönstret lika bra och därmed påverkar resultatet i delmål 3.

Umphress & Williams (1985) skriver att vana datoranvändare troligtvis har mer pålitliga mätvärden, då de har tränat in sin egen skrivrytm till skillnad från ovana datoranvändare. Umphress & Williams (1985) använder själva endast vana datoranvändare (17 programmerare). Denna rapport har med tanke på detta valt att endast låta vana datoranvändare vara deltagare i det första experimentet. Furnell & Ord (2000) poängterar att det är viktigt att samla in mätvärdena över en längre tid, för att på så sätt kunna få bort variationer i användares skrivsättsmönster. Dessa variationer kan bero på saker som sinnesstämning, trötthet eller liknande. Furnell & Ord (2000) samlade själva in sina mätvärden under 6 månaders tid. Det finns ingen direkt möjlighet i detta projekt att göra så. Antagandet som görs i det första experimentet i denna rapport är att sinnesstämning och trötthet ej påverkar de tendenser i mönster som skall analyseras för delmål 1.

4.2 Metod för delmål 2

Delmål 2 går ut på att skapa en ny metod för keystroke dynamics med hjälp av den analys som gjordes i delmål 1. Algoritmen skall även använda resultat från tidigare forskning inom området. För att få reda på vilka metoder som tidigare använts ansågs en litteraturstudie vara ett naturligt angreppssätt.

Som stöd i analysen kommer ett datorprogram skapas med möjligheter att visa grafer och statistik för inmatningarna. En möjlig metod hade kunnat vara att använda existerande programvara såsom MatLab för att grafiskt representera datamängden. Detta uteslöts då det ansågs gå snabbare att specialdesigna verktyg (grafer etc.), för just den information som skall undersökas, om ett eget datorprogram skapas för ändamålet. Det egna programmet blir också en del av själva analysen. För att exempelvis titta närmare på ett specifikt fenomen blir det enkelt att lägga till specialdesignad funktionalitet i programmet.

4.3 Metod för delmål 3

Delmål 3 går ut på att jämföra prestanda mellan ett par existerande metoder och den nyutvecklade.

Prestanda ur olika perspektiv

Prestanda för en algoritm kan ses ur olika perspektiv. Ett perspektiv är *effektivitet*. Ett möjligt sätt att jämföra prestanda hos olika metoder ur effektivitetsperspektivet, är att genomföra ett komplexitetstest. Detta har dock valts bort då de metoder som kommer att jämföras ej tros ha någon högre komplexitet.

Ytterligare ett sätt att mäta prestanda för olika implementationer skulle kunna vara ett benchmarktest där antalet klassificeringar per sekund skulle kunna jämföras. Beräkningstiderna tros dock bli väldigt låga vilket gör att möjliga prestandaskillnader mellan de olika metoderna i detta avseende ej bör vara avgörande i valet av metod vid implementering. Prestandajämförelsen kommer i stället utgå ifrån perspektivet *funktionalitet*, vilket i detta fall då blir andelen korrekta klassificeringar (verifieringar). De två prestandamåtten utifrån det perspektivet är FAR och FRR vilka beskrivs i bakgrundskapitlet 2.1.4.

Tidigare arbetens experiment

Ett möjligt sätt att jämföra en nyutvecklad metod med existerande metoder är att använda samma mätvärden som används i andra arbeten. Detta medför att inga nya experiment behövs utöver prestandatest på den nya metoden vilket är väldigt tidseffektivt. Det skulle också betyda att inga andra metoder behöver implementeras vilket sparar en hel del tid. Detta alternativ har dock uteslutits då tidigare insamlat material användbart för rapportens ansats ej finns tillgängligt[1].

[1] Författare till frekvent refererade forskningsartiklar inom området kontaktades för att få tillgång till deras material. Aykin Guven (Guven & Sogukpinar, 2003) var den ende som skickade sitt experimentmaterial. De hade dock inte sparat latencies och durationer för inloggningsförsöken av bedragare, ej heller vilka lösenord som användes vilket gör att denna rapport ej kan använda sig av hans material. Förfrågningarna gjordes med e-post och för de som ej svarade finns en stor risk att meddelandet ej gick fram. En av de tillfrågade svarade med att han ej hade kvar sitt material.

Eget experiment

Ett annat sätt att jämföra prestanda mellan den nyutvecklade metoden och existerande metoder är att samla in eget material.

För att kunna visa prestandaskillnader krävs att en stor mängd mätvärden samlas in från olika användare. Den metod som har mest gemensamt med den nyutvecklade i fråga om funktion är antagligen Bleha, Hussien & Slivinskys (1990) metod. Syftet med deras metod är statisk verifiering med en vanlig inloggningsprocedur (användarnamn och lösenord). De använder förvisso både förnamn och efternamn som lösenord, vilket gör att användarnas lösenord är mellan 11-17 bokstäver.

Bleha, Hussien & Slivinsky (1990) utförde sina tester med 14 användare under en fyraveckors period. Denna metod ger mycket material att analysera och är på så sätt väldigt bra. Detta projekt kan dock av resursskäl ej utföra tester under en fyraveckorsperiod. I och med att syftet med delmål 3 är att jämföra prestanda mellan olika metoder är tanken att tiden mellan varje testtillfälle ej spelar en avgörande roll då endast metodernas prestanda relativt till varandra skall analyseras.

5. Analys av skrivsättsmönster

Detta kapitel beskriver vilka mönster som aktivt söktes i datamängden, hur analysen av skrivsättsmönster genomfördes och vilka mönster som slutligen kunde påvisas.

5.1 Förutsättningar för analysen

Det som först och främst söktes efter i skrivsättsmönstret hos användare var:

- Om användare ofta skriver ett ord med högre hastighet desto mer träning på ordet de har.
- Om stabila lutningar existerar.

För att besvara den första punkten krävs att mätningarna är kronologiskt ordnande. Om detta är gjort så är det enkelt att jämföra de tidiga uppmätta inmatningarna med de senare. Ifall hastigheten för inmatningarna stiger efter hand så kan detta tas till vara i en metod för keystroke dynamics.

För att visa den andra punkten krävs att en analys av närliggande latencies (och durationer) görs. Exempelvis om lösenordet är ”kalle” så jämförs de latencies för k-a med latencies för a-l.

Även andra tänkbara mätbara karaktäristika försökte identifieras i skrivsättsmönstret genom att analysera mätdata i experimentet. Detta utfördes genom experimenterande med olika datorprogram.

5.2 Genomförande av experiment

För att ha möjlighet att på ett enkelt sätt samla in information om latencies och durationer för användares lösenord så skapades ett datorprogram för detta ändamål.

Sju vana datoranvändare ur en klass av datastudenter ombads skriva in ett lösenord 70 gånger i programmet varav de första 30 används som referenssignaturer och de sista 40 som ytterligare testmaterial. Användarna instruerades att använda det tangentbord som de hade mest vana att skriva på. Detta för att de inträngade mönstren tydligare skulle kunna urskiljas. För att de skulle ha möjlighet att använda sitt *mest använda* tangentbord så skickades datorprogrammet ut med e-post till samtliga deltagare tillsammans med noggranna instruktioner.

Som lösenord valdes personens förnamn. Detta på grund av att användare tros ha skrivit sitt eget namn ett stort antal gånger tidigare och på så sätt redan tränat in ett tydligt skrivsättsmönster. Det optimala borde sannolikt ha varit att låta dem skriva sitt mest använda lösenord. Det angreppssättet uteslöts dock då det misstänktes påverka motivationen för användarna att ställa upp i testet. Innan skapandet av referenssignaturen fick de veta att de skulle skriva in sitt lösenord på det sätt som de vanligen gör. Detta för att ingen deltagare i testet skulle skapa en icke-karaktäristisk referenssignatur.

5.3 Resultat av experiment

Detta kapitel beskriver analysen av de mätvärden som samlades in i första experimentet. Varje underkapitel beskriver en tendens i datamängden.

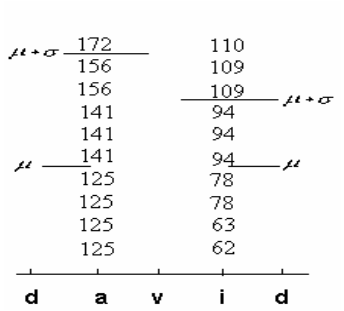
5.3.1 Skillnad i stabilitet mellan inmatningarnas latencies och durationer

Vissa durationer och latencies för en användares inmatningar tycks enligt de initiala experimenten variera mer än andra. Nedan visas ett exempel på detta:

Tabell 1 – durationer i millisekunder för ordet ”david”

inmatning	**d**	**a**	**v**	**i**	**d**
1	78	125	78	63	78
2	94	141	78	94	78
3	125	141	94	78	125
4	125	141	78	109	94
5	110	125	78	62	94
6	156	156	78	78	188
7	94	125	93	94	94
8	125	156	94	110	94
9	125	125	94	109	125
10	125	172	109	94	94

I exemplet ovan så varierar durationen för ’a’ mellan 125 och 172 och har en standardavvikelse på 15.6 och ett medelvärde på 140.7. Standardavvikelsen för ’i’ är 17.2 och medelvärdet på 89.1. Om standardavvikelsen ihop med medelvärdet används som övre och undre tillåten gräns för varje tangentnedslag (likt vissa statistiska metoder som beskrivits i bakgrunden) kommer gränsen för ’a’ missas 1 gång medan gränsen för ’i’ missas 3 gånger. Detta visas i figur 12.



Figur 12.- Illustration över hur de övre gränserna för'a' respektive 'i' missas olika antal gånger då medelvärde och standardavvikelse används för att sätta dem.

Detta tyder på att vissa element i mätvärdena har olika stabilitet vilket kan utnyttjas i en algoritm. Fenomenet utnyttjas också på det sättet i viktad sannolikhet (se kapitel 2.2.4).

5.3.2 Stabila lutningar

Vissa användare misstänks ha stabila förhållanden i skillnaden mellan olika latencies (Joyce & Gupta, 1990), det vill säga det som denna rapport benämner stabila lutningar.

Samtliga deltagare i det initiala experimentet tycks ha liknande förhållanden i sina inmatningar för både latencies och durationer. Nedan visas en tabell över antalet stabila lutningar för varje deltagare i det initiala experimentet. I tabellen visas endast de stabila lutningar vilka förekommer i samtliga av de 20 sista inmatningarna i referenssignaturen.

Tabell 2. – antalet stabila lutningar i de olika användarnas lösenord

Användare	**Bokstäver i lösenord**	**S.L. för Durationer**	**S.L. för Latencies**
A	5	1	0
B	10	2	2
C	5	2	1
D	6	1	1
E	5	0	3
F	6	0	1
G	5	1	1

Det är uppenbart att chansen för stabila lutningar ökar för användare med längre lösenord. Notera dock att användare E har 3 sådana stabila lutningar för latency vilket är

det största antalet stabila lutningar bland alla användare. Användare E har således en stabil lutning mellan samtliga sina latencies, vilket tyder på att just den användaren har ett väldigt tydligt skrivsättsmönster.

5.3.3 Mönstret bryts oftare genom högre tider

Användarna i det initiala experimentet tycks bryta sitt normala skrivsättsmönster på ett relativt förutsägbart sätt. Användarna bryter oftare sitt skrivsättsmönster genom att vissa tider för latencies och durationer blir högre än i referenssignaturen. Det verkar dock vara så att de mer sällan blir lägre än i referenssignaturen. Detta skulle kunna förklaras av att vid tillfälliga distraktioner så stannar användaren en kort stund för att tillfälligt koncentrera sig på distraktionen (exempelvis att en dörr stängs igen eller liknande) (Garcia, 1986). Detta ger då högre tider än normalt.

Användare tycks dock ha en tendens att få lägre och lägre tider för latencies ju mer träning på lösenordet de har. Detta kan tyckas vara självklart men bekräftas av experimentet. Tabellen nedan visar ett exempel på detta (tabellen skall endast ses som en illustration över hur de flesta användare tycks skriva *”snabbare och snabbare”*).

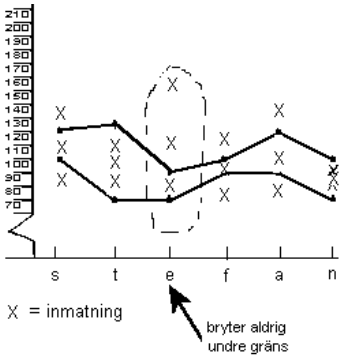
Tabell 3.- Medelvärden för latencies mellan första och andra bokstaven i lösenordet för alla användare. De flesta användare skriver snabbare och snabbare. Användare B har dock omvänt beteende (markerat med fet stil).

Användare	**Medelvärde de första 10**	**Medelvärde sista 10**
A	188.4	131.3
B	**329.9**	**370.3**
C	83.4	78.3
D	501.5	425.1
E	349.5	270.7
F	114.2	81.0
G	246.8	186.0

5.3.4 Karaktäristiskt sätt att bryta skrivsättsmönstret

Vid de tillfällen som användarna i det initiala experimentet bröt sitt normala skrivsättsmönster gjordes detta ofta på ett karaktäristiskt sätt. Exempelvis fanns en användare i experimentet vars latencies för de första tre bokstäverna hade väldigt lika mätvärden i samtliga inmatningar medan latency mellan de sista två bokstäverna i en fjärdedel av inmatningarna låg väldigt mycket högre än de övriga tre fjärdedelarna. Viktad sannolikhet gör normalt sett inte skillnad på om den övre eller undre gränsen

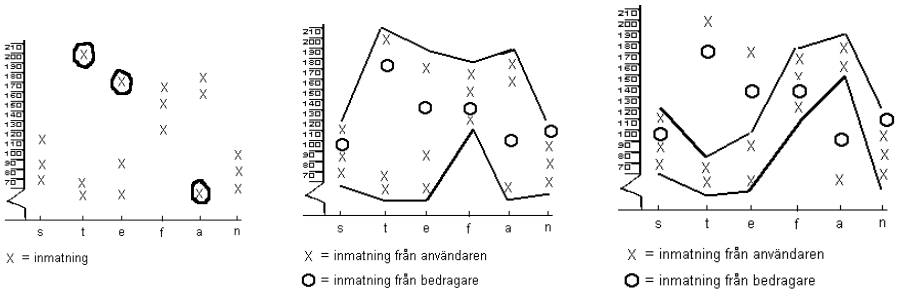
missas av inmatningen. Detta borde dock kunna tas tillvara på i en metod för keystroke dynamics.



Figur 10.- Tre inmatningar för lösenordet ”stefan” illustrerar ett karaktäristiskt sätt att bryta sitt mönster på

5.3.5 Tillfälliga avvikelser

Då ett skrivsättsmönster för en användare skall uppfångas är det vitalt att försöka avgöra exakt vad som är en del av användarens naturliga skrivsättsmönster (Garcia, 1986). Ofta förekommer ett fåtal värden i datamängden som tydligt avviker från de andra värdena. Dessa *tillfälliga avvikelser (outliers)* i referenssignaturen kan tänkas bero på en tillfällig distraktion från användaren (Garcia, 1986). Om dessa två värden används som en del i mönstret kan det göra att det naturliga skrivsättsmönstret för användaren ej fångas upp lika bra. Detta kan i sin tur göra att inloggningsförsök från bedragare felaktigt accepteras i större utsträckning än om tillfälliga avvikelser plockas bort. I figurerna 13 a-c visas exempel på konsekvenser tillfälliga avvikelser i referenssignaturen kan ge.



Figur 13. – a) Tre tillfälliga avvikelser, i de tre första inmatningarna, är inringade. b) Visar möjlig konsekvens med dessa i referenssignaturen. Bedragarnas inmatningar liknar mönstret för användaren då tillfälliga avvikelser räknas med. c) Visar att om tillfälliga avvikelser tas bort minskar risken för att bedragare lyckas logga in.

För statistiska metoder finns det stor poäng att ta bort dessa tillfälliga avvikelser då de påverkar medelvärdet och standardavvikelsen väldigt mycket (Legget & Williams, 1988).

5.4 Användning av analysen för utveckling av algoritm

Föregående kapitel 5.3 beskriver de mätbara karaktäristika som pekades ut av analysen av det insamlade materialet i delmål 1. En algoritm för keystroke dynamics bör ta tillvara på samtliga av dessa fem specifika karaktäristika för att avgöra skillnader mellan skrivsättsmönster. Utvecklingen av algoritmen bör således styras genom de karaktäristika som kunde pekas ut och följaktligen designas så att alla fem karaktäristika behandlas.

6. Utveckling av Algoritm

Detta kapitel beskriver utvecklingen den nya algoritmen. Beskrivningen görs i separata underkapitel för att underlätta förståelsen[1].

Algoritmen som utvecklas i denna rapport kallas för *Suddig C*. Detta för att den är starkt inspirerad av fuzzy logic (därav *Suddig*) och innehåller tre steg (därav *C*).

I utvecklingsstadiet av algoritmen gjordes en analys av resultatet från det första (initiala) experimentet vilket användes som motivering för algoritmens egenskaper (se kapitel 5.3). Det vill säga; varje av algoritmens steg är ett försök att ta tillvara på ett karaktäristika vilken upptäcktes i analysen av det initiala experimentet. Dessa karaktäristika är:

- Tillfälliga avvikelser
- Skillnad i stabilitet mellan inmatningarnas latencies och durationer
- Mönstret bryts oftare genom högre tider
- Karaktäristiskt sätt att bryta skrivsättsmönstret
- Stabila lutningar

Utvecklingen styrs således med tanke på att kunna hantera (ta till vara på) dessa fem karaktäristika. Även tidigare influenser från befintliga algoritmer har styrt utvecklingen vilket förklaras löpande i kommande kapitel. Algoritmen beskrivs i olika steg för att på ett enkelt sätt förklara olika karaktäristikas betydelse för funktionen av algoritmen.

För varje verifieringssteg måste den egenskap i skrivsättsmönstret som skall verifieras först *initieras*. Denna initiering fungerar som en notering av referenssignaturens egenskaper och behöver endast utföras en gång. Verifieringen utförs dock varje gång användaren loggar in.

Algoritmens initieringsdel

I algoritmen initieras tre olika steg som senare används vid verifieringen. Det vill säga varje steg i initieringsdelen har ett tillhörande steg i verifieringsdelen.

1. Initiering av initialt skrivsättsmönster
    a. Ta bort tillfälliga avvikelser
    b. Beräkna initialt skrivsättsmönster
    c. Beräkna viktighetsvärden
2. Initiering av stabila lutningar
3. Initiering av brutet skrivsättsmönster

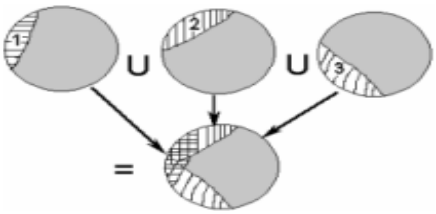
[1] I detta kapitel görs ett försök att beskriva algoritmen så enkelt som möjligt för att underlätta förståelsen. Faktisk implementation av algoritmen (i datorprogram) bör ske på ett mer optimerat sätt (i fråga om minnesanvändning och antal beräkningar) än vad som beskrivs i kapitlet.

Algoritmens verifieringsdel

Varje verifiering av användare i Suddig C utförs i tre steg:

1. Verifiera initialt skrivsättsmönster
2. Verifiera stabila lutningar
3. Verifiera brutet skrivsättsmönster

Om inmatningen ej verifieras vid ett steg utförs ej konsekutiva steg, algoritmen avslutas och inmatningen betraktas av systemet som ett inloggningsförsök av en bedragare. Detta schema visas i figur 14.



Figur 14.- Venndiagram över vilka inmatningar som nekas av systemet. Den grå arean representerar verifierade inmatningar. De nekade inmatningarna är således unionen av de som nekas i steg 1, 2 och 3.

Uppdelning av inmatningarna i referenssignaturen

För att kunna förklara denna algoritm på ett enkelt sätt införs två nya begrepp; *initial-referens* (R) och *träningsmängd* (T). Med R menas referenssignaturens första (*RSize_Constant*) inmatningar. Med T menas resterande inmatningar i referenssignaturen. Det vill säga referenssignaturen delas upp i två delar där R är de första upp till en konstant (*RSize_Constant)* och T de resterande.

R = inmatning 1 till *RSize_Constant*

T = inmatning (*RSize_Constant* + 1) till N

Där N är antalet inmatningar i referenssignaturen. Detta kapitel beskrivs verifieringsstegen med tillhörande initieringssteg i samma underkapitel (för varje steg) i syfte att underlätta förståelsen.

6.1 Verifiera initialt skrivsättsmönster

Detta kapitel beskriver de steg i algoritmen som tas för att utföra verifieringen av initialt skrivsättsmönster. För att verifieringen skall kunna utföras måste först en initiering göras. Denna initiering beskrivs först, följt av verifieringen i varje underkapitel.

6.1.1 Initiering av initialt skrivsättsmönster

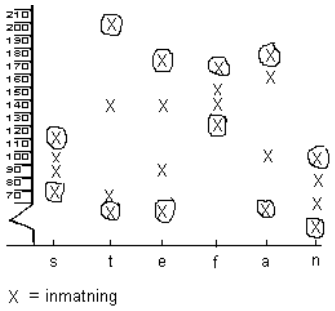
Initieringen av initialt skrivsättsmönster sker i tre steg vilka beskrivs nedan.

Steg 1: Ta bort tillfälliga avvikelser

Detta steg har sin motivering i kapitel 5.3.5. Många tidigare metoder tar bort tillfälliga avvikelser på något sätt. Suddig C tar endast bort tillfälliga avvikelser i R. Detta beroende på att Suddig C förväntas ha en hög kapacitet att acceptera dessa tillfälliga avvikelser som en naturlig del i användarens skrivsättsmönster. De tros även förbättra prestandan för Suddig C eftersom de kan peka ut vilka tangentnedslag för användaren som har oförutsägbara latencies eller durationer.

Beskrivning hur tillfälliga avvikelser tas bort

De högsta och lägsta uppmätta latencies och durationer tas bort från R (initial-referensen) (genom att de ersätts med medelvärdet). Anledningen till att de ersätts med just medelvärdet är för att vara säker på att de ej längre är högst (alternativt lägst), vilket är det enda som spelar någon roll för vidare beräkningar. Tillfälliga avvikelser behandlas endast i R beroende på att möjliga avvikelser i T fyller en funktion i senare steg där T används. I figur 15 visas ett exempel för fyra inmatningar (*RSize_Constant=4*).



Figur 15. – Tillfälliga avvikelser vilka skall tas bort från R visas inringade.

R med de av funktionen borttagna avvikelserna kommer att betecknas R^I. Pseudokoden för hur dessa avvikelser tas bort presenteras nedan.

Pseudokod

```
RemoveOutliers() : derives R^I from R
Begin
        For i:=1 to password_length do
        Begin
                RemoveOutliersInColumn(R_durations , i)
```

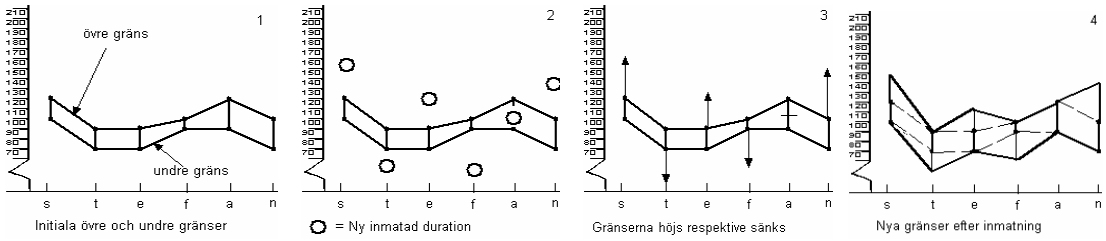
```
        End

        For i:=1 to (password_length -1) do
        Begin
                RemoveOutliersInColumn(R_latencies, i)
        end

end

RemoveOutliersInColumn(R, I)
Begin
        For i:=0 to |R| do
        Begin
                Mean := GetMean(R, i)

                J := Find_highest(R, i)
                R[j][i] := Mean
                J:= Find_lowest(R, i)
                R[j][i] := Mean
        End
End

Find_highest(R, i)
Begin
        highest := 1

        For a := 2 to |R| do
        Begin
                If R[a][i] > R[highest][i] then highest := a
        End

        Return highest
end

GetMean(R, i) : returns Mean value
Begin
        sum:= 0

        For a:=1 to |R| do
        Begin
                sum+= R[a][i]
        End

        Return (sum/ |R|)
End
```

Steg 2: Beräkna initialt skrivsättsmönster

Detta steg är ett försök att undvika det problem som finns med statistiska metoder där medelvärdet och standardavvikelsen används som övre och undre gränser (se 5.3.1 och 2.3.1).

Övre gränsen sätts initialt till den duration med högst värde i R^I. Likaså sätts den undre gränsen till den lägst uppmätta durationen i R^I. Samma procedur utförs för latencies.

Utifrån R^I skapas sedan R^{II} på så sätt att T (resterande inmatningar som utgör referenssignaturen) söks igenom inmatning för inmatning och varje gång en undre eller övre gräns ej uppnås höjs respektive sänks gränsen mot det nya högre eller lägre värdet (se figur 13).



Figur 13. – Övre och undre gränser för durationer förändras efter ny inmatning

Pseudokod

//Denna algoritm utförs för både latencies och durationer, men för läsbarhetens skull antar algoritmen //nedan att R^I endast innehåller durationer.

GetR^{II}(initial-referens: R^I) : returns R^{II}
Begin
For i:= 1 **to** *password_length*
Begin
$R^{II}_{highest}[i]$ = highest(R^I, i)
$R^{II}_{lowest}[i]$ = lowest(R^I [i], i)
end

For i:= 1 **to** *password_length* **do**
Begin
For j:= 1 **to** |T | **do**
Begin
// K och K2 är konstanter (avgör hur mycket en
//gräns höjs/sänks vid varje iteration)
incr = | T[j][i] - $R^{II}_{highest}[i]$ | / *K*
decr = | T[j][i] - $R^{II}_{lowest}[i]$ | / *K*

If T[j][i] > $R^{II}_{highest}[i]$ **then** $R^{II}_{highest}[i]$ += min(incr, *K2*)
Else if T[j][i] < $R^{II}_{lowest}[i]$ **then** $R^{II}_{lowest}[i]$ -= min(decr, *K2*)
end
End

Return R^{II}
End

Highest(R^I, i)
Begin
H **:=** $R^I[1][i]$

For j:=2 **to** | R^I |
Begin

```
                if R^I[j][i] > H then H := R^I[j][i]
        end

        return H
end

Lowest(R^I, i)
Begin
        L := R^I[1][i]

        For j:=2 to | R^I |
        Begin
                if R^I[j][i] < L then L := R^I[j][i]
        end
        return L
end
```

Detta skapar diskreta gränser för vilka tider varje latency och duration i varje inmatning måste befinna sig. För att dra nytta av de egenskaper som fuzzy logic besitter, där en inmatning exempelvis skulle kunna tänkas vara *ganska lik* referenssignaturen, blir nästa steg att göra gränserna mer suddiga.

Steg 3: Beräkna viktighetsvärden

Steg 3 syftar till att dra nytta av de egenskaper som viktad sannolikhet besitter genom att använda lite av den idé som finns i Bleha, Hussien & McLarens (1989) metod; fuzzy K-NN. Detta kan ses som en fusion av Fuzzy K-NN med viktad sannolikhet, då metoden har vissa likheter med båda. Idén med viktad sannolikhet kan motiveras av att datamängden ej antar normalfördelning. Se ytterligare beskrivning och motivering i kapitel 5.3.1.

Likt initieringen av medlemskapsvärden i fuzzy K-NN delas ett slags *viktighetsvärde* ut till varje latency och duration. Detta värde varierar utifrån hur karaktäristiskt ett mätvärde (för en specifik latency eller duration) är för användaren som referenssignaturen tillhör och fungerar på så sätt som en slags viktighetsstämpel. Detta är inspirerat av viktad sannolikhet som utvärderats av bland annat Monrose & Rubin (1997), Monrose & Rubin (2000) och Guven & Sogukpinar (2003) och är till för att ta hänsyn till att olika mätvärden kan ha olika signifikans för att klassificera en inmatning korrekt.

För att åstadkomma detta så görs ett försök att anpassa klassificeringsalgoritmen Fuzzy K-NN så att den fungerar för verifiering. Varje duration och latency får ett värde mellan 0.0 och 1.0 på hur karaktäristiska de är för användaren. Där 0.0 är väldigt lite karaktäristiskt och 1.0 mycket karaktäristiskt. Detta värde blir en slags motsvarighet till ”vikten” i viktad sannolikhet och medlemskapsvärdet i Fuzzy K-NN. Viktighetsvärdet *F* beräknas:

$$F_{q_i} = \max(1 - \frac{(d_{q_i})^m}{|T|}, 0)$$

q : De karaktäristika som används. I detta fall antingen latency eller duration.

i: En enskild duration eller latency

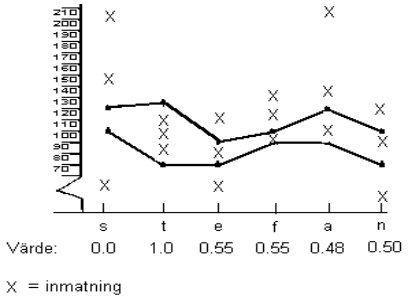
Hq_i : Den högre gränsen i R^{II} (för exempelvis [duration: 3] är *q*=duration och *i*=3).

Lq_i : Den lägre gränsen i R^{II}.

d_{q_i} : Antalet element i T för q som är mindre än Lq_i eller större än Hq_i.

m : En på förhand vald konstant som fungerar som en typ av fuzzifier vilken fyller liknande funktion som i fuzzy K-NN.

Hur karaktäristiskt en duration eller latency är för en användare avgörs på så sätt genom att använda de tidigare övre och undre gränserna (i R^{II}). Om exempelvis en duration ofta klarar de övre och undre gränserna får den ett högt värde. Om däremot en duration sällan är mellan de övre och undre gränserna får den ett lågt värde.



Figur 14.- Viktighetsvärden för olika durationer av ordet ”stefan”

Dessa värden används sedan för att avgöra likhetsgraden mellan en ny inmatning och referenssignaturen.

6.1.2 Verifiering

Verifieringen av varje inmatning sker i fem faser, vilka beskrivs nedan:

1. Summera samtliga viktighetsvärden för latencies (i figur 14 så skulle det bli 0.0+1.0+0.55+0.55+0.48+0.50). Ta detta värde gånger ett tröskelvärde (mellan 0.0 och 1.0) och spara i en variabel *Lf_sum*.
2. Summera samtliga viktighetsvärden för durationer. Ta detta värde gånger ett tröskelvärde (mellan 0.0 och 1.0) och spara i en variabel *Df_sum.*
3. Summera de värden vars gränser för durationer i R^{II} vilka klaras av inmatningens durationer. Denna summa sparas i en variabel *Dsum*.
4. Summera de värdena vars gränser för latencies i R^{II} vilka klaras av inmatningens latencies. Denna summa sparas i en variabel *Lsum*.
5. Om $(Lf\_sum + Df\_sum) \leq (Lsum + Dsum)$ så accepteras inmatningen i detta steg och algoritmen fortsätter till nästa steg. I annat fall nekas åtkomst till systemet.

Pseudokod för verifiering

```
// LatencyThreshold och DurationThreshold är här tröskelvärdena
// för latencies och durationer
VerifyInitialPattern(input) : returns TRUE if verified else FALSE
Begin
        FsumL := 0
        FsumD := 0
        Lsum :=0
        Dsum :=0

        CalculateSubPatternLatencies(input, lowvalue_importance_value_for_latencies, FsumL, Lsum)
        CalculateSubPatternDurations(input, lowvalue_importance_value_for_durations, FsumD, Dsum)
        Fsum := (FsumL * LatencyThreshold) + (FsumD * DurationThreshold)

        If Fsum <= (Lsum + Dsum) then return TRUE

        Return FALSE
End

// här menas att R^II_highest är R^II_highest för latencies, likaså är R^II_lowest för latencies
// detta är så med tanke på läsbarheten
CalculateSubPatternLatencies(input, lowvalue_importance, &Fsum, &sum)
Begin

        For i:= 1 to |input| do
        Begin
                Fsum += Latencies_importance_value[i]
                If input [i] <= R^II_highest[i] and input[i] >= R^II_lowest[i] then
                Begin
                        sum += Latencies_importance_value[i]
                Else if input_property[i] <= R^II_lowest[i] then
                        sum += Latencies_importance_value[i] * lowvalue_importance
                end

        End
end

CalculateSubPatternDurations(input, lowvalue_importance, &Fsum, &sum)
(Se algoritmen för att beräkna latencies)
```

Om en inmatning verifieras i detta steg så går algoritmen vidare till steg 2, nämligen att verifiera stabila lutningar. I annat fall klassificeras inmatningen som ett inbrottsförsök och algoritmen avslutas

6.2 Verifiera stabila lutningar

Kapitel 5.3.2 påvisar en typ av mätbart mönster i inmatningarna; *stabila lutningar*. Detta kapitel beskriver hur Suddig C försöker utnyttja den typ av mönster. Först beskrivs

initieringen av den information i referenssignaturen som behövs för att verifiera de stabila lutningarna. Efter detta beskrivs själva verifieringen.

6.2.1 Initiering

För att underlätta förklaringen av hur stabila lutningar tas tillvara i Suddig C införs en ny definition *extrastabil lutning*. En extrastabil lutning är då den stabila lutningen förekommer i 100% av inmatningarna i T. Initieringen blir då att *märka ut* dessa extrastabila lutningar.

6.2.2 Verifiering

Själva verifieringen sker i två steg:

1. Om antalet stabila lutningar, som matchar de extrastabila lutningarna för latencies i T, delat med antalet extrastabila lutningar i T, överskrider ett tröskelvärde[1] så verifieras stabila lutningen för inmatningen. I annat fall nekas inmatningen.
2. Utför samma sak för de stabila lutningarna för durationer.

Pseudokod för verifieringen

```
VerifyStableSlopes(input) : returns TRUE iff verified
Begin
        Matches := GetSlopeMatchesLatencies(input)

        If (Matches / Extra_stable_slopes_count_latencies) < SlopeThreshold_latency then
                return FALSE

        Matches := GetSlopeMatchesDurations(input, duration)

        If (Matches / Extra_stable_slopes_count_durations) < SlopeThreshold_duration then
                return FALSE

        Return TRUE
end

// Av utrymmesskäl beskrivs endast GetSlopeMatchesLatencies. GetSlopeMatchesDurations fungerar
// på samma sätt som GetSlopeMatchesLatencies
GetSlopeMatchesLatencies(input) : returns highest number of matching slopes
begin
        Highest :=0

        For j:= 0 to |Extra_stable_slopes_latencies| do
        Begin
                Matches :=0
                For i:= 1 to |input_leatencies| do
                Begin
```

[1] Tröskelvärden är normalt ett tal mellan 0.0 och 1.0, vilket även är fallet här.

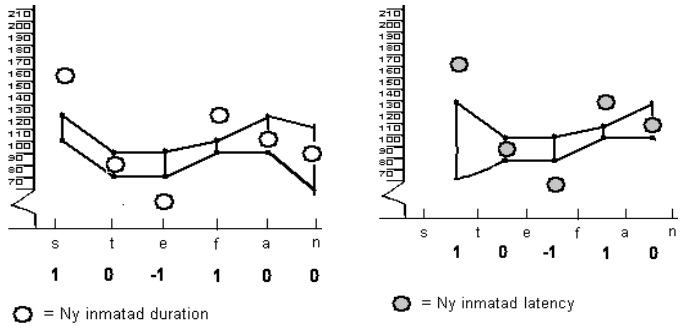
```
                    If input_latencies[i] = Extra_stable_slopes_latencies [j][i] then Matches++
                End

                If Matches > Highest then Highest := Matches
            End

            Return Highest
End
```

6.3 Verifiera brutet skrivsättsmönster

Om användaren bryter sitt vanliga skrivsättsmönster på ett karaktäristiskt sätt är det en fördel att veta att om skrivsättsmönstret bryts; så bryts det just på det karaktäristiska viset. Att sådana samband existerar påvisas i kapitel 5.3.4. Detta kapitel beskriver hur Suddig C försöker verifiera användaren med hjälp av denna information. Först beskrivs initieringen av dessa brutna skrivsättsmönster (det vill säga vilka mönster som finns och hur de ser ut), varefter själva verifieringen beskrivs.

6.3.1 Initiering

En rad i en matris M1 (för durationer) och M2 (för latencies) för varje inmatning i T skapas. Varje rad innehåller information över vilka gränser i R som bryts av inmatningarna i T. Informationen består av 1 då den övre gränsen bryts, 0 då ingen gräns bryts samt (-1) då den undre gränsen bryts.



Figur 15.- a) Durationer för inmatningen av lösenordet ”stefan”, b) Latencies för lösenordet ”stefan.

För inmatningen i figur 15 skapas således raden {1, 0, -1, 1, 0, 0 } i matrisen M1 och raden { 1, 0, -1, 1, 0} i matrisen M2.

6.3.2 Verifiering

För varje ny inmatning som skall verifieras skapas en vektor U1 för durationer och en vektor U2 för latencies på samma sätt som raderna i M1 respektive M2. Dessa vektorer

jämförs vid varje verifieringstillfälle med samtliga rader i M1 respektive M2. Själva verifieringen görs med följande algoritm.

```
// Duration_Threshold och Latency_Threshold är här tröskelvärden (mellan 0.0 och 1.0)
// för algoritmen
VerifyPattern()
Begin
        If VerifySubPattern(M1, U1, Duration_Threshold) and
        VerifySubPattern(M2, U2, Latency_Threshold) then
                return TRUE

        return FALSE
End

VerifySubPattern(M, U, PatternThreshold)
Begin
        Higest :=0

        For i:=1 to |T| do
        Begin
                lika :=0

                for j := 1 to |U| do
                begin
                        if U[j] = M[i][j] then lika++
                end

                if lika > Higest then Highest := lika
        end

        if Highest / |T| >= PatternThreshold then return TRUE

        return FALSE
end
```

7. Analys av algoritmens prestanda

Detta kapitel beskriver genomförandet och resultatet av det andra experiment (huvudexperimentet) vilket utgör delmål 3. Det vill säga att jämföra prestandan av den nyutvecklade algoritmen med två existerande.

7.1 Utförande av huvudexperimentet

Det andra experimentet genomfördes på så sätt att fem företag inom IT-branschen kontaktades om att låta sina anställda ställa upp på experimentet. Dessa företag hittades i telefonkatalogen över Skövde. Fyra företag lånade ut sina anställda för experimentet. Varje deltagare i experimentet fick svara på två frågor nämligen ålder och antalet timmar de använder datorer varje vecka. Dessutom noterades kön på varje deltagare. Varje deltagare informerades sedan om vad experimentet gick ut på i stora drag. Efter detta ombads de att skriva in ett på förhand valt lösenord 70 ggr i ett datorprogram utformat för att spara information om latencies och durationer för varje inmatat lösenord. Även felaktiga inmatningar sparades. 5 olika lösenord valdes ut.

1. jumbobear
2. parabol
3. zelda
4. vikdax
5. stagecoach

Just dessa lösenord valdes dels på grund av att de alla har olika längd (5, 6, 7, 9 och 10 bokstäver) och dels för att de tillsammans innehåller hela 21 bokstäver av alfabetets 28. På detta sätt går det analysera olika bokstävers inverkan på skrivsättsmönstret. Det går även att se hur olika längd på lösenordet påverkar prestanda för de olika algoritmerna. Lösenorden är också valda med tanke på att de ej vanligen används i det skrivna språket. Det gör att alla deltagare får liknande förutsättningar för just dessa lösenord. Inga övriga tangentbordsnedslag (såsom radslut) än de som tillhör lösenordet mäts i algoritmerna.

Datorprogrammet kördes på en och samma dator för att så långt som möjligt undvika skillnader i brus som kan bero på skillnader i datorernas prestanda. Varje användare använde dessutom samma tangentbord för sina inmatningar. Ingen av dem hade tidigare använt just det tangentbordet.

7.2 Jämförelse av prestanda

I detta kapitel beskrivs hur jämförelsen av de olika metodernas prestanda genomfördes, samt vilka resultat som kunde säkerställas.

7.2.1 Motivering till val av metoder till jämförelsen

Två metoder; en metod som använder ren statistik och en som använder euklidiskt distansmått valdes ut med tanke på att de tidigare utvärderats i ett flertal artiklar (Joyce & Gupta, 1990, Umphress & Williams, 1985). Metoderna är därmed väl beskrivna i tidigare artiklar och enkla att implementera. Utöver detta är dessa två metoder grundläggande och därmed lätta att förklara. Dessutom har metoderna tidigare visat lovande resultat för statisk verifiering trots sin enkelhet (Joyce & Gupta, 1990).

7.2.2 Sättet som jämförelsen genomfördes

De olika algoritmerna (statistisk, euklidiskt distansmått och den nya algoritmen) implementerades i ett datorprogram. Detta datorprogram skapades med tanke på en stor alternativrikedom vad gäller att justera algoritmernas parametrar och grafiskt representera datamängden.

Totalt utfördes experimentet på 21 användare i 4 olika företag, vilket gav totalt 21 * 70 inmatningar att analysera, varav 4 för samma lösenord (5 personer för sista lösenordet) vilka på så sätt kunde testas mot varandra. Detta gjordes på följande sätt.

1. De första 10 inmatningarna avfärdas (ses som en period för användarna att bekanta sig med lösenordet).
2. De 30 nästföljande inmatningarna används som referenssignatur
3. De sista 30 inmatningarna fungerar som egna inloggningsförsök
4. De övriga anställdas inmatningar för samma lösenord fungerar som inbrottsförsök

Detta genomfördes för samtliga användare i experimentet vilket skapade förutsättningar för att mäta FRR för totalt 630 inmatningar, samt att mäta FAR för totalt 4760 inmatningar.

7.2.3 Jämförelse av prestanda

För att sätta de olika tröskelvärdena i de olika metoderna implementerades en liten sökalgoritm i ett datorprogram som slumpade fram en stor mängd olika tröskelvärden för metoderna och valde de tröskelvärden som gav bäst resultat i fråga om (FAR+FRR).

Algoritmen premierade låg FAR framför låg FRR[1]. Den först testade användarens inmatningar (med tillhörande inbrottsförsök från de övriga användarna) från varje lösenordsgrupp valdes ut för att ”tränas” i algoritmen. De övriga användes för testning.

Jämförelse med en statistisk metod och euklidiskt distansmått

Nedan visas en tabell över resultaten från klassificeringstesten. De bästa värdena för FAR respektive FRR till varje lösenord är markerat med fet text.

Tabell 4. – Jämförelse av de tre implementerade metoderna i fråga om FAR och FRR i procent (avrundat till 1 decimal). Referenssignaturen är här skapad med 30 inmatningar av lösenordet.

Metod			**Suddig C**		**Statistisk**		**Euklidisk**	
Ord	**Egna inlogg**	**Inbrott**	**FAR**	**FRR**	**FAR**	**FRR**	**FAR**	**FRR**
jumbobear	30	210	8.8	**9.2**	**6.3**	26.7	7.2	33.3
Parabol	30	140	**10.8**	**11.1**	14.3	21.1	11.4	33.3
stagecoach	30	140	**4.3**	**11.1**	6.5	15.6	15.9	17.8
vikdax	30	140	**19.7**	17.8	29.0	**12.2**	49.3	21.1
Zelda	30	140	**4.7**	**21.1**	5.8	**21.1**	17.1	36.6
Medel			**9.6**	**13.7**	**12.0**	**19.8**	**19.4**	**28.5**

Som syns i tabellen ovan har Suddig C något högre prestanda med avseende på FAR och FRR än metoden implementerad med statistik och metoden implementerad med euklidiskt distansmått. Det går också att utläsa att Suddig C har högst prestanda, i fråga om FAR, för fyra av de fem lösenorden. Suddig C har även högst prestanda, i fråga om FRR, för fyra av de fem lösenorden.

Det som tydligt går att utläsa av tabell 4 är att euklidiskt distansmått fungerar betydligt sämre än de övriga två.

Övriga observationer

Utifrån tabell 4 tycks det vara så att ju längre lösenordet är desto bättre fungerar samtliga metoder. Felklassificeringarna tycks minska för varje bokstav i lösenordet. Detta bekräftar all tidigare forskning inom området och kan antagligen anses som självklart. Nedan visas en tabell över detta förhållande mellan prestanda och längd på lösenord.

[1] Sökalgoritmen som användes för att få fram tröskelvärden kan ha inverkat negativt på resultatet för framförallt Suddig C. Detta då den implementationen har flest tröskelvärden och sökalgoritmen på så vis har lättare för att hamna i ett lokalt optima. Detta beskrivs mer i kapitel 8.2.

Tabell 5. – Visar medelklassificeringsfel ((FAR+FRR) / 2) i procenttal avrundat till en decimal för samtliga metoder. Ordet ”zelda” tycks bryta mönstret för samtliga metoder, då det ordet har lågt medelfel trots att det endast har 5 bokstäver.

Ord\metod	**Suddig C**	**Statistisk**	**Euklidisk**
Zelda	12.9	13.4	26.8
Vikdax	18,8	20.6	35.2
parabol	10.9	17.7	22.3
jumbobear	9.0	16.5	20.2
stagecoach	7.9	11.0	16.8

Det tycks också vara så att vissa lösenord är olämpliga för keystroke dynamics. Ett sådant är ”vikdax” vilket, vid användning, tycks ge ett högt antal felklassificeringar för samtliga metoder.

Vissa ord tycks vara svårare att skriva in än andra. I detta experiment utmärkte sig orden ”jumbobear” och ”stagecoach”. Dessa är också de två längsta orden, vilket gör att antalet fel tycks ha ett starkt samband med längden på lösenordet. Ordet ”vikdax” har också ett högt felantal i jämförelse med de övriga vilket skulle kunna bero på att ordet är svårt att stava (x används sällan i det svenska språket etc.).

Tabell 6. - Antalet felaktiga inmatningar. Kolumnen ’uträkning’ visar summering av alla enskilda användares antal felaktiga inmatningar.

Ord	**Uträkning**	**Summa antal**	**Medelvärde**
jumbobear	10+19+17+16+8	70	14.0
parabol	2+2+1+15	20	5.0
Stagecoach	15+26+17+33	91	22.8
Vikdax	10+2+15+9	36	9.0
Zelda	4+2+9+10	25	6.3

7.3 Resultat kontra förväntat resultat

Det förväntade resultatet var att den nya algoritmen skulle ha en högre prestanda i fråga om FAR och FRR i förhållande till de två andra metoderna. Resultaten i rapporten pekar också på att så är fallet. Skillnaderna i prestanda är dock små mellan den nya metoden och den statistiska metod som implementerades. Resultaten för den statistiska metoden i denna rapport ligger nära tidigare arbetens resultat för statistiska metoder. Resultaten för metoden med euklidisk distansmått är dock betydligt sämre än i tidigare arbeten vilket är något förvånande. Som tidigare nämnts är dock inte resultaten i denna rapport direkt jämförbara med tidigare arbetens resultat. Detta exempelvis då lösenorden är betydligt kortare än i övriga arbeten (exempelvis Joyce & Gupta (1990)).

En ytterligare faktor som kan ha bidragit till låg prestanda för metoderna är att ingen av användarna hade någon vana med det tangentbord som användes. Dessutom valdes lösenord som användarna sällan eller aldrig tidigare skrivit, till skillnad från många tidigare arbeten vilka använt det egna namnet som lösenord. Detta bör dock inte påverka jämförelsen av de olika metodernas relativa prestanda i och med att alla implementerade metoder i detta arbete använde samma datamängd och samma tangentbord (det vill säga exakt samma förutsättningar).

8. Slutsatser

I detta arbete har en ny algoritm för att implementera keystroke dynamics introducerats. För att veta hur karaktären i skrivsättsmönstret hos användare kan tas tillvara utfördes ett initialt experiment. Utifrån analysen av experimentets resultat utvecklades den nya algoritmen så att den skulle kunna hantera de utpekade egenskaperna inom skrivsättsmönstren i testmaterialet. För att undersöka den nya algoritmens prestanda utfördes ytterligare ett experiment där användare fick skriva ett på förhand valt lösenord 70 gånger. Prestanda jämfördes därefter med två existerande metoder. Detta kapitel beskriver de slutsatser som kunde dras utifrån resultatet av huvudexperimentet. Kapitlet innehåller även övergripande reflektioner och möjliga vidare undersökningar inom området.

8.1 Prestanda för Suddig C

Målet med denna rapport är att skapa en ny algoritm och sedan jämföra den med två tidigare vanligt använda algoritmer. Om den nya algoritmen har högre prestanda än de två övriga algoritmerna kan slutsatsen dras att den nya algoritmen är lovande för implementering av keystroke dynamics. I annat fall kan den förkastas och visa att framtida ansträngning bör ligga på annat håll.

Resultaten av det andra experimentet (huvudexperimentet) visar att Suddig C har högre prestanda, med avseende på FAR och FRR, i jämförelse med euklidiskt distansmått och den statistiska metod som implementerades. Suddig C kan således ses som en lovande algoritm för keystroke dynamics. Av detta följer därmed att målet med rapporten är uppnått.

8.2 Reflektion

I detta kapitel görs en reflektion av arbetet som helhet. Även erfarenheter och detaljer från olika delar av processen diskuteras kortfattat.

Tidigare arbeten

Inom området keystroke dynamics finns en stor mängd artiklar. Dock finns det relativt få författare för dessa artiklar. Trots detta varierar notationerna (matematiska beteckningar etc.) mellan artiklarna. Detta upplevdes vid litteraturstudien som lite krångligt men var en viktig erfarenhet.

Det var lite olyckligt att tidigare arbetens testmaterial inte kunde användas i detta arbete. Då hade det gått att mer exakt bestämma prestanda för den nyutvecklade algoritmen. I de fall där författarna till vissa artiklar svarade att de kastat eller tappat bort testmaterialet bör det kunna ses som en brist för validiteten av deras resultat. Det kan dock vara förståeligt att testmaterial inte kan hållas tillgängligt för andra arbeten hur länge som helst.

Det initiala experimentet

Det misstänktes att det initiala experimentet utfördes med en alltför snäv användargrupp för att kunna ge en bra grund till att utveckla en ny algoritm. Det visade sig stämma till viss del. Urvalsgruppen i det initiala experimentet tycktes generellt ha en högre stabilitet i sitt skrivsättsmönster än urvalsgruppen i huvudexperimentet. Det påverkar dock inte resultatet från huvudexperimentet, men påverkade de val som gjordes i utvecklingen av den nya algoritmen. Även ifall urvalsgruppen i huvudexperimentet var vana datoranvändare noterades det under experimentet att de flesta tittade på tangentbordet medan de skrev. Detta var oväntat och kan ha påverkat resultatet negativt.

Utveckling av algoritm

Algoritmen utvecklades som tidigare nämnts med hjälp av det initiala experimentet. Det upplevdes som ganska enkelt att komma fram till logiken i algoritmen medan det upplevdes rätt svårt att beskriva hur den fungerade i rapporten. Ett stort antal alternativa beskrivningar prövades innan den nuvarande beskrivningen med pseudokod valdes. Bland annat undersöktes möjligheten att använda matematiska uttryck, vilket blev en generell och exakt beskrivning av funktionen. Det blev dock väldigt svårläsligt och tog lång tid att förstå. Därför valdes varianten med pseudokod och beskrivningar i text.

Jämförelse av prestanda

Den sökalgoritm som användes för att leta fram tröskelvärden till de tre implementerade metoderna kan ha påverkat resultatet för metoderna negativt. Framförallt då för Suddig C vilket är den metod med flest tröskelvärden av de tre. Ju fler parametrar (tröskelvärden) till algoritmen desto större risk att hamna i ett lokalt optima. I och med att förhållandet mellan FAR och FRR generellt är ömsesidigt uteslutande (låg FAR hög FRR etc.) tros dock att medelklassificeringsfelet ((FAR+FRR)/2) påverkas ytterst lite av sökalgoritmens valda tröskelvärden.

De skillnader i prestanda som kunde visas mellan den statistiska metoden och Suddig C påverkas antagligen en del av tillfälligheter i vilka inmatningar som valdes ut till träningsmängd och vilka som valdes som testmängd. Den största skillnaden tros dock bero på att Suddig C presterar mer jämnt mellan olika användare. Eftersom det inte vid en implementering av statisk verifiering direkt går att träna på obehöriga användare kan ökad stabilitet vara till stor fördel. Detta då det blir enklare att säga hur ett tröskelvärde påverkar inloggningen.

Val av lösenord påverkar prestanda

Hur *svårt* ett lösenord är att skriva tycks ha ett starkt samband med hur *bra* metoderna för keystroke dynamics fungerar för lösenordet. Även längd på lösenordet har en stor betydelse för hur väl de olika metoderna fungerar.

Användbarhet

Det är tydligt att ingen av de tre metoder som beskrivs i denna rapport är optimal som ensam säkerhetsmetod med de förutsättningar som fanns i dessa experiment. Både FAR och FRR är antagligen för höga för att någon av metoderna skall accepteras av användarna (i fråga om frustration vid False Rejection) och systemadministratören (i fråga om extra säkerhet vid inloggning). Det går dock inte att avfärda Suddig C som en lovande metod att bygga vidare på; eftersom den har högre prestanda än de övriga implementerade metoderna i denna rapport och stora möjligheter till förbättringar. Dessutom är det ingen tvekan om att keystroke dynamics förbättrar säkerheten för statisk verifiering (om viss FRR accepteras) då en stor mängd bedragare nekas tillgång till systemet av samtliga av de tre implementerade metoderna i denna rapport.

Framtiden för keystroke dynamics

Detta arbete har inte kunnat säkerställa att keystroke dynamics för statisk verifiering på något sätt överträffar befintliga biometriska system. Vissa biometriska säkerhetsmetoder, som bygger på fysiologiska attribut såsom fingeravtryck, har en betydligt högre säkerhet. Keystroke dynamics har dock, som tidigare sagts, stora fördelar gentemot övriga säkerhetsmetoder, men ytterligare forskning inom området krävs kontinuerligt för att höja säkerheten.

8.3 Framtida arbeten

Det kan vara intressant att titta vidare på hur det skulle kunna gå att kombinera flera olika algoritmer i en ny metod. Om en algoritm lyckas fånga upp en del typer av inloggningsförsök från ”hackers” och en annan algoritm andra typer så skulle algoritmerna kunna kombineras för att på så sätt uppnå en högre säkerhet (sänka FAR). Detta skulle kunna göras genom att implementera flera olika algoritmer och vid varje verifiering låta varje algoritm rösta om vad de tycker är likt referenssignaturen. Detta blir samma idé som *N-Version Programming/NVP-voting* använder för att åstadkomma felsäkerhet. Optimalt borde dessa olika algoritmer ha olika angreppssätt.

Det är egentligen omöjligt att på ett rättvist sätt jämföra de resultat som tas fram i denna rapport med resultat av tidigare resultat i andra arbeten. Detta i och med att inte samma datamängd används och att utvärderingsmetoderna och implementationerna skiljer sig från varandra. Det enda som egentligen kan göras är att diskutera på ett mindre exakt plan hur lika resultaten är utan att lägga någon större värdering i vilken metod som är att föredra. Det skulle vara väldigt intressant att implementera metoder som använder neurala nätverk och testa dessa under exakt samma förutsättningar med samma datamängd som användes i denna rapport.

I Suddig C finns stort utrymme för förbättringar vilka enkelt kan utföras. Varje steg i algoritmen kan vidareutvecklas och finslipas. R (initial-referensen) väljs nu ut som ett visst antal av de *första* inmatningarna i referenssignaturen. Ifall inmatningarna i R *slumpmässigt* väljs ut från referenssignaturen skulle möjligtvis ge bättre resultat. Även balans mellan storlek på R och T kan säkerligen undersökas vad som är optimalt.

Suddig C är nu specialanpassad för just keystroke dynamics. På grund av enkelheten i algoritmen blir det dock ganska lätt att göra algoritmen mer generell och på så vis använda den för andra tillämpningar.

Referenser

Abramowitz, M., Stegun, I.A., (1965) *Handbook of Mathematical Functions*. Dover Publications, Inc., New York.

AfB Association for Biometrics, 2003 Glossary of Biometric Terms, Tillgänglig på Internet: http://www.afb.org.uk/ [Hämtad 2004-02-11].

Alves-Foss, J., Barbossa, S., (1995) Accessing Computer Security Vulnerability, *ACM SIGOPS Operating System Review, 29,* No. 3.

Bergadano, F., , Gunetti, D., Picardi, C., (2002) User Authentication through Keystroke Dynamics, *ACM Transactions on Information and System Security, 5,* No. 4, 367–397.

Bleha, S.A., Hussien, B., McLaren, R., (1989) An application of fuzzy algorithms in a computer access security system, *Pattern Recognition Letters, 9*, 38-43.

Bleha, S.A., Knopp, J., Obaidat, M.S., (1992) Performance on the Perceptron Algorithm for the Classification of Computer Users, *Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's* .

Bleha, S.A., Slivinsky, C., Hussein, B., (1990) Computer-Access Security Systems Using Keystroke Dynamics, *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12,* No. 12, 1217-1222.

Brodley, C.E., Lane, T., (2000) Data Reduction Techniques for Instance-Based Learning from Human/Computer Interface Data, *Proceedings of the Seventeenth International Conference on Machine Learning.*

Cantú, F.J., Gutiérrez, F.J., Lerma-Rascón, M.M., Salgado-Garza, L.R. (2002) Biometrics and Data Mining: Comparison of Data Mining-Based Keystroke Dynamics Methods for Identity Verification, *Lecture Notes in Artificial Intelligence (MICAI-2002), Springer-Verlag 2313*, 460-469.

Chalmers, L., (1984) User Identification, Access Control and Audit Requirements, *Proceedings of the 1984 annual conference of the ACM on The fifth generation challenge.*

Downland, P.S., Furnell, S.M., Illingworth, H.M., Reynolds, P.L., (2000) Authentication and Supervision: A Survey of User Attitudes, *Computers & Security, 19,* 529-539.

Elkman, C., (1997) Naïve Bayesian Learning, Dept. Computer Science and Engineering Technical Report No. CS97-557.

Furnell, S.M., Ord, T., (2000) User authentication for keypad-based devices using keystroke analysis, *Proceedings of the Second International Network Conference (INC 2000), Plymouth, UK.*

Garcia, J., (1986) Personal Identification apparatus. Patent:4,62§,334., U.S. Patent and Trademark Office, Washington D.C.

Gifford, M.M., McCartney, D.J., Seal, C.H., (1999) Networked Biometrics Systems – requirements based on iris recognition, *BT Technology Journal, 17, No. 2,* 162-169.

Guven, A., Sogukpinar, I., (2003) Understanding Users' keystroke patterns for computer access security, *Computers & Security, 22, No. 8*, 695-706

Ibbetson, D., (1963) Algorithm 209: Gauss, *Communications of the ACM, 6*, No. 10, 616.
Ibbetson, D., (1967) Remarks on Algorithm 209: Gauss, *Communications of the ACM, 10*, No. 6, 377.
Ilonen, J., (2003) Keystroke dynamics, Lappeenranta University of Technology, Lappeenranta. Tillgänglig på Internet: http://www.it.lut.fi/kurssit/03-04/010970000/seminars/Ilonen.pdf [Hämtad 2004-02-01].
Joyce, R., Gupta, G., (1990) Identity Authentication Based on Keystroke Latencies, *Communications of the ACM, 33*, No. 2
Kim, H-J., (1995) Biometrics Is it A Viable Proposition for Identity Authentication and Access Control?, *Computers & Security*, *14*, 205-214
Korotkaya, Z., (2003), Biometric Authentication: Odor, Lappenranta University of Technology, Lappeenranta, Tillgänglig på Internet: http://www.it.lut.fi/kurssit/03-04/010970000/seminars/Korotkaya.pdf [Hämtad 2004-02-08].
Legget, J., Williams, G., (1988) Verifiying identity via keystroke characteristics, *International Journal of Man-Machine Studies, 28*, 67-76.
Leggett, J., Williams, G., Usnick, M., (1991) Dynamic identity verification via keystroke characteristics, *International Journal of Man-Machine Studies, 35*, 859-870.
Mahar, D, Napier, R., Wagner, M., Laverty, W., Henderson, R.D., Hiron, M., (1995) Optimizing digraph-latency based biometric typist verification systems, *International Journal of Human-Computer Studies,43*, No.4,579-592.
Matayas Jr, S.M., Stapleton, J., (2000) A Biometric Standard for Information Management and Security, *Computers & Security*, *19*, 428-44
Meadows, C., (1997) Paradigms in Computer Security, *Proceedings of the 1997 workshop on new security paradigms.*
Michalewicz, Z., Fogel, D., (2000) *How to Solve It: Modern Heuristics*, Springer Verlag, New York.
Monrose, F., Reiter, M.K. & Wetzel, S., (1999) *Password hardening based on keystroke dynamics*, Proceedings of the 6th ACM conference on Computer and communications security, 73-82.
Monrose, F., Rubin, A., (1997) Authentication Via Keystroke Dynamics, Proceedings of the 4th ACM conference on Computer and communications security.
Monrose, F., Rubin, A., (2000), Keystroke Dynamics as a Biometric for Authentication, *Future Generation Computer Systems, 16*, 351-359.
Neumann, P.G., (2000) Risks to the Public and Computers and Related Systems, *Software Engineering Notes, 25*, No. 4, 7-11
Norvig, P., Russel, S., (1995) *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey.
Obaidat, M.S., (1995) A Verification Methodology for Computer System Users, *Proceedings of the 1995 ACM symposium on Applied computing*
Perrig, A., Song, D., Venable, P., (1997) User Recognition by Keystroke Latency Pattern Analysis, Tillgänglig på Internet: http://gs207.sp.cs.cmu.edu/~adrian/file.ps [Hämtad 2004-01-10].
Rejman-Greene, M.,. (2001) Biometrics – real identities for a virtual world, *BT Technology Journal, 19, No 3*.

Rejman-Greene, M., (2002) Secure authentication using biometric methods, *Information Security Technical Report, 7, No.3*, 30-40.

Störr, H.P., (2002) A Compact Fuzzy Extension of Naïve Bayesian Classification Algorithm, *Proceedings InTech/VJFuzzy'2002*, 172-177

Umphress, D., Williams, G., (1985) Identity verification through keyboard characteristics, *International Journal of Man-Machine Studies, 23*, 263-273.

Valluro, R., (1995) *C++ Neural Networks and Fuzzy Logic*, IDG Books Worldwide Inc.

Young, J.R., Hammon, R.W., (1989) Method and apparatus for identifying an individual's identity., Patent: 4,805,222., U.S. Patent and Trademark Office, Washington, D.C.

Yu, E., Cho, S., (2003) Biometrics-based Password Identity Verification - Some Practical Issues and Solutions, Department of Industrial Engineering, Seoul National. Tillgänglig på Internet: http://dmlab.snu.ac.kr/IEA2003.pdf. [Hämtad 2004-01-20].

Yager, R., Zadeh, L., (1992) An Introduction to Fuzzy Logic Applications in Intelligent Systems, Kluwer Academic Publishers, Massaschussetts, USA.

Zadeh, L., Kacprzyk, J., (1992) Fuzzy Logic for the Management of Uncertainty, John Wiley & Sons, Inc., Canada.

Zadeh, L., (1994) Fuzzy Logic, Neural Networks and Soft Computing, *Communications of the ACM*, 37, No.3.

Zilberman, G., A., (2002) Security Method andApparatus Employing Authentication by Keystroke Dynamics, Patent: 6,442,692., U.S. Patent and Trademark Office, Washington, D.C.

Bilaga A – *Modeller för statistik*

Nedan beskrivs kortfattat vanligt använda matematiska modeller som används i denna rapport. Denna bilaga kan användas som en formelsamling för de som vill göra egna beräkningar för att förstå helheten. Bilagan kan också ses som kompletterande material då den innehåller en del information som utelämnades i bakgrundskapitlet av läsbarhetsskäl.

Standardavvikelse

Standardavvikelsen är ett mått på hur stora variationer det finns i en datamängd vilket används i många matematiska modeller för att beräkna sannolikheter. Standardavvikelsen betecknas σ och beräknas:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} x_i^{\,2} - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n}}$$

Där n är antalet element i datamängden och x_i är element i, inom datamängden

Normalfördelning

En slumpvariabel X är normalfördelad om den kan beskrivas med följande modell:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

För att beräkna sannolikheter för sådana variabler måste den *standardiserade normalfördelningen* användas. Detta innebär att standardavvikelsen är noll och väntevärdet ett vilket ger följande förenklade modell:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

Sannolikheten för att en standardiserad normalfördelad variabel Z är mindre än y betecknas *Prob(y)* vilket beräknas genom att integrera *f(Z)*. Det vill säga beräkna:

$$Prob(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{\left(-\frac{1}{2}y^2\right)}$$

Detta är i realiteten omöjligt men går lyckligtvis att approximera med hög precision. En enkel modell för detta är Abramowitz & Steguns formel (1965) vilket ger en precision på 4 decimaler.

$$Prob\ (y) = 1 - \frac{1}{2}\left(1 + \sum_{i=1}^{4} C_i y^i\right)$$

Där $C = \{0.196854, 0.115194, 0.000344, 0.019527\}$.

Ibbetson (1963) visar hur detta kan göras på ett sätt vilket ger en precision på 7 decimaler (Ibbetson, 1967). Även den formeln är enkelt uppbyggd men är lite för stor för att rymmas i denna bilaga. Den används dock för att implementera sannolikhetsberäkningar i ett av de datorprogram som utvecklades för denna rapport vilket gör att den kan vara bra att känna till.

Bilaga B – *Implementation av statistisk metod*

Denna bilaga beskriver mer formellt hur implementationen av den statistiska metod som användes i denna rapport utfördes. Algoritmen använder standard statistik så ingen närmare förklaring kommer att göras.

Pseudokod

```
VerifyStatistic(input)
Begin

        K := 0

        For i=1 to length(input) do
        begin
                If MV[i] + SD[i] * Threshold > input[i] then
                        If MV[i] – SD[i] * Threshold > input then
                                If 2-T-Test(input[i], Alpha_value) then K := K + 1
                        Endif
                Endif
        End

        If K > P-Threshold then return TRUE

        Return FALSE
End;
```

2-T-Test (x, alpha) :	Utför ett 2-Tailed-Test för att x skall tillhöra referenssignaturen med alfavärdet alpha
MV :	Medelvärdet för en specifik latency eller duration i referenssignaturen.
SD :	Standardavvikelsen för en specifik latency eller duration i referenssignaturen.
Threshold :	Ett tröskelvärde för skillnad mellan varje specifik latency/duration
P-Threshold :	Ett tröskelvärde för hur många gränser som kan missas
Alpha-Value:	Förutbestämt alfa-värde

Endast om detta test klaras för både latency och durationer i en inmatning så verifieras inmatningen. Det vill säga:

```
VerifyInput(latencies, durations)
Begin
        If VerifyStatistic(latencies) and verifyStatistic(durations) then return TRUE

        Return FALSE
End
```

Bilaga C – *Implementation av metod med euklidiskt distansmått*

Denna bilaga beskriver kortfattat hur metoden som använder euklidiskt distansmått implementerades.

Referenssignaturen skapades genom att varje inmatning i träningsmängden mäts med euklidiskt distansmått (latencies och durationer var för sig). Medelvärdet och standardavvikelsen av dessa värden räknas ut.

Vid verifiering uppmäts den inmatning som skall verifieras på samma sätt. Två tröskelvärden finns i modellen. Ett för latencies och ett för durationer. Tröskelvärdena används som T * standardavvikelsen i referenssignaturen. Om exempelvis en inmatning får ett distansmått för sina latencies under T * standardavvikelsen i referenssignaturen så är inmatningen verifierad för latencies, i annat fall icke verifierad för latencies. Dessa trösklar måste klaras för både durationer och latencies för att inmatningen skall verifieras.

Bilaga D – *Försökspersonernas fördelning*

Försökspersonernas företag Academic AB (IT-gymnasiet), Partner IT-Utbildarna, Tieto Enator eller WM-Data (representerat av A, B, C och D), kön (M/Q), ålder (heltal) och uppskattat antal timmar datoranvändning i veckan (heltal).

lösenord: zelda

1. A, M, 29, 40
2. A, M, 32, 45
3. A, M, 37, 30
4. A, Q, 29, 25

lösenord: vikdax

1. A, M, 31, 50
2. B, Q, 36, 20
3. B, Q, 55, 30
4. C, M, 34, 30

lösenord: stagecoach

1. C, M, 29, 45
2. C, M, 30, 40
3. C, M, 44, 30
4. C, Q, 31, 40

lösenord: parabol

1. C, M, 36, 35
2. C, M, 33, 40
3. C, M, 35, 30
4. C, M, 45, 20

lösenord: jumbobear

1. D, M, 40, 40
2. D, M, 25, 50
3. D, M, 37, 60
4. D, M, 23, 60
5. D, Q, 43, 30