NTNU
Norwegian University of
Science and Technology

Sikkerhet i mobilinfrastruktur/autentisering

Forfattere
Henriette Kolby Rohde Garder
Linn-Mari Kristiansen
Sturla Høgdahl Bae

Bachelor i informasjonssikkerhet
20 ECTS
Institutt for informasjonssikkerhet og kommunikasjonsteknologi
Norges teknisk-naturvitenskapelige universitet,

16.05.2018

Veileder Basel Katt

Sammendrag av Bacheloroppgaven

Tittel:	**Sikkerhet i mobilinfrastruktur/autentisering**
Dato:	16.05.2018
Deltakere:	Henriette Kolby Rohde Garder Linn-Mari Kristiansen Sturla Høgdahl Bae
Veiledere:	Basel Katt
Oppdragsgiver:	Eika Gruppen AS
Kontaktperson:	Jon Hagen, jhag@eika.no, 90133202
Nøkkelord:	Sikkerhet, mobilinfrastruktur, mobilautentisering, nøkkellager, biometrisk autentisering
Antall sider:	132
Antall vedlegg:	7
Tilgjengelighet:	Åpen

Sammendrag:

Mange nettsteder tilbyr den dag i dag brukere å autentisere seg med en engangskode på SMS i tillegg til å oppgi passord. Dette gjøres fordi man antar at brukeren må ha tilgang til mobiltelefonen sin for å motta engangskoden. Hvor sikker er denne infrastrukturen, og hva er den beste metoden for å autentisere en bruker med en mobiltelefon? Målet til dette prosjektet var å samle inn nok data for å gi gode svar til de gitte problemstillingene. En del av målet vårt var å lage en applikasjon som et konseptbevis for sterk kommunikasjon ved bruk av fingeravtrykk opp mot en server. I den første delen samlet vi inn og analyserte data fra forskjellige kilder for å kunne si hvor sikker den mobile infrastrukturen egentlig er. Vi konkluderte med at SMS ikke er egnet for autentisering fordi det eksisterer mange sårbarheter i infrastrukturen som gir muligheten til å overvåke telefonsamtaler og SMS-er. Det har blitt implementert tiltak for å hindre denne type overvåkning, men effekten av disse tiltakene er ukjent. I del to av prosjektet identifiserte vi hvordan nøkler kan bli lagret sikkert i mobiltelefoner, og dette ble brukt til å implementere konseptbeviset.

Summary of Graduate Project

Title:	**Security in mobile infrastructure/authentication**
Date:	16.05.2018
Authors:	Henriette Kolby Rohde Garder Linn-Mari Kristiansen Sturla Høgdahl Bae
Supervisor:	Basel Katt
Employer:	Eika Gruppen AS
Contact Person:	Jon Hagen, jhag@eika.no, 90133202
Keywords:	Security, telecommunications infrastructure, mobile authentication, security, Keystore, biometric authentication
Pages:	132
Attachments:	7
Availability:	Open

Abstract:

Many websites today offer users to authenticate with a one-time code sent via SMS in addition to providing a password. This is done because it is assumed that the user must have access to their mobile phone to receive the one-time code. How safe is this infrastructure, and what is the best method to authenticate a user on a mobile device? The goal of this project was to gather enough data to give good answers to the given problems. A part of our goal was to provide a proof of concept application for strong authentication with the use of a fingerprint against a server system. In the first part we gathered and analysed data from many different sources to determine how secure the mobile infrastructure really is. We concluded that SMS is not suitable for authentication because there exists many vulnerabilities in the mobile infrastructure that allows monitoring of phone calls and SMS. There has been implemented controls to prevent this kind of monitoring, but the effect of these controls are unknown. In the second part we identified how keys can be stored securely in mobile devices and used this to implement the proof of concept.

Forord

Eika Gruppen AS er en del av Eika Alliansen og leverer produkter og tjenester til lokalbanker i Norge [1]. Eika alliansen består av Eika-bankene, Eika Gruppen og Eika Boligkreditt. Til sammen har Eika alliansen over én million kunder og er dermed en av de største aktørene i det norske finansmarkedet [2].

Som en leverandør av produkter og tjenester til banker er det viktig å ha oversikt over hvor sikre ulike autentiseringsmetoder er, og Eika Gruppen ønsker derfor at vi skal undersøke sikkerheten til nettopp dette.

Vi vil gjerne takke Basel Katt for all veiledningen han har gitt oss gjennom semesteret. Vi vil også takke våre kontaktpersoner ved Eika Gruppen, Thomas Eriksson og Jon Hagen.

Innhold

Forord . . . . . **iii**
Innhold . . . . . **iv**
Figurer . . . . . **viii**
Tabeller . . . . . **ix**
Listings . . . . . **x**
1 Introduksjon **1**
1.1 Problemstilling med bakgrunn 1
1.1.1 Problemstilling 1
1.2 Formålet med prosjektet 1
1.2.1 Målgruppe 2
1.2.2 Egen bakgrunn og kompetanse 2
1.2.3 Avgrensing 2
1.2.4 Rammer for arbeidet 2
1.3 Metode 3
1.4 Rapportstruktur 3
2 Bakgrunn **5**
2.1 Sentrale begreper knyttet til sikkerhet i mobilinfrastruktur 5
2.1.1 SS7 5
2.1.2 Diameter 5
2.1.3 IMSI nummer 6
2.1.4 IMSI-fanger 6
2.1.5 Rainbow-tables 6
2.2 Sentrale begreper knyttet til sikkerhet i mobilautentisering 7
2.2.1 Miste privilegium 7
2.2.2 Sandboxing 7
2.2.3 Adgangskontroll 7
2.2.4 Diskkryptering 7
2.2.5 Biometri 7
2.2.6 Kryptografiske nøkler 7
2.2.7 HTTPS 8
2.2.8 Certificate pinning 8
2.2.9 Certificate Transparency 8
2.2.10 Nøkkellager 8
3 Analyse av sikkerheten i mobil infrastruktur **10**
3.1 Rapporter om sikkerhet i mobilinfrastruktur 10

3.1.1 Positive Technologies . . . 10
3.1.2 GSMA . . . 10
3.1.3 ENISA . . . 11
3.1.4 CSRIC V - Working Group 10 . . . 11
3.1.5 NKOM . . . 12
3.2 Potensielle angrep mot mobil infrastruktur . . . 12
3.2.1 Angrepsvektorer . . . 12
3.2.2 Mulige konsekvenser . . . 14
3.2.3 Konkrete angrep . . . 15
3.3 Tiltak . . . 16
3.3.1 Tiltak implementert av mobiloperatører . . . 16
3.3.2 Tiltak for forbrukere . . . 16
3.4 Sikkerhet i norsk mobilinfrastruktur . . . 17
3.4.1 Falske basestasjoner i Oslo sentrum . . . 17
3.4.2 Tiltak mot SS7-sårbarheter i norsk mobilinfrastruktur . . . 17
3.4.3 Kryptering i norske mobilnettverk . . . 18
3.4.4 Huawei . . . 18
3.5 Vurdering av kodebærere . . . 19
3.5.1 SMS som kodebærer . . . 19
3.5.2 Alternative kodebærere . . . 21
3.5.3 Sammenligning av kodebærere . . . 23
3.5.4 Anbefalt kodebærer . . . 24
4 Analyse av sikkerhet i mobil autentisering . . . 25
4.1 Fellestrekk for sikkerhet i Android og iOS . . . 25
4.1.1 Prinsippet om minste privilegium . . . 25
4.1.2 Sandboxing . . . 25
4.1.3 Tvungen adgangskontroll . . . 25
4.1.4 Sikker oppstart . . . 26
4.1.5 Diskkryptering . . . 26
4.1.6 Autentisering med biometri . . . 26
4.2 Sikkerhet spesielt for Android . . . 28
4.2.1 Påvirkning av produsent-tilpasning . . . 28
4.2.2 Diskkryptering på Android . . . 29
4.2.3 Oppdatering av enheter . . . 29
4.2.4 Installasjon av applikasjoner . . . 30
4.3 Sikkerhet spesielt for iOS . . . 32
4.3.1 Diskkryptering på iOS . . . 32
4.3.2 Oppdatering av enheter . . . 32
4.3.3 Installasjon av applikasjoner . . . 32
4.4 HTTPS . . . 33

4.4.1 Problemer med HTTPS . . . 33
4.4.2 Certificate pinning . . . 34
4.4.3 Certificate Transparency . . . 34
4.5 Lagring av kryptografiske nøkler . . . 35
4.5.1 Lagre nøkler i applikasjonsmappen . . . 35
4.5.2 Lagre nøkler i et nøkkellager . . . 35
4.5.3 Maskinvarestøttet nøkkellager . . . 36
4.5.4 Sammenligning av ulike metoder for å lagre kryptografiske nøkler . 39
5 Konseptbevis for biometrisk autentisering mot et serversystem . . . 41
5.1 Krav og Design . . . 41
5.1.1 Kravspesifikasjon . . . 41
5.1.2 Arkitektur . . . 44
5.1.3 Standarder for sterk autentisering . . . 44
5.1.4 Autentisering for iOS . . . 45
5.1.5 Autentiseringsmetode for autentiserte økter . . . 46
5.1.6 Sikkerhet ved autentiseringsmetoden . . . 52
5.1.7 Autentiseringsmetode for førstegangsautentisering . . . 55
5.1.8 Beskyttelse mot replay-angrep . . . 56
5.2 Implementasjon . . . 56
5.2.1 Utvalgte kodesnutter . . . 57
5.2.2 GUI . . . 59
5.3 Kvalitetssikring . . . 59
5.4 Dokumentasjon . . . 59
5.5 Kildekode . . . 60
5.6 Trusselmodellering . . . 60
5.6.1 Misbrukstilfeller . . . 60
5.6.2 Dataflytdiagram (DFD) . . . 60
5.6.3 Angrepstrær . . . 60
5.6.4 CVSS . . . 65
5.7 Trusler . . . 65
5.7.1 Microsoft SDL Threat modeling tool . . . 65
5.7.2 Identifiserte trusler . . . 66
5.7.3 Foreslåtte tiltak . . . 67
6 Konklusjon . . . 69
6.1 Diskusjon rundt resultater . . . 69
6.2 Kritikk av oppgaven . . . 69
6.3 Fremtidig arbeid . . . 70
6.4 Evaluering av gruppens arbeid . . . 70
6.5 Konklusjon . . . 70
Bibliografi . . . 72

A Ordforklaringer **88**
B Relevant kommunikasjon **91**
B.1 E-post fra Henning Lunde 91
C CVSS Scores **93**
C.1 CVSS Utregninger 93
C.2 Metode for utregning av CVSS score 94
D GANTT og Prosjektplan **95**
D.1 GANTT 95
D.2 Prosjektplan 98
E Prosjektavtale **114**
F Møtereferater **118**
G Arbeidslogg **123**

Figurer

1 Beskrivelse av transaksjoner fra banken kan gjøre autentiseringen sikrere. Dersom Alice er oppmerksom vil hun oppdage at både summen og mottakeren er feil, og dermed ikke fullføre transaksjonen. 20
2 Arkitekturen til et maskinvarestøttet nøkkellager Illustrasjon av Android Open Source Project, utgitt under CC BY 3.0 lisensen. Endret til vektorgrafikk. 38
3 Brukstilfeller 42
4 Illustrasjon av "distribuert funksjon" 44
5 FIDO UAF registrering [3]. Endret font og nedskalert fra original. 46
6 FIDO UAF autentisering [3]. Endret font og nedskalert fra original. 47
7 Et sekvensdiagram som illustrerer hvordan førstegangsautentiseringen fungerer 49
8 Et sekvensdiagram som illustrerer hvordan autentisere med fingeravtrykk og opprette en økt 50
9 Et sekvensdiagram som illustrerer hvordan autentisere en handling under en økt 51
10 Et sekvensdiagram som illustrerer hvordan man autentiserer seg med fingeravtrykk, uten å opprette en økt 53
11 Et sekvensdiagram som illustrerer hvordan autentisere en transaksjon med fingeravtrykk 54
12 Misbrukstilfeller 61
13 Dataflytdiagram 61
14 Et angrepstre som illustrerer hvordan en aktør kan angi seg for å være en bruker 63
15 Et angrepstre som illustrerer hvordan en aktør kan selge informasjon/tilgang til andre aktører 64
16 Et angrepstre som illustrerer hvordan en aktør kan gjennomføre tjenestenektangrep 64

Tabeller

1 Sammenligning av ulike kodebærere 24
2 Fordeling av versjon blant Android-enheter. Data samlet inn ved å se på versjonen til Android-enheter som besøkte Google Play Store fra den 10. til 16. april 2018. Utgitt av Android Open Source Project under CC BY 2.5 lisensen. 30
3 Sammenligning av ulike metoder for å lagre kryptografiske nøkler 40
4 Brukstilfelle for “førstegangsautentisering” 42
5 Brukstilfelle for “autentiser med fingeravtrykk mot server” 43
6 Brukstilfelle for “autentiser handling” 43
7 Identifiserte trusler med CVSS-verdi 66
8 Identifiserte trusler for førstegangsautentisering 67
9 Identifiserte operasjonelle trusler 67

Listings

5.1 Koding av nøkkel til PEM-format 57
5.2 Verifisering av signatur 57
5.3 Volley tar imot parametere med en hashmap 58
5.4 Nettverkssikkerhets-konfigurasjonsfil 59

1 Introduksjon

Introduksjonen beskriver først og fremst problemstillingen med bakgrunn og rammene rundt oppgaven.

1.1 Problemstilling med bakgrunn

Mobile enheter har blitt en naturlig del av hverdagen til de aller fleste nordmenn, og det har blitt godt integrert i samfunnet. De brukes til å kommunisere, betale regninger, kjøpe og oppbevare billetter, kart og mye mer. Mange nordmenn er flinke til å installere antivirusprogrammer og er forsiktige med hva de laster ned på datamaskinene sine, men færre tenker på sikkerheten til smarttelefonen sin [4]. Ettersom det har blitt vanlig å bruke mobiltelefon til alle de overnevnte funksjonene, blir man nødt til å stille seg et spørsmål hvorvidt sikkerheten i programvaren og maskinvaren er tilstrekkelig for å hindre andre i å få tak i sensitiv informasjon. Hvordan vet man at en applikasjon ikke kan få tak i passordet ditt når man logger inn et annet sted på mobiltelefonen?

To-faktor og to-trinns autentisering er en voksende trend. Stadig flere tjenester tar i bruk to-faktor autentisering [5], men likevel er det få som benytter det, og det er mange som ikke er klar over muligheten. Google har rapportert at de har 1,2 milliarder registrerte Gmail brukere, men mindre enn 10% av disse har tatt i bruk to-trinns autentisering [6].

Ved bruk av to faktorer i autentiseringen skal det mye mer til for at en potensiell angriper skal kunne ta kontroll over en annen brukers konto, og ettersom SMS er noe folk flest har tilgang til [7], er dette en metode som brukes på mange nettsider [5]. Man kan fort tenke at man er fullstendig beskyttet når man trenger en SMS på sin egen telefon for å logge inn, men er det virkelig bare senderen og mottakeren som kan lese innholdet i en SMS?

1.1.1 Problemstilling

På bakgrunn av utfordringene beskrevet i 1.1 har vi valgt følgende problemstilling:

«Hvor sikkert er det å bruke engangskoder på SMS til autentisering, og hvor sikker er moderne autentiseringsmetoder på mobile enheter?»

1.2 Formålet med prosjektet

En populær metode å implementere to-faktor autentisering på er ved hjelp av engangskode på SMS [5]. Vi har fått i oppdrag av Eika Gruppen å undersøke sikkerheten i mobil infrastruktur knyttet til bruken av disse engangskodene og sikkerheten i mobil autentisering, som innebærer analyse av programvaren og maskinvaren i mobile enheter. Alternativer til engangskode på SMS skal vurderes i forhold til dette, og angrep mot infrastrukturen skal beskrives for å gi best mulig oversikt over hvor sikker denne praksisen er og hvorfor.

Det skal utvikles en applikasjon med sterk autentisering knyttet opp mot et server-

system, som et konseptbevis for sikker mobilautentisering. Sikkerheten i denne applikasjonen skal analyseres ved hjelp av blant annet trusselmodellering for å identifisere eventuelle måter en angriper kan komme seg rundt autentiseringen.

1.2.1 Målgruppe

Målgruppen for denne rapporten er personer som er interessert i å lære om sikkerhet i mobilinfrastruktur og mobilautentisering. For å få fullt utbytte av denne rapporten, vil det være nyttig å ha grunnleggende IT-kompetanse.

Rapporten er spesielt relevant for alle som planlegger å implementere to-faktor autentisering eller en applikasjon som autentiserer brukere med fingeravtrykk/FaceID mot et serversystem.

1.2.2 Egen bakgrunn og kompetanse

Vi visste ikke så mye om sikkerhet i hverken mobilinfrastruktur eller mobilautentisering før vi begynte med prosjektet. En av oss hadde kompetanse innen Java-applikasjonsutvikling fra før av, men ingen av oss hadde utviklet mobilapplikasjoner.

Alle måtte derfor sette seg inn i sikkerhet knyttet til mobilinfrastruktur og mobilautentisering, og to personer lærte seg applikasjonsutvikling til Android. Fordi vi ikke hadde noen kompetanse rundt mobilinfrastruktur, måtte vi først sette oss inn i hvordan infrastrukturen sikres og hvilke sårbarheter som eksisterer, før vi kunne undersøke enkelte områder nærmere. Innen mobilautentisering hadde vi litt mer kunnskap fra før av, ettersom Android og iOS bygger på Linux/Unix, og benytter seg av velkjente sikkerhetspraksiser. Vi hadde allikevel mye vi måtte lære oss knyttet til sikre miljøer, nøkkellagre og hvordan man benytter seg av dette i en applikasjon.

1.2.3 Avgrensing

Siden oppgaven sier lite om hvor dypt vi skal undersøke sikkerheten, må vi avgrense problemet for at vi skal rekke å dekke hele oppgaven før innleveringsfristen til rapporten. I tillegg til rammene som er satt fra oppdragsgiver, har vi valgt å avgrense prosjektet på følgende måte: Ettersom sikkerhet i mobil arkitektur er et veldig vidt tema, skal vi kun undersøke sikkerheten i programvare og maskinvare som kan knyttes til autentisering. Dette vil for eksempel innebære sikkerhet knyttet til lagring av hemmelige nøkler, fordi man kan bruke lagrede nøkler til autentisering. Analysen av programvaresikkerheten går ikke like dypt for iOS som for Android, ettersom Android har åpen kildekode og iOS har lukket kildekode. Dersom det ikke er mulig å få tak i tilstrekkelig informasjon om sikkerheten tilknyttet en funksjon, vil vi heller ikke analysere den gjeldende sikkerheten, slik at vi unngår å måtte gjøre antakelser.

1.2.4 Rammer for arbeidet

Oppgaven er delt inn i 2 deler. Den første delen går ut på å undersøke sikkerheten i mobilinfrastrukturen generelt og med fokus på bruk av engangskoder på SMS til autentisering. Den andre delen går ut på å vurdere sikkerheten i moderne autentiseringsmuligheter på mobiltelefon og lignende enheter. I tillegg skulle vi utvikle en applikasjon som skal fungere som et konseptbevis. Ettersom sikkerhet i mobilinfrastruktur og mobilautentisering er veldig store temaer, har vi sammen med oppdragsgiver satt opp noen rammer for hver del av oppgaven.

Rammer for del 1

Første del av prosjektet går ut på å undersøke sikkerheten i mobil infrastruktur, med fokus på bruk av engangskoder på SMS som autentisering.

- Prosjektet skal undersøke sikkerheten basert på at:
  - Personer reiser til andre land (roaming)
  - Personer har ulike modeller fra ulike produsenter (Apple, Samsung, Sony etc.)
  - Noen har nye mobiltelefoner og noen har eldre mobiltelefoner
- Fokuset vil være på SMS, men prosjektet vil vurdere sikkerheten til andre to-faktor autentiserings metoder og sammenligne sikkerheten til disse metodene mot SMS

Rammer for del 2

Den andre delen av prosjektet går ut på autentiseringsmetoder på mobile enheter, og å vurdere sikkerheten knyttet til disse.

- Prosjektet trenger ikke å evaluere om sikkerhetsfunksjonene er korrekt implementert.
- Prosjektet skal undersøke hvordan standarder er implementert ulikt hos ulike produsenter, men skal kun ta for seg de mest vanlige produsentene.
- Applikasjonen skal utvikles til iOS eller Android.

1.3 Metode

Prosjektet baserer seg i stor grad på eksisterende forskning. Først og fremst lette vi etter bøker og vitenskapelige rapporter på NTNU sitt universitetsbibliotek og andre databaser med vitenskapelige artikler.

For å lære om sikkerhet i mobilinfrastrukturen oppsøkte vi rapporter fra en rekke myndigheter som arbeider med å sikre infrastrukturen. Disse rapportene refererte til andre rapporter fra uavhengige sikkerhetsselskaper, som vi også har lest gjennom.

Innen mobilautentisering hentet vi ut mest mulig informasjon/dokumentasjon direkte fra organisasjoner som Apple og Google/Android Open Source Project (førstehåndskilder), da disse ga en god oversikt over hvordan operativsystemene er bygd opp og hvilke sikkerhetsmekanismer som er implementert. For analyse av sikkerheten i disse systemene har vi lest gjennom vitenskapelige rapporter og sett på tidligere foredrag fra sikkerhetseksperter.

Vi har valgt disse kildene ettersom de både gir pålitelig og objektivt informasjon om systemene.

Der vi ikke fant noen rapporter eller dokumentasjon, har vi benyttet oss av velkjente nettsteder for å finne informasjon. I noen tilfeller har vi kunnet bekrefte informasjonen ved hjelp av egne erfaringer, og i andre tilfeller har vi sammenlignet informasjonen fra ulike kilder.

1.4 Rapportstruktur

Kapittel 2 beskriver teknologier og konsepter relatert til mobilinfrastruktur og mobil autentisering. Deretter kommer kapittel 3 som handler om sikkerhet i mobilinfrastruktur. Etter dette kommer kapitlene om sikkerhet i mobilautentisering. Kapittel 4 handler om

hvordan kryptografiske nøkler sikres og hvordan man kan sikre autentisering over internett, og kapittel 5 handler om konseptbeviset vi har utviklet og trusselmodelleringen vi har gjennomført for denne.

2 Bakgrunn

Før vi går inn i de forskjellige resultatene, er det viktig å forklare noen konsepter relatert til mobil infrastruktur og mobilsikkerhet.

2.1 Sentrale begreper knyttet til sikkerhet i mobilinfrastruktur

Store deler av den mobile infrastrukturen er basert på gamle elementer. For å få et helhetsbilde i hva som gjør infrastrukturen sårbar, starter vi med en gjennomgang av disse. Det er noen elementer i infrastrukturen som er veldig attraktive for en angriper, siden disse kan bli utnyttet for å oppnå målet lettere. Det finnes også en rekke verktøy som er laget spesielt for å utnytte disse elementene.

2.1.1 SS7

Signaling System No. 7, forkortet SS7, er en protokoll for å sette opp samtaler og sende tekst og data i telefonnettverk ved hjelp av ulike signaler. En mobiloperatør har gjerne et internt SS7-nettverk som kobler infrastrukturen til operatøren sammen. I tillegg er gjerne operatøren koblet til et globalt SS7-nettverk som kobler operatørens infrastruktur sammen med andre mobiloperatører. Grunnen til at mobiloperatørene er koblet til hverandre med et slikt nettverk, er at forbrukere skal kunne benytte seg av andre operatører sine telefonnettverk, for eksempel hvis man reiser til utlandet. Å bruke en annen operatørs telefonnettverk kalles “roaming” eller “gjesting”.

SS7 er en gammel protokoll fra 1975 [8]. På denne tiden var det kun noen få operatører som stolte på hverandre og disse koblet sammen mobilnettverkene, og sikkerhet var derfor ikke i fokus. Utstyr som kommuniserer med SS7 stoler normalt blindt på signalene det mottar. Dette har vist seg å bli et stort problem etter hvert som antallet mobiloperatører har økt. GSMA, den globale organisasjonen som forvalter interessene til mobiloperatører, skriver at de forener nesten 800 operatører i 220 land på nettsiden sin [9]. Det er ingen offentlige tall tilgjengelig på hvor mange operatører som er koblet sammen, men med tanke på at Telenor oppgir priser for å ringe til 204 ulike land, i tillegg til å skrive at det er mulig at de har avtaler med land som ikke er på listen [10], er det sannsynligvis mange hundre operatører som er koblet sammen.

I et foredrag fra 2014 fortalte den tyske sikkerhetsforskeren Karsten Nohl at det var omtrent 800 operatører som var koblet til det globale nettverket, men at mange av disse gjerne hadde andre (både lovlige og ulovlige) aktører som benytter seg av tilkoblingen til operatørene. Han påpekte også at internett er en gammel protokoll som ble bygget uten sikkerhet i tankene, på lik linje med SS7, men man har i senere tid klart å sikre internett godt ved hjelp av kryptering, autentisering og filtrering [11].

2.1.2 Diameter

Diameter er en autorisering, autentisering og regnskapsprotokoll [12]. Diameter benyttes i stedet for SS7 i 4G nettverk. Den baserer seg på IP og kan derfor være enklere for angripere å utnytte, ettersom dette er protokoller som er vanligere å lære om. Det har

vært lite fokus rundt sikkerheten knyttet til Diameter, men at det ikke har blitt dokumentert noen angrep som benytter seg av denne protokollen. Det har allikevel blitt påvist av sikkerhetsforskere at Diameter er sårbar [13].

2.1.3 IMSI nummer

IMSI (International mobile subscriber identity) nummeret er et unikt nummer som blir brukt for å identifisere et SIM-kort. Det består av opptil 15 siffer, og i Norge starter det med landskoden(47) [14]. Problemet som kan oppstå hvis IMSI-nummeret til en person lekkes, er at dette nummeret kan brukes til å spore opp eieren ved hjelp av sårbarheter i SS7 protokollen. Operatører kan beskytte kundene sine ved å ikke godkjenne tilgang til rutingsinformasjonen fra utsiden, men noen av sårbarhetene kan føre til lekkasje av et IMSI-nummer, som ikke operatøren kan gjøre mye for å hindre.

For å unngå at nummeret skal sendes i åpen form, sendes det vanligvis som et midlertidig nummer (TMSI), med unntak av når mobiltelefonen starter opp og må autentisere seg ovenfor nettverket. Det er derfor ikke mulig å spore personer ved å kun sniffe mobiltrafikken til mobile enheter i nærheten.

2.1.4 IMSI-fanger

En IMSI-fanger er en falsk basestasjon som kan brukes for å plukke opp stedsdata og mobiltrafikk [15]. Dette kan føre til store brudd på personvern, som har ført til at bruk av en IMSI-fanger er ulovlig i Norge i dag, med unntak fra bruk av myndighetene ved spesielle kriminalsaker [16].

Det finnes flere typer falske basestasjoner, men alle typene kalles gjerne IMSI-fanger, selv om de gjør andre ting enn å kun plukke opp IMSI-nummeret. IMSI-fangere kan bli brukt i man-in-the-middle angrep, hvor de er plassert mellom mobiltelefonen og basestasjonen for å plukke opp mobiltrafikk. Trafikken er normalt kryptert, men basestasjonen kan i mange tilfeller få den mobile enheten til å bytte om til 2G-nettverket der basestasjonen ikke trenger å autentisere seg, og deretter nedgradere eller deaktivere krypteringen [17]. Det finnes også billige IMSI-fangere for 3G og 4G nettverk, men disse kan kun overvåke bruksmønstre og posisjon [18].

Passive IMSI-fangere som kan dekryptere 3G trafikk er det mulig å lage, selv om det ikke ser ut til at dette markedsføres noe sted. Dersom man fanger opp mobiltrafikken til samtaler ved hjelp av en antenne som opererer på riktig frekvens, vil man kunne benytte seg av SS7-protokollen for å be om krypteringsnøkkelen og deretter dekryptere samtaler, SMS-er og datatrafikk. Ettersom dette forutsetter at man har tilgang til SS7 systemet og man allerede kan benytte SS7-signaler direkte til avlytting og overvåkning, er det ikke så overraskende at dette er en lite attraktiv løsning [11].

2.1.5 Rainbow-tables

Rainbow-tables, eller “regnbuetabeller”, er ferdigberegnede datasett som kan benyttes til å knekke nøkler raskere enn ved tradisjonelle “brute-force” angrep. Ved et brute-force angrep vil man prøve seg frem med alle mulige kombinasjoner, men dette kan være svært tidkrevende og upraktisk dersom man vil dekryptere data raskt. Rainbow-tables tar lang tid å generere, men når man benytter seg av dem til å dekryptere data vil dekrypteringen gå svært mye raskere enn ved et brute-force angrep. Rainbow-tables brukes normalt for å reversere hash-funksjoner, men kan også benyttes til å identifisere krypteringsnøkler

dersom man vet deler av klarteksten [19].

2.2 Sentrale begreper knyttet til sikkerhet i mobilautentisering

Også innen mobilautentisering er det en del begreper man må vite hva er for å kunne forstå hvordan mobilautentisering sikres.

2.2.1 Miste privilegium

Prinsippet om minste privilegium går ut på at ingen skal ha flere rettigheter, eller tilgang, enn de absolutt trenger. Når en person eller et program kun har de nødvendige rettighetene, vil konsekvensene av misbruk være mindre fordi man ikke har rettighet til å ødelegge eller misbruke andre ressurser enn de som er absolutt nødvendig for normalt bruk.

2.2.2 Sandboxing

Sandboxing er en teknikk brukt for å separere kjørende programmer. Dette er en sikkerhetsmekanisme som er laget for å hindre at sårbarheter i en prosess påvirker andre deler av systemet. Dette blir også brukt til å kontrollere hvilke ressurser et program har tilgang til, og vil i de fleste tilfeller følge prinsippet om minste privilegium [20].

2.2.3 Adgangskontroll

Adgangskontroll tilknyttet programvaren som kjører på en mobil enhet er på mange måter lik adgangskontroll i den fysiske verden også. Adgangskontroll benyttes til å begrense hvem som har tilgang til hva, for eksempel hvilke brukere som har lov til å lese fra hvilke filer. Ved bruk av obligatorisk adgangskontroll vil man ikke ha mulighet til å overstyre de obligatoriske reglene som er bygget inn i systemet.

2.2.4 Diskkryptering

Diskkryptering er en teknikk brukt for å øke sikkerheten til brukerens data. All data som ligger på brukerpartisjonen av disken blir kryptert med en nøkkel, og for at brukeren skal få lese og bruke data, må han eller hun autentisere seg med en pin-kode eller et passord som brukes til å dekryptere data [21].

2.2.5 Biometri

Biometri er i dette tilfellet bruk av biometri for autentisering. Denne metoden for autentisering baserer seg på noe man er, som for eksempel bruk av fingeravtrykk, ansiktsgjenkjenning eller iris-skanning. Systemene som tar i bruk dette lagrer for eksempel fingeravtrykket til en person, og hvis personen som skal autentisere seg har samme fingeravtrykk (fingeravtrykkene "matcher"), godkjennes autentiseringen [22].

2.2.6 Kryptografiske nøkler

Kryptografiske nøkler er data som bestemmer resultatet fra en kryptografisk operasjon. Det er to forskjellige typer nøkler, symmetriske og asymmetriske. Symmetrisk kryptografi bruker samme nøkkel til både kryptering og dekryptering, mens asymmetrisk kryptografi har både en offentlig nøkkel og en privat nøkkel.

Kryptografi med offentlig nøkkel blir ofte brukt til både kryptering og autentisering. Det er bare eieren som skal ha den private nøkkelen, så hvis man krypterer noe med

eierens offentlige nøkkel, er det bare eieren som skal ha mulighet til å dekryptere meldingen. Dersom man krypterer noe med den private nøkkelen, vil det kunne dekrypteres med den tilhørende offentlige nøkkelen. Dette brukes som oftest til autentisering og kalles gjerne "signering" av data [23].

2.2.7 HTTPS

HTTPS er en kommunikasjonsprotokoll som er brukt for sikker kommunikasjon over internett. Denne protokollen sørger autentisering, at data som skal sendes over nettverket er kryptert, og at integriteten til dataen er beskyttet [24].

2.2.8 Certificate pinning

Certificate pinning er en metode for å låse et domene til en hash av et sertifikat. Når en klient har låst et domene til en sertifikat-hash, vil klienten kreve at et av sertifikatene i sertifikat-kjeden til serveren har den gitte hashen. Dersom ingen av sertifikatene i sertifikat-kjeden har den angitte hashen, vil klienten nekte å koble til serveren.

På websider blir en pin gjerne lastet ned hver gang man besøker en nettside som ber nettleseren om å benytte seg av certificate pinning. Nettleseren får vite hvilke hash-er som skal være tillatt og en utløpsdato for låsen. Etter denne utløpsdatoen vil nettleseren begynne å akseptere andre sertifikater på nytt.

2.2.9 Certificate Transparency

Certificate Transparency (CT) er en annen metode man kan benytte seg av for å sikre HTTPS. CT kan beskrives som en form for kontinuerlig, offentlig revisjon av utstedte sertifikater og baserer seg på tre komponenter. Disse er "certificate logs", "monitors" og "auditors".

Certificate logs er offentlige logger der sertifikater legges til, men ikke kan fjernes. Når en CA utsteder et sertifikat legger også CA-en sertifikatet inn i en sertifikat-logg. Monitors er offentlige servere som overvåker sertifikatloggene for å oppdage mistenkelig aktivitet. Auditors er klienter som kontakter monitors for å bekrefte at et sertifikat ikke er falskt. Dersom et sertifikat ikke befinner seg i en sertifikatlogg, eller en monitor rapporterer om at sertifikatet er utstedt uten autorisasjon av domene-eieren, vil klienten som har aktivert certificate transparency nekte å koble til serveren.

Fordi alle sertifikater må befinne seg i en offentlig sertifikat-logg som kan leses av alle, blir det veldig enkelt å oppdage at uautoriserte sertifikater har blitt utstedt, og dermed oppdager man det raskt dersom en CA kompromitteres [25].

Før man aktiverer verifikasjon av CT på en webside eller applikasjon, må man sjekke om CA-en som utsteder sertifikatet til domenet legger sertifikatet til i en offentlig sertifikat-logg.

2.2.10 Nøkkellager

Et nøkkellager er et oppbevaringssted for nøklene man genererer, og skal helst ligge i et trygt sted i operativsystemet. Nøkkellageret er designet for at nøklene skal være sikre, og en av de viktigste måtene nøkkellageret hindrer hemmelige nøkler fra å komme på avveie, er at nøkkellageret er implementert slik at det ikke skal være mulig å hente nøkkelen ut av nøkkellageret. Applikasjonen som eier nøkkelen kan be nøkkellageret om å gjennomføre en kryptografisk operasjon med nøkkelen, men kan ikke selv hente ut

nøkkelen for å deretter gjennomføre den kryptografiske operasjonen. Dersom en angriper tar kontroll over en applikasjon vil angriperen altså kunne benytte seg av de hemmelige nøklene til å gjennomføre handlinger på enheten, men vil ikke kunne hente de ut for å benytte nøklene fra en annen enhet.

Beskyttelsen mot uthenting av nøkler gjør at nøkkellageret ikke kan benyttes til å lagre nøkler som er ment til å hentes ut [26], men nøkkellageret kan allikevel være nyttig dersom man vil lagre en nøkkel som skal kunne hentes ut. Disse nøklene kan beskyttes ved å kryptere og dekryptere de med en nøkkel som lagres i det sikre nøkkellageret.

3 Analyse av sikkerheten i mobil infrastruktur

Sikkerhet i mobilinfrastruktur går ut på å sikre konfidensialitet, integritet og tilgjengelighet til kommunikasjon over mobilnettverket. Mobilinfrastrukturen vår er global og knyttet sammen med mobilnettverk fra hundrevis av andre operatører ved hjelp av signalsystemer som SS7, og kan derfor være vanskelige å sikre.

3.1 Rapporter om sikkerhet i mobilinfrastruktur

Det har blitt forsket og undersøkt mye rundt dette temaet de siste årene, og i denne seksjonen presenterer vi de viktigste funnene. Mange av disse rapportene baserer seg på tester som ble gjennomført for to til tre år siden, og det er derfor mulig at sikkerheten har bedret seg vesentlig siden den gang.

3.1.1 Positive Technologies

I rapporten fra Positive Technologies[1], utgitt i 2016, kom det frem at ingen av SS7 nettverkene er sikre. 3/4 av SS7 nettverkene tilhører APAC (Asia, Stillehavet, Amerika and Karibia) mens den siste fjerdedelen tilfører EMEA (Europa, Russland, Midtøsten og Afrika). Positive Technologies kom frem til at EMEA er mindre sikkert enn APAC og at mindre operatører hadde mindre sikkerhet implementert, enn de større operatørene. Dette vil ikke si at de større operatørene er sikre, noe de ikke er. Positive Technologies utførte flere tester for å teste SS7 nettverkene, og kom frem til følgende resultater.

- 80% av angrepene rettet mot lekkasje av sensitiv data var vellykket.
- 67% av angrepene rettet mot svindel var vellykket.
- 77% av angrepene rettet mot det operasjonelle var vellykket.

Hovedproblemene var knyttet til konfigurasjonsfeil(38%) og hovedsakelig problemer med protokoller og systemets infrastruktur(61%). Sårbarhetene i SS7 gir mulighet for en angriper å blant annet lekke data, overvåke posisjonen din og fange opp SMS-er. Kun 1% av de vellykkede angrepene var på grunn av programvarefeil. Vi antar at dette er knyttet til programvarefeilen ved Telenor sitt utstyr som førte til at mobilnettverket deres var utilgjengelig i nesten fire timer i februar 2016 [27, 28].

3.1.2 GSMA

En av medlemmene i GSMA, Evolved Intelligence, annonserte i 2016 at de samarbeidet med en europeisk mobiloperatørgruppe for å sikre nettverket med en SS7 signaliseringsbrannmur [29]. Dette senker sannsynligheten for å kunne utnytte SS7-sårbarheter basert på signaler betydelig blant de operatørene som benytter seg av denne løsningen. PT SS7 Attack Discovery er ett av hovedproduktene som Positive Technologies tilbyr for å beskytte nettverket. De mener at dette er nødvendig tiltak siden de nyeste mobilnettverkene (4G og 5G) fortsatt inneholder mange av SS7-sårbarhetene [30]. GSMA har også laget

[1] Rapporten til Positive Technologies https://www.ptsecurity.com/upload/ptcom/SS7-VULNERABILITY-2016-eng.pdf

flere retningslinjer for mobiloperatører for å sikre seg bedre mot SS7 angrep, men disse er i dag bare tilgjengelige for GSMA medlemmer [31].

3.1.3 ENISA

EU sitt organ for nettverk- og informasjonssikkerhet, ENISA, publiserte i mars 2018 en rapport som tar for seg problemstillingene rundt sikkerhet i SS7, Diameter og 5G[2].

Rapporten er nokså ny og oppdatert, og oppsummerer mange av funnene fra andre rapporter vi har kommet over. Dersom man er interessert i å lære mer om sikkerheten i mobilinfrastrukturen, er det denne rapporten vi helst vil anbefale å lese.

En av ENISAs bekymringer er problemene rundt Diameter protokollen. Rapporten kommer frem til at det kan være flere grunner til at det ikke har vært gjennomført noen angrep via Diameter enda. En mulighet er at SS7 fortsatt gir tilfredsstillende resultater, og en annen mulighet er at de kriminelle enda ikke har hatt tid til å sette seg inn i hvordan de skal utnytte Diameter.

Rapporten påpeker også hvordan en SS7/Diameter-brannmur ikke vil være perfekt, og at falske positiver lett kan oppstå dersom en annen operatør har feil konfigurasjon på utstyret eller nettverket sitt. Dersom en operatør blokkerer legitim trafikk, kan operatøren muligens bli holdt ansvarlig for konsekvensene både økonomisk og juridisk. Gjennom en undersøkelse blant europeiske mobiloperatører, fant de ut at de fleste mobiloperatører har implementert grunnleggende beskyttelse i nettverkene sine, og det er kompleksitet og kostander som hindrer de fleste i å implementere mer avansert beskyttelse. Mange operatører har også veldig gammelt utstyr som er vanskelig eller umulig å implementere beskyttelse for. I undersøkelsen rapporterer 33% av operatørene at juridiske begrensninger hindrer bedre signaleringsbeskyttelse [32]. Dette har også vært diskutert i Norge, der Nkom forholder seg til ekomloven (lov om elektronisk kommunikasjon) og datatilsynet forholder seg til personopplysningsloven. Mye tyder på at norske operatører lagrer store mengder data om kundene sine for å kunne analysere dette, finne mistenkelig aktivitet og bruke resultatene for å beskytte seg mot angrep via SS7 [33].

3.1.4 CSRIC V - Working Group 10

Amerikanske Communication Security, Reliability, Interoperability Council (CSRIC), en rådgivningskomite for FCC, opprettet i 2016 en arbeidsgruppe som skulle undersøke hvordan man kunne redusere risikoen tilknyttet bruk av gamle systemer i mobilinfrastrukturen. I rapporten som ble gitt ut i mars 2017, konkluderer arbeidsgruppen med at den mest effektive måten å hindre et angrep mot SS7 på, er en SS7 brannmur. De legger vekt på at risikoene kommer til å øke når man bruker gamle nettverk som SS7 over IP ved hjelp av SIGTRAN, fordi det er et mye større trusselbilde i IP-nettverket som SS7 ikke er bygd for å takle. De anbefaler på det sterkeste at mobiloperatører tar i bruk sikkerhetsveiledningene utarbeidet av GSMA og 3GPP for å takle disse utfordringene. Noen av punktene CSRIC vil fokusere mer på fremover, er sikkerheten rundt 5G, Diameter og ikke-GSMA signaliseringsnettverk [34].

[2]Signalling Security in Telecom SS7/Diameter/5G: https://www.enisa.europa.eu/publications/signalling-security-in-telecom-ss7-diameter-5g

3.1.5 NKOM

I NKOMs risiko- og sårbarhetsanalyse fra 2017 kom det frem at den norske mobile infrastrukturen er ganske godt sikret, og det er et tett samarbeid blant nordiske bedrifter for å rette sårbarheter i nettverkene. Det er likevel veldig vanskelig å finne en løsning som sikrer SS7 og Diameter helt.

Truslene med høyest risiko i Norge er programvarefeil og oppgradering- eller vedlikeholdsfeil, siden disse kan slå ut mobiltjenesten for mange kunder samtidig. Små deler av den norske mobilinfrastrukturen er ofte nede som følge av strøm- eller fiberbrudd, spesielt i Nord-Norge der været ofte er mer ekstremt og det tar lang tid å reparere brudd fordi avstandene er store. Disse bruddene pleier kun å ramme en liten del av befolkningen, og har derfor mye mindre konsekvens enn for eksempel programvarefeil [35].

Selv om det finnes mange rapporter om hvor sårbart SS7 er, vil nok dette bli brukt i mange år fremover. Telenor skriver at de kommer til å beholde SS7-baserte systemer i lang tid fordi de må ta hensyn til internasjonale operatører. Dersom man skulle slutte å benytte seg av SS7 vil ikke forbrukere kunne benytte seg av mobile tjenester i utlandet [36].

3.2 Potensielle angrep mot mobil infrastruktur

Som mange andre kommunikasjonsnettverk har den mobile infrastrukturen en rekke sårbarheter som kan utnyttes, og det er et populært mål for angripere. Dette nettverket har store mengder av konfidensiell data som blir fraktet mellom forskjellige punkter, som for eksempel en telefonsamtale eller en SMS. Sikkerhetsloven bestemmer at det er forbudt å overføre sikkerhetsgradert informasjon over mobilnettet [37], men passord, engangskoder for to-trinns verifisering og bedriftshemmeligheter kan forekomme og vil være attraktive mål.

3.2.1 Angrepsvektorer

Hvis målet til en aktør er å angripe infrastrukturen, vil det første steget som oftest være å få tilgang til nettverket som infrastrukturen bruker som kommunikasjonsbærer. I dette tilfellet vil SS7-protokollen være et veldig attraktivt mål. Dette er et sett av gamle protokoller brukt for telefoni fra 1975, og var ikke designet med sikkerhet i fokus. Det har vært mange sikkerhetsrelaterte hendelser gjennom årene, som har ført til innføringen av flere tiltak for å sikre denne.

Kjøpe tilgang til SS7 fra kriminelle aktører

Noe som ser ut til å være en vanlig metode for å kjøpe seg tilgang til SS7, er ved hjelp av tjenester på dypnettet. Det eksisterer høyst sannsynlig aktører som har tilgang til SS7-nettverket og som videreselger denne tilgangen til andre. For å gjøre tjenestene så attraktive som mulig, har det blitt opprettet websider som gir et grensesnitt mot SS7-nettverket. Alt man trenger er altså å koble seg opp til Tor-nettverket, som er et nettverk laget for anonym nettsurfing [38], og betale for å få tilgang.

I tillegg til tjenester som gir direkte tilgang til SS7-nettverket, er det noen tjenester man kan kjøpe som er mer spesifikke. Disse tjenestene krever ofte ingen kunnskap om SS7-nettverket eller hvordan man utnytter sårbarheter i dette, fordi tjenesten tar seg av den jobben for deg – som for eksempel å overvåke posisjonen eller alle innkommende

SMS-er for et gitt mobilnummer.

Tjenesten som har fått mest medieomtale er "Interconnector", som skal ha gitt full tilgang til SS7-nettverket for 500 dollar i måneden. Denne tjenesten har blitt omtalt som en svindel fordi mange har rapportert at de ikke fikk tilgang etter å ha betalt for dette [39, 40]. Det er allikevel høyst sannsynlig at det eksisterer lignende tjenester på dypnettet. Det har vært rapportert gjentatte ganger at hackere kan kjøpe seg tilgang til SS7-nettverket. I en video fra Positive Technologies, der de demonstrerer hvordan man kan benytte seg av SS7-nettverket for å få tak i engangskoder sendt på SMS og deretter bruke disse for å få tilgang til andres brukerkontoer, er det tydelig at de benytter seg av en nettside som gjør det mulig å utnytte sårbarheter i SS7 [41].

Da eieren av "Interconnector" ble spurt om hvordan hun klarte å beholde tilgangen til SS7-nettverket, ville hun ikke svare på spørsmålet. Hun fortalte at hun ikke hadde møtt noen større utfordringer selv om operatører jevnlig blandet seg inn [39].

Kjøpe seg inn i SS7-nettverket som en falsk mobiloperatør

Å kjøpe seg inn som en mobiloperatør er tidkrevende og kostbart, og kan ikke gjennomføres anonymt i Norge [42]. Å bli en godkjent virtuell mobiloperatør som benytter seg av infrastrukturen til en dedikert operatør, krever godkjenning og sertifisering som kan ta mange måneder, og sannsynligheten for misbruk av nettverket blir oppdaget er høy [43]. Dersom det eksisterer falske aktører med tilgang til SS7, holder de sannsynligvis til i mindre utviklede land der slike trusler ikke håndheves like strengt. Det har vært ett tilfelle der det ble rapportert om at kriminelle benyttet seg av en falsk mobiloperatør, men det forekommer ikke noen detaljer rundt den falske mobiloperatøren [44]. Den originale kilden(en tysk avis) benytter ikke "falsk aktør" som beskrivelse, men heller "tvilsom aktør". Dette kan bety at det er snakk om en mobiloperatør som normalt tilbyr mobile tjenester til forbrukere, men som også tilbyr tjenester til kriminelle for ekstra inntekt [45].

Utpresning og utnyttelse av misfornøyde ansatte

Kaspersky rapporterte i august 2016 at kriminelle benyttet seg av ansatte i mobiloperatørselskaper for å få tilgang til mobilinfrastruktur. Ved å rekruttere misfornøyde ansatte eller benytte seg av utpresning av ansatte, kan de kriminelle få tilgang til mobile nettverk, som for eksempel SS7 eller Diameter [46].

Statlige Aktører

I mange land har statlige aktører mulighet til å bestemme over en eller flere mobiloperatører. I noen land eier staten mobiloperatøren, og i mange andre land vil staten trolig kunne presse mobiloperatører til å gjøre som de blir fortalt. Av den grunn kan man regne med at statlige aktører benytter seg av sårbarhetene i SS7 for etterretning, bekjempelse av kriminalitet og lignende [47, 48, 49]. Det finnes også private aktører som tilbyr lignende tjenester som man kan finne på dypnettet, men som kun tilbyr tjenester til statlige aktører som har myndighet til å avlytte og spore enheter [50]. Aktuelle kunder for slike tjenester kan befinne seg i land som mangler kompetansen som kreves for å gjennomføre slike angrep på egenhånd.

Feil i konfigurasjon

Hvis det er en feil i konfigurasjonen hos mobiloperatøren, kan det fort føre til en alvorlig sårbarhet som kan utnyttes. Et godt eksempel på dette er signaliseringstrafikken som slo ut Telenor-nettet i flere timer i 2016. Trafikken kom fra Luxembourg, og sikkerhetsmekanismene til en server levert av Ericsson var ikke laget for å håndtere disse signalene [51]. Dette problemet har blitt løst, men det kan fort skje at en annen mobiloperatør får samme problemet.

SIGTRAN protokollen blir brukt for å transportere SS7 signaler over Internett, og dette åpner opp for mange sårbarheter relatert til at SS7 blir sendt over IP. Det er svært viktig at mobiloperatører tar hensyn til dette.

Roaming

Nordmenn reiser ofte til utlandet, både for ferie og jobb. For å kunne bruke mobiltelefonen i utlandet må man benytte seg av roaming. For at man skal kunne bruke mobiltelefonen sin i utlandet må operatøren i det gjeldende landet ha tilgang til all trafikken som går mellom den mobile enheten og operatøren hjemme i Norge. Dette vil si at uansett hvilke tiltak hjemmeoperatøren har kommet med, kan de ikke beskytte deg. Det er kun gjesteoperatørens regler som gjelder [52].

Flere land, som blant annet Storbritannia, benytter seg av overvåkning av mobile enheter [53]. I november 2016 innførte Storbritannia en ny lov tilknyttet overvåkning. Denne loven tvinger alle internett- og teleselskaper til å lagre alle briters kommunikasjonshistorie i et helt år [53]. Dette vil sannsynligvis ikke bare gjelder britene, men også alle andre personer som befinner seg i landet i tillegg til trafikk som passerer gjennom landet.

Andre land som Frankrike [54], USA [55] og Sverige [56] har også blitt avslørt i å overvåke mobiltelefoner. Sverige er i motsetning til de andre landene åpne om overvåkningen. Lovene deres tillater lytting etter spesifikke ord, utrykk, telefonnummer og IP-adresser [56].

Dersom staten eier mobiloperatøren, eller som i Storbritannia innfører lover som innebærer overvåkning, kan de gi statlige organer lovlig tilgang til å overvåke mobile enheter uten å måtte benytte seg av sårbarhetene i SS7. Mobiloperatørene China Mobile i Kina [57, 58], Mobily i Saudi Arabia [59] og Telenor i Norge [60] er eksempler på mobiloperatører som er helt, eller delvis eid av staten.

Falske basestasjoner

Falske basestasjoner kan også benyttes til å overvåke posisjon, samtaler og SMS til mobile enheter. Falske basestasjoner kan kun overvåke mobile enheter som befinner seg i nærheten av basestasjonen, men er i gjengjeld enklere og billigere å anskaffe.

Falske basestasjoner er nokså billige å kjøpe, og kan også komme i små formfaktorer. I 2015 var det mulig å kjøpe en slik basestasjon for omtrent 10 000 kroner [61].

3.2.2 Mulige konsekvenser

Når man først har fått tilgang til infrastrukturen, er det mange muligheter som aktøren kan benytte seg av for å samle inn konfidensiell informasjon eller påføre skader med store konsekvenser.

Dekryptering

MAP-protokollen, som blant annet benyttes av SS7, har en melding som kalles *sendIdentification*. Denne meldingen kan brukes for å finne krypteringsnøkkelen som tilhører et gitt anrop, som gjør det enkelt for angriperen å dekryptere hele anropet [62]. Meldingen brukes normalt mellom MSC-er (Mobile Switching Centre), som er kjernen i nettverket som kobler basestasjoner til hverandre og står for ruting av mobiltrafikk [63]. MAP brukes av noder i mobilnettverket for å utveksle informasjon [64]. Meldingen *sendIdentification* er nødvendig å bruke når en mobil enhet beveger seg til et nytt område og en annen basestasjon må ta over kommunikasjonen, og burde derfor normalt kun forekomme fra mobiloperatører i naboland. Dersom man ikke implementerer filtrering av SS7-signaler, vil mobiloperatører fra alle verdensdeler kunne hente ut dekrypteringsnøkkelen med denne meldingen.

Oppfanging av SMS

En annen MAP melding som kan brukes av angripere er *updateLocation*. Denne meldingen sendes til HLR (Home Location Register) for å fortelle at mobiltelefonen til et offer har endret område, og befinner seg på et sted kontrollert av angriperen. Når noen prøver å sende SMS-en til offeret vil SMS-en sendes til angriperen i stedet, som deretter kan videresende SMS-en til offeret [65].

Sporing av brukere

MAP-meldingen *provideSubrscriberInfo* er ikke ment til bruk av andre enn operatørene selv, og kan føre til sårbarheter hvis andre får tilgang til denne. Denne meldingen brukes for å hente informasjon om en abonnent fra en MSC. Hvis angriperen har fanget opp IMSI-nummeret til abonnenten og adressen til MSC-en, er dette et aktuelt angrep for å kunne spore abonnenten. Hvis man også tar i bruk CAMEL protokollen i dette tilfellet, som er en protokoll for roaming og kartlegging av brukeren [66], er det mulig å spore brukeren helt ned til gatenivå. Hver basestasjon har en unik cell id, og hvis man finner denne kan man se nøyaktig hvor basestasjonen er plassert (For eksempel via databasen bak Opencellid). Denne plasseringen er en god indikator på hvor abonnenten befinner seg, spesielt i storbyer hvor det er høy konsentrasjon av basestasjoner [67].

3.2.3 Konkrete angrep

Til tross for at det har blitt vist gjentatte ganger hvordan angripere kan benytte seg av sårbarheter i SS7 til overvåkning og omdirigering av SMS-er, har det ikke blitt rapportert så mange angrep. En sannsynlig grunn til dette er at slike angrep baserer seg på flere kommunikasjonskanaler og dermed kan være vanskelig å oppdage.

Det mest konkrete angrepet som har blitt avdekket, ble rapportert av den tyske avisen Süddeutsche Zeitung i mai 2017. Den tyske mobiloperatøren O2-Telefonica bekreftet at kriminelle hackere hadde benyttet seg av sårbarheter i SS7 til å plukke opp SMS-er med engangskoder som ble benyttet som faktor nummer to av banker. Engangskoder på SMS er naturligvis ikke nok til å autentisere seg for banken, så angrepet bestod av flere deler. Først benyttet angriperne seg av Phishing via e-post, som ledet ofrene til nettsider som lignet på banken de normalt benyttet. Etter at angriperne hadde plukket opp innloggingsdetaljene til brukerne benyttet de seg av sårbarhetene i SS7 til å selv motta engangskodene som ble sendt via SMS og dermed autentisere seg som brukeren

[44, 45]. Den tyske avisartikkelen rapporterte at bankene er skremt siden de alltid har stolt på mobiloperatørenes sikre systemer [45].

3.3 Tiltak

For å forbedre sikkerheten i mobil infrastruktur kan man implementere en rekke tiltak. Vi har delt tiltakene opp i tiltak som vanlige forbrukere kan benytte seg av, og beskrivelse av hvilke tiltak som mobiloperatørene har implementert.

3.3.1 Tiltak implementert av mobiloperatører

Det er ikke mulig å si nøyaktig hvilke tiltak som er implementert ettersom mobiloperatørene holder dette hemmelig. Mobiloperatørene oppgir gjerne at de har implementert tiltakene som anbefales av GSMA og Nkom, men tiltakene fra disse aktørene er konfidensielle og kun tilgjengelig for mobiloperatører og statlige aktører som regulerer mobilnettverk [32, side 16][33].

Noen anbefalinger har blitt publisert, men dette er fra andre aktører som mobiloperatørene ikke har uttalt at de følger anbefalingene til. En av tilnærmingene er utformet av tyske Security Research Labs og har blitt trukket frem som et eksempel på tiltak av CSRIC V. En av tiltakene er bruk av brannmurer som blokkerer visse type signaler og sjekker opprinnelsen til en gitt type andre signaler [34].

En annen, mer praktisk tilnærming ble presentert av franske P1 Security på Black Hat konferansen i USA 2017. De har utviklet en SS7/Diameter-brannmur de kaller SigFW. Kildekoden til brannmuren er åpen og tilgjengelig på Github[3]. Det advares riktig nok om at implementasjonen kun er ment som referanse og ikke ment til å settes ut i drift, så den er trolig ikke like omfattende som filtreringen de norske mobiloperatørene benytter seg av [68].

Det kan være vanskelig å skille mellom legitim og ondsinnet trafikk ettersom mobiloperatørene må regne med at forbrukere reiser til utlandet. Utover å si at det ikke er mulig å beskytte helt mot angrep via SS7 [69], er det ikke noen informasjon om hvor effektive tiltakene som anbefales av GSMA og Nkom er. Det er derfor heller ikke mulig å si akkurat hvor god sikkerheten i mobilinfrastrukturen er den dag i dag.

3.3.2 Tiltak for forbrukere

Et tiltak som kan beskytte mot avlytting fra falske basestasjoner er å tvinge mobiltelefonen sin til å kun benytte 3G- og 4G-nettverket ettersom mange falske basestasjoner kun opererer på 2G-nettverket, men det er dessverre ikke alle mobile enheter som har støtte for dette [37]. Dersom man har en rootet Android-enhet med Qualcomm-brikkesett, kan man installere applikasjonen SnoopSnitch[4], som er utviklet av Security Research Labs. Applikasjonen kan advare mot blant annet falske basestasjoner ved å undersøke om en basestasjon ber om å bytte til svakere/ingen kryptering.

Et annet tiltak som beskytter mot avlytting via både falske basestasjoner og SS7, er bruk av ende-til-ende kryptering. Visse mobilapplikasjoner, som for eksempel Signal[5],

[3] SigFW: https://github.com/P1sec/SigFW
[4] SnoopSnitch: https://play.google.com/store/apps/details?id=de.srlabs.snoopsnitch
[5] Signal: https://signal.org/

WhatsApp[6] og iMessage[7], krypterer meldinger og samtaler hele veien frem til mottakeren.

Andre kjente applikasjoner som Skype[8], Google Hangouts[9] og Facebook Messenger[10], krypterer trafikk under transport mellom mobilapplikasjonen og serveren til tjenesteleverandøren. Facebook Messenger støtter ende-til-ende kryptering av meldinger, men ikke tale/videosamtaler. Dersom man stoler på tjenesteleverandøren, er transport-kryptering et godt alternativ til ende-til-ende kryptering.

3.4 Sikkerhet i norsk mobilinfrastruktur

Sikkerheten i norsk mobilinfrastruktur har allerede blitt dekket av Aftenposten etter avsløringene rundt falske basestasjoner i Oslo sentrum, men dette omfatter kun en liten del av sikkerheten i norsk mobilinfrastruktur. I denne seksjonen beskriver vi hvordan sikkerheten i mobilnettene til de største norske operatørene er.

3.4.1 Falske basestasjoner i Oslo sentrum

I Oktober 2013, startet et par journalister i Aftenposten på et tilsynelatende lite prosjekt. De skulle avsløre om det fantes falske basestasjoner (IMSI-fanger) i Oslo. Prosjektet førte til at store ressurser (i form av blant annet politietterforskninger og opprettelse av nye arbeidsgrupper) ble brukt i kampen mot mobilovervåkning. Aftenposten kom frem til at det var og muligens er en svært høy sannsynlighet for IMSI-fangere rundt flere sentrale bygninger i Oslo. Norske myndigheter og teleselskapene kunne ifølge Aftenposten, på det tidspunktet artikkelen ble publisert, gjøre langt mer for å beskytte mobilbrukerne [61].

Etter Aftenpostens avsløringer har det blitt registrert flere tilfeller der politiet har benyttet falske basestasjoner til overvåkning. Oslo-politiet skal i 2015 ha innrømmet at de benytter seg av falske basestasjoner gjennomsnittlig én gang i uken [70].

3.4.2 Tiltak mot SS7-sårbarheter i norsk mobilinfrastruktur

For å kartlegge sikkerheten i norsk mobilinfrastruktur, spurte digi.no i mars 2016 de største norske mobiloperatørene om hva de har gjort for å sikre mobilnettverket sitt mot angrep som benytter seg av sårbarheter i SS7-nettverket.

Telenor svarte at de har implementert filtrering av SS7-signaler slik GSMA og NKOM anbefaler. Ice svarte at de jobbet med å implementere de samme tiltakene, og med tanke på at dette var to år siden har de sannsynligvis implementert dette nå [69].

Telia ga et veldig vagt svar når de ble spurt av digi.no, og vi valgte derfor å sende dem en e-post og spørre litt mer spesifikt. Vi fikk da opplyst at Telia også har implementert tiltakene som anbefales av GSMA/NKOM[B.1].

Telenor har også opplevd at mobilnettverket gikk ned i nesten fire timer på grunn av et ukjent signal over SS7-nettverket, men det var en programvarefeil i programvare fra Ericsson som førte til denne nedetiden, og ikke en sårbarhet i SS7 [27].

Selv om de norske operatørene har implementert tiltakene som anbefales for å sikre mot angrep via SS7, er det ukjent om de har implementert tilsvarende tiltak for Diameter.

[6] WhatsApp: https://www.whatsapp.com/
[7] iMessage: https://support.apple.com/no-no/HT207006
[8] Skype: https://www.skype.com/
[9] Google Hangouts: https://hangouts.google.com/
[10] Facebook Messenger: https://www.messenger.com/

Hvor effektive tiltakene for SS7 er, forblir ukjent enn så lenge. Telenor skal ha oppgitt at ingen operatører var i stand til å hindre misbruk av SS7 helt da de svarte på spørsmålene til digi.no [69], og man kan dermed anta at tiltakene som anbefales av GSMA og NKOM ikke er nok til å sikre seg helt.

3.4.3 Kryptering i norske mobilnettverk

Krypteringsalgoritmen som kalles A5/1 og som har vært brukt gjennom mange år, er uheldigvis lett å knekke ved hjelp av rainbow-tables. Av den grunn må man benytte seg av sterkere kryptering, som for eksempel A5/3 krypteringen, også kjent som KASUMI. A5/3 krypteringen er standard på alle 3G-nettverk [71]. På 4G LTE-nettverk benyttes ulike algoritmer av ulike land. De mest vanlige krypteringsalgoritmene er SNOW 3G som er utviklet i Sverige, AES som er spesifisert av amerikanske NIST og ZUC som er utviklet i Kina [72, side 22].

For å øke sikkerheten i mobilnettverket har både Telenor og Telia innført støtte for A5/3-kryptering av trafikk som går over 2G [71][B.1]. Dette krever at mobiltelefonen støtter A5/3-krypteringen, men så godt som alle mobiltelefoner som har kommet ut etter 2007 støtter dette [73]. En viktig grunn til å bruke sterkere kryptering også på 2G nettverket til tross for at mobiltelefonene gjerne støtter 3G eller 4G, er at telefonsamtaler ofte går over 2G-nettverket. 4G baserer seg helt på pakke-svitsjing, og mobiltelefoner som ønsker å gjennomføre telefonsamtaler på 4G nettverket må derfor støtte "Voice Over LTE" (VoLTE). Dersom man har en telefon som ikke støtter VoLTE og ikke har 3G dekning, må man altså benytte seg av 2G-nettverket til å gjennomføre telefonsamtaler [74].

Trafikken er kun kryptert mellom den mobile enheten og mobiloperatørens utstyr, og det er opp til mobiloperatøren hvordan/om de krypterer trafikken under transport i sitt eget nettverk eller på vei til andre nettverk.

3.4.4 Huawei

Den største leverandøren til Telenor, og en av de største leverandørene til Telia når det gjelder mobilinfrastruktur, er Huawei [75]. Huawei er et privateid kinesisk selskap, og en av verdens største leverandører av utstyr til mobilinfrastruktur. I 2012 og 2013 startet en amerikansk og australsk boikott av Huawei, angivelig på grunn av sikkerhetsmessige årsaker [76]. USA mener at Huawei har bånd til den kinesiske stat og derfor legger inn bakdører til etterretning og industrispionasje [77].

Seks år etter at USA sin boikott av Huawei startet, har det fortsatt ikke blitt avdekket noen bakdører i Huawei-utstyr. Uavhengige analytikere fra Asia har påpekt at boikotten trolig skyldes politikk fremfor sikkerhetsmessige årsaker ettersom kun Huawei og ZTE boikottes, samtidig som problemstillingen er like aktuell for andre selskaper. Det påpekes også at veldig mange amerikanske selskaper har produksjon i Kina, uten at dette ser ut til å være et problem [78].

PST har uttrykt bekymring om bruken av Huawei-utstyr ettersom Norge ikke har sikkerhetssamarbeid med Kina [79], men Telenor oppgir at de har analysert risikoen og at den er akseptabel. Formuleringen til Telenor kan også tyde på at valget har vært drøftet med, og trolig klarert av norske myndigheter [80].

Huawei har tidligere blitt gransket av EU, men dette var på grunn av mistanke om prisdumping for å presse konkurrenter ut av markedet og ikke på grunn av mistanke om

bakdører [81]. Mange EU-land skal ha vært imot denne granskningen ettersom ingen av konkurrentene hadde levert noen klage til EU, og EU valgte senere å droppe denne saken [82].

3.5 Vurdering av kodebærere

En sentral del av problemstillingen går ut på hvor sikkert det er å bruke engangskoder på SMS til autentisering. I dette kapittelet vil vi først vurdere praksisen med å bruke SMS som kodebærer, og deretter sammenligner vi med andre kodebærere. Med "kodebærer" menes en teknologi som kan autentisere brukeren gjennom et annet medie enn mediet der brukeren startet autentiseringen. Mange er vant til slike kodebærere gjennom BankID, der man enten benytter seg av en kodebrikke eller et kodekort.

3.5.1 SMS som kodebærer

Som nevnt eksisterer det en rekke sårbarheter i mobilinfrastrukturen, og sikkerheten kan være ekstra dårlig i utlandet. De fleste rapportene som ble nevnt i seksjon 3.1 anbefaler å kryptere data som er sensitiv, og forutsatt at engangskodene benyttes til å autentisere brukere, burde engangskodene regnes som sensitive.

På grunn av hvor lett det er å plukke opp SMS-er, kan man ikke regne det som at en bruker autentiserer seg med "noe man har", som for eksempel en fysisk nøkkel. Brukeren autentiserer seg med noe tjenesteleverandøren sendte. En angriper må ikke ha mobiltelefonen til et offer for å autentisere seg, man trenger kun to ting brukeren vet: Passordet og koden som ble tilsendt på SMS.

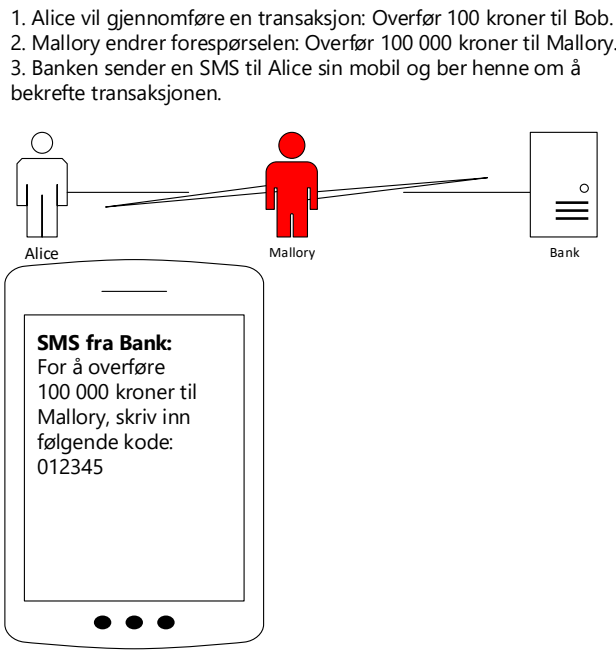
Denne koden kan enten plukkes opp ved å utnytte den sårbare infrastrukturen eller ved å benytte seg av phishing. Dette er grunnen til at flere aktører kaller autentisering ved hjelp av passord og engangskoder via SMS for to-trinns verifisering og ikke to-faktor autentisering [83, 84, 85].

I tillegg til sårbarhetene i mobil infrastruktur som lar angripere plukke opp og lese SMS-er i sanntid, er det også mulig å benytte seg av sosial manipulering for å kunne plukke opp andres SMS-er. Et vanlig angrep som kalles "SIM swap" (SIM-bytte) går ut på at angriperen skaffer seg et SIM-kort med telefonnummeret til offeret. Dette gjøres gjerne ved å utgi seg for å være offeret og be om et nytt SIM-kort fra operatøren til offeret [86].

En angriper kan i mange tilfeller finne nok informasjon om et offer fra Facebook, LinkedIn og andre kilder på internett. Dersom man klarer å få tak i et SIM-kort med telefonnummeret til offeret, trenger man bare sette SIM-kortet inn i en mobiltelefon og lese alle meldinger som kommer inn. Det er lite man kan gjøre for å beskytte seg mot slike angrep annet enn å beskytte personinformasjonen sin.

Til tross for ulempene, finnes det allikevel en fordel med SMS-er – de kan inneholde beskrivelse av transaksjoner. Dersom noen skulle bli offer for sosial manipulering eller et man-in-the-middle-angrep, kan beskrivelse av transaksjoner gjøre angrepet vanskeligere å gjennomføre. Et eksempel på dette er vist i figur 1 der Alice prøver å overføre 100 kroner til Bob.

Uheldigvis vil ikke alle personer være like oppmerksomme, og som beskrevet i en rapport om å styrke sikkerheten til SMS-basert autentisering på en brukervennlig måte fra 2008, var det kun 79% som var oppmerksomme på transaksjonsdetaljene når de



Figur 1: Beskrivelse av transaksjoner fra banken kan gjøre autentiseringen sikrere. Dersom Alice er oppmerksom vil hun oppdage at både summen og mottakeren er feil, og dermed ikke fullføre transaksjonen.

gjennomførte flere transaksjoner. Samme rapport kom med forslag om at brukerne skal kunne legge til kjente kontoer i en hviteliste. Dette gjør at man kun trenger å være oppmerksom når man skal betale/overføre til nye kontoer [87]. Det ser ikke ut til at effektiviteten av dette har blitt testet, men en slik løsning burde føre til enda bedre oppmerksomhet.

Når det gjelder andre aspekter enn sikkerhet, er SMS mye mer attraktivt enn konkurrentene. 98% av Norges befolkning hadde egen mobiltelefon i 2016 [88]. Samtidig var det omtrent 82% av befolkningen over 15 år som hadde smarttelefon [89]. Engangskoder på SMS er altså tilgjengelig for en større andel av befolkningen enn en autentiseringsapplikasjon.

SMS må heller ikke tilpasses til å støtte noe spesielt operativsystem på samme måte som en applikasjon. Dersom man utvikler en applikasjon, vil det trolig være nok å utvikle til Android og iOS, ettersom disse operativsystemene benyttes 99,6% av alle smarttelefoner solgt i siste kvartal av 2016 [90].

Til tross for disse fordelene er det alt for mange sikkerhetsproblemer til at vi vil anbefale praksisen med bruk av SMS til engangskoder. Anbefalte alternativer beskrives i seksjon 3.5.4.

Når er det greit å bruke SMS som kodebærer?

Man kan fint forsvare bruk av SMS som kodebærer dersom alternativet er mindre sikkert. Det vil for eksempel være mindre sikkert å kun autentisere brukere med ett trinn (som for eksempel passord).

Dersom man normalt benytter seg av en applikasjon for å generere engangskoder, vil det være hensiktsmessig å la brukere som ikke har en smarttelefon benytte seg av engangskoder via SMS. Dersom man tilbyr brukere å autentisere seg med engangskoder via SMS i tillegg til andre former for engangskode, er det derimot svært viktig å ikke la SMS-autentiseringen fungere som bakdør. Brukere som normalt benytter seg av en annen kodebærer skal altså ikke kunne be om en engangskode via SMS.

3.5.2 Alternative kodebærere

SMS er langt ifra den eneste kodebæreren man kan benytte seg av. Det finnes mange andre typer, og i denne seksjonen beskrives og sammenlignes alternative kodebærere som kan benyttes til å autentisere brukere.

E-post

E-post er på mange måter lik SMS fordi man kan sende valgfri tekst til en bruker, og brukeren er ikke avhengig av å ha noen spesiell applikasjon eller enhet for å kunne motta koden. I de fleste tilfeller trenger man kun tilgang til internett. Allikevel er det nok ikke like lett for alle brukere å benytte seg av e-post som kodebærer fordi de ikke nødvendigvis mottar e-poster på den mobile enheten sin. Hvis de kun benytter seg av Webmail, kan det være plagsomt å måtte logge inn på en slik portal hver gang de skal hente ut en kode. Omtrent 91% av Norges befolkning benyttet seg av internett til e-post i løpet av 2017, så de fleste vil ha mulighet til å motta engangskoder på e-post selv om det ikke nødvendigvis er praktisk [91].

Sikkerheten til e-post kan variere veldig. Dersom en bruker benytter seg av transportkryptering, som for eksempel TLS, vil sikkerheten være god. Det samme gjelder webmail så lenge webmail-portalen leveres over HTTPS. Dersom en bruker derimot ikke benytter seg av kryptering, vil alle e-poster sendes i klartekst. Uavhengig av om brukeren benytter seg av kryptering eller ikke, vil løsningen være utsatt for phishing på lik linje med SMS.

Fordi brukere kan oppleve det som vanskelig å hente ut koder fra e-post og at man ikke har kontroll over om brukere benytter seg av kryptering, vil vi anbefale å unngå e-post som kodebærer.

Fysisk kodegenerator

Fysisk kodegenerator er en type kodebærer som er godt utbredt i Norge takket være BankID. Dette innebærer både elektronisk kodebrikke og kodekort. En fysisk kodegenerator vil i de fleste tilfeller tilby god sikkerhet, men er allikevel sårbar ovenfor phishing på samme måte som SMS.

Den største ulempen med fysisk kodegenerator er nok prisen. Dersom man ikke benytter seg av BankID autentisering (som tar betalt for hver eneste autentisering som gjennomføres), men heller utvikler en egen autentiseringsmetode, må man også utstede en kodebrikke til hver eneste bruker. Man må også regne med ekstra kostnader for å erstatte kodegeneratorer som har gått tom for batteri eller som har blitt mistet.

Brukere kan oppleve det som litt plagsomt å måtte ha kodegeneratoren med seg til enhver tid de skal logge seg inn. Av den grunn kan det være hensiktsmessig å kombinere en slik løsning med en programvarebasert kodegenerator (applikasjon) som lar brukere med Android eller iOS benytte seg av smarttelefonen sin i stedet for en fysisk kodegenerator.

Sikkerhetsnøkler (Security Keys)

FIDO U2F security keys, også kalt sikkerhetsnøkler, er en lite utbredt [5], men veldig lovende metode for en faktor nummer to til autentisering [92]. For å autentisere seg med en sikkerhetsnøkkel, må man koble til sikkerhetsnøkkelen hver gang man ønsker å autentisere seg. Dette gjøres gjerne med USB på datamaskiner, og NFC eller lav-energi blåtann på mobile enheter. Sikkerhetsnøkler som støtter NFC og blåtann er gjerne litt dyrere, og enheter som støtter blåtann må i tillegg lades en gang i blant.

Når man knytter en sikkerhetsnøkkel opp mot brukeren sin på et domene, vil sikkerhetsnøkkelen overføre en offentlig nøkkel til serveren og lagre en tilkobling mellom det akutelle nøkkelparet og domenet.

Når man skal autentisere seg med en sikkerhetsnøkkel, vil sikkerhetsnøkkelen finne frem den private nøkkelen som er knyttet til domenet brukeren prøver å autentisere seg for, og deretter benytte den private nøkkelen til å signere data som brukes til autentisering [93]. Mer informasjon om FIDO U2F standarden er tilgjengelig på FIDO alliansen sine nettsider[11].

Den største fordelen med dette er at sikkerhetsnøklene ikke er sårbare ovenfor phishing. Man kan ikke lure sikkerhetsnøkkelen til å autentisere brukeren på feil domene, og man kan heller ikke lure brukeren til å gi fra seg noen kode på samme måte som med en engangskode på SMS.

Selv om sikkerheten til sikkerhetsnøkler er veldig god, kan det i noten tilfeller være vanskelig å benytte seg av. På iPhone kan man kun benytte sikkerhetsnøkler som støtter lav-energi blåtann, fordi Apple ikke tillater at applikasjoner fra App Store benytter seg av NFC-leseren. På Android-enheter må man benytte seg av NFC eller blåtann og vil dermed også være begrenset til de dyreste sikkerhetsnøklene. En av de mest kjente produsentene av FIDO U2F sikkerhetsnøkler er Yubico [94], og sikkerhetsnøkkelen deres støtter både USB og NFC. YubiKey NEO, koster 50 amerikanske dollar direkte fra produsenten. Dersom man skal kjøpe denne fra en norsk forhandler, kommer prisen fort opp i over 500 kroner med mva [95]. Yubico produserer også en billigere sikkerhetsnøkkel som kun har USB type A grensesnitt, men denne kan ikke brukes med mobile enheter [96].

Sikkerhetsnøkler er en veldig sikker løsning, men også ganske dyr. Av den grunn er det lite trolig at sikkerhetsnøkler vil være passende som faktor nummer to i en autentiseringsprosess for en tjeneste med mange brukere. Autentisering med sikkerhetsnøkler kan derimot være et veldig godt alternativ man kan tilby brukere som allerede har en sikkerhetsnøkkel og som ønsker ekstra sikkerhet.

Programvarebasert kodegenerator

Programvarebaserte kodegeneratorer er en stadig voksende trend. Det er enkelt å implementere og alle brukere med smarttelefon kan benytte seg av det, og det er derfor ikke så overraskende at mange tjenester har støtte for denne autentiseringsmetoden [5, 97].

Selv om "kun" 80% av Norges befolkning har smarttelefon, er andelen hele 99% for personer mellom 12 og 49 år. Avhengig av målgruppen kan man altså potensielt nå svært store deler av befolkningen med programvarebaserte kodegeneratorer [98].

De vanligste programvarebaserte kodegeneratorene er Authy og Google Authentica-

[11] Universal 2nd Factor (U2F) Overview: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html

tor. Disse og mange andre applikasjoner er implementasjoner av RFC6238 (TOTP: Time-Based One-Time Password Algorithm)[12].

Kort fortalt går metoden ut på at serveren genererer og lagrer en hemmelig nøkkel når brukeren registrerer seg. Denne nøkkelen sendes så til brukeren - gjerne som tekst og en QR-kode. Brukeren legger så nøkkelen inn i applikasjonen ved å scanne QR-koden eller skrive inn nøkkelen. Hver gang brukeren skal autentisere seg for serveren i ettertid, må brukeren oppgi en kode som kan hentes fra applikasjonen. Koden beregnes ved hjelp av en funksjon som tar den hemmelige nøkkelen og tidspunktet som parametere. Koden sendes så til serveren, som validerer om koden er korrekt ved å selv utføre samme beregning.

På lik linje med SMS er programvarebaserte kodegeneratorer sårbare ovenfor phishing ettersom brukeren må skrive inn en kode [99]. En metode som blant annet Google, Microsoft og Facebook benytter seg av for å unngå dette problemet, er at de ikke benytter seg av koder som tastes inn, men kun ber brukeren om å bekrefte innloggingen i en applikasjon. Når brukeren åpner denne applikasjonen, får de beskjed om å bekrefte ny innlogging fra angitt enhet som befinner seg på et omtrentlig sted.

En liten utfordring ved programvarebaserte kodegeneratorer som kun lagrer den hemmelige nøkkelen på selve enheten (som Google Authenticator), er at brukere som mister telefonen sin ikke vil kunne autentisere seg. Av den grunn må man ha en reserveløsning for å autentisere brukere på samme måte som man har mulighet til å gjenopprette et passord dersom man glemmer det, og dette er ikke alltid så lett å implementere sikkert. Mange angrep baserer seg på at en applikasjon har svak beskyttelse på gjenoppretting av passord, og det samme gjelder gjenoppretting av kodebærer [100, 101].

Noen tjenester løser dette ved å gi brukeren noen gjenopprettingskoder i det brukeren knytter kodebæreren til brukerkontoen sin, og som brukeren blir bedt om å skrive ned og oppbevare et trygt sted. Dette kan regnes som ganske sikkert men ikke til å stole på, fordi brukere kan miste disse kodene. Andre tjenester sjekker om brukeren ber om å gjenopprette kontoen sin fra en IP-adresse/enhet brukeren tidligere har autentisert seg på.

3.5.3 Sammenligning av kodebærere

For å finne ut hvilke kodebærere som er best egnet til ulike bruksområder har vi sammenlignet ulike kodebærere ved å vurdere på følgende områder:

- Hvor godt koder sikres mot at andre får tak i dem, som for eksempel om koden sendes over et nettverk og om autentisering med kodebæreren er sårbar ovenfor phishing.
- Hvor stor del av Norges befolkning kan benytte seg av den gjeldende kodebærer.
- Hvor mye koster det for en tjenesteleverandør å tilby autentisering ved hjelp av kodebæreren.

Oversikten er illustrert i tabell 1.

[12] RFC6238: https://tools.ietf.org/html/rfc6238

Tabell 1: Sammenligning av ulike kodebærere

	Kriterier				
Kodebærer		Sikkerhet	Sårbar ovenfor phishing?	Tilgjengelighet	Pris
	SMS	Dårlig	Ja	98% av Norges befolkning	Lav
	E-post	Varierer veldig	Ja	91% av Norges befolkning	Ubetydelig
	Fysisk kode-generator	God	Ja	Potensielt alle	Dyrt å utstede dersom man utsteder til alle brukere.
	Sikkerhets-nøkler	Veldig god	Nei	80% av Norges befolkning	Dyrt å utstede dersom man utsteder til alle brukere som ikke har.
	Programvare-basert kode-generator	God	Ja	80% av Norges befolkning	Ubetydelig

3.5.4 Anbefalt kodebærer

Det finnes ingen fasitsvar på hvilken kodebærer som er "best". Ulike kodebærere har ulike egenskaper, og av den grunn kan det i mange tilfeller være nyttig å tilby flere alternativer som brukerne kan velge mellom.

En god kodebærer som gjør det godt på de fleste områder er programvarebaserte kodegeneratorer, gjerne i form av applikasjoner til Android og iOS. Dersom man skal utvikle en autentiseringsportal som forventer mange brukere der alle brukerne skal benytte seg av en form for kodebærer, må man tilby alternative kodebærere slik at brukere uten smarttelefon også kan autentisere seg.

Dersom kostnad ikke er noe problem og sikkerhet står i fokus, vil vi anbefale å bruke fysiske kodegeneratorer eller eventuelt å benytte seg av BankID for brukere uten smarttelefon. Dersom man derimot har et mer begrenset budsjett, vil vi anbefale å benytte SMS som kodebærer for alle brukere som ikke har en Android eller iOS smarttelefon.

Dersom man kombinerer andre autentiseringsmetoder med engangskoder på SMS, må man sørge for at kun de som absolutt trenger å benytte seg av SMS gjør det, for eksempel ved å kun tilby SMS-basert autentisering for brukere som oppgir at de ikke har en Android eller iOS smarttelefon.

Dersom man ønsker å tilby ekstra sikkerhet, kan man gjerne implementere støtte for sikkerhetsnøkler. Ved å gjøre dette kan brukere som allerede har en sikkerhetsnøkkel benytte seg av denne og dermed sikre kontoen sin ekstra godt.

4 Analyse av sikkerhet i mobil autentisering

Sikkerhet knyttet til mobil autentisering og lagring av kryptografiske nøkler er avhengig av sikkerheten til operativsystemet som benyttes på den mobile enheten. Vi vurderte derfor sikkerheten i Android og iOS før vi gikk videre på sikkerhet knyttet til lagring av nøkler.

4.1 Fellestrekk for sikkerhet i Android og iOS

En rekke detaljer rundt sikkerheten til de mobile operativsystemene er felles. I denne seksjonen beskrives de felles sikkerhetsfunksjonene og effekten de har på lagring av kryptografiske nøkler og autentisering.

4.1.1 Prinsippet om minste privilegium

Prinsippet om minste privilegium er et viktig begrep når det gjelder Android og iOS. Da iOS og Android ble utviklet hadde man allerede lært av feilene som førte til at eldre operativsystemer var sårbare. Applikasjonene skulle ikke ha flere rettigheter enn nødvendig, og de skulle ikke ha mulighet til å kommunisere med hverandre annet enn via en definert API. I de tidligste versjonene av Android og iOS var det mange systemprosesser som kjørte med root-privilegier [102, 103]. Dette førte til at det ble funnet sårbarheter oftere og at sårbarhetene var mer alvorlige. I dag er det få prosesser som kjører med root-privilegier, og de fleste prosessene har kun de rettighetene de absolutt trenger. Dette har også ført til at både antallet sårbarheter og alvorligheten til sårbarhetene har sunket betraktelig [102], og dermed gjort det vanskeligere å angripe nøkkellagre.

4.1.2 Sandboxing

Android og iOS benytter seg av sandboxing for å isolere applikasjoner. På Android benyttes sikkerhetsfunksjoner som allerede eksisterer i Linux-kjernen ved at hver applikasjon kjører som en unik bruker [104, side 12-13]. På iOS er sandboxing implementert på samme måte som i macOS, der alle applikasjoner installert fra App Store kjører som samme bruker, men prosessene kun har tilgang til sin egen mappe. Disse "sandkassene" er også forsterket ved hjelp av tvungen adgangskontroll. På grunn av at sandboxing isolerer applikasjoner, vil ikke én applikasjon ha tilgang til hverken nøkler eller data til andre applikasjoner med mindre man eksplisitt angir at noe skal være delt.

4.1.3 Tvungen adgangskontroll

Både Android og iOS benytter seg av tvungen adgangskontroll (Mandatory Accesss Control, MAC) for å hindre uautoriserte handlinger. MAC-regler håndheves av kjernen til operativsystemet, og vil på den måten kunne beskytte systemet selv om en angriper skulle få root-privilegier i operativsystemet. Android benytter seg av SELinux og iOS benytter seg av TrustedBSD MAC rammeverket. Hvilke regler som er aktive og håndheves i de ulike operativsystemene er det vanskelig å få full oversikt over, men SELinux i Android har vist seg å være svært effektiv for å redusere konsekvensene av eventuelle sårbarheter

[102]. Selv om en angriper skulle få root-privilegier på en enhet er det ikke sikkert at angriperen vil kunne gjennomføre noen angrep mot nøkkellageret eller autentiseringen ettersom adgangskontrollen kan nekte prosessen tilgang.

4.1.4 Sikker oppstart

Både Android og iOS benytter seg av sikker oppstart (Secure Boot). Sikker oppstart går ut på at enheten bekrefter at programvaren ikke har blitt endret på mens enheten var avslått. Dersom en angriper skulle klare å endre på systemfilene til operativsystemet eller systemfilene til det sikre miljøet, vil angriperen kunne komme seg forbi alle sikkerhetsmekanismene. Sikker oppstart benytter seg av en form for tillitskjede der én sikker komponent bekrefter integriteten til en litt mindre sikker komponent.

Når iOS-enheter starter opp, vil prosessoren alltid starte med å kjøre kode fra en minnebrikke som kalles Boot-ROM. Denne brikken kan kun leses fra, og burde derfor ikke kunne endres på. Denne koden sjekker integriteten til bootloaderen, som sjekker integriteten til operativsystemet [105, side 5]. På Android-enheter utføres en lignende sjekk, der kjerne-funksjonen *"dm-verity"* benytter et hash-tre for å bekrefte integriteten til hele partisjonen der operativsystemet ligger. Ettersom *"dm-verity"* er en del av kjernen, må integriteten til *"dm-verity"* først bekreftes. Hvordan eller om integriteten til *"dm-verity"* bekreftes, er opp til de ulike smarttelefonprodusentene, men det anbefales sterkt [106]. Både Qualcomm og Samsung oppgir at de benytter seg av en form for slik integritetssjekk [107, 108], og man kan derfor anta at mesteparten av Android-smarttelefoner [109] benytter seg av sikker oppstart.

Denne sikre oppstarten er også med på å sikre at programvaren som kjører i TEE-en ikke er modifisert og sikrer dermed også nøkler i nøkkellageret. Dersom en angriper kunne endret på programvaren som kjørte i det sikre miljøet, ville angriperen kunne lagt inn programvare som henter ut nøklene.

4.1.5 Diskkryptering

Android-enheter med Android 4.4 eller nyere, og iPhone 3GS og nyere iOS-enheter, har støtte for diskkryptering. Dersom man aktiverer diskkryptering vil nesten alle filer være kryptert. Det eneste unntaket er filer som trengs for å vise frem brukergrensesnittet og deretter låse opp filer ved hjelp av pin-koden eller passordet som brukeren skriver inn. Data er kryptert hele tiden, men dekrypteres i sanntid når det leses fra disk. Dette krever at dekrypteringsnøkkelen ligger i minnet til enheten. Når brukeren låser opp enheten sin første gang etter den har startet opp, brukes låseskjerm-koden sammen med en maskinvarelagret nøkkel for å generere nøkkelen som dekrypterer data. Diskkrypteringen vil gjøre det vanskeligere å hente ut nøkler ettersom nøkkelen vil lagres kryptert, og dermed ikke mulig å lese, selv om man har fysisk tilgang til enheten og kapabilitet til å lese data direkte fra internlagringen.

4.1.6 Autentisering med biometri

Mange av dagens smarttelefoner, og så godt som alle toppmodeller, kommer med en eller flere sensorer som kan benyttes til biometrisk autentisering. Fordi dagens smarttelefoner er veldig tynne og full av funksjonalitet, må disse sensorene være svært små.

Fingeravtrykklesere på dagens smarttelefoner leser ofte kun en del av fingeren hver gang man legger fingeren på leseren. Av den grunn blir man ofte bedt om å presente-

re fingeren sin flere ganger når man setter opp fingeravtrykkautentisering. Dette gjøres for å sikre brukervennligheten ved at man ikke trenger å legge fingeren mot fingeravtrykkleseren nøyaktig på samme måte hver gang man skal autentisere seg i etterkant. En bivirkning av dette er at det blir lettere å lure fingeravtrykkleseren for andre aktører siden de kun trenger å matche én av de registrerte avtrykkene. Dersom brukeren har lagt inn flere fingeravtrykk, økes sannsynligheten for at andre aktører kan lure fingeravtrykkleseren ytterligere [110].

En alvorlig bivirkning av at kun deler av fingeravtrykket leses, er at det skal mindre til for at et annet fingeravtrykk matcher når man ikke sammenligner hele fingeravtrykket. I en rapport som ble publisert i april 2017 ble det avdekket hvordan man kunne generere fingeravtrykk som matcher med store deler av verdens befolkning. Ved hjelp av 5 genererte fingeravtrykk skal det ha vært mulig å utgi seg for å være 6,88% av brukerne i databasen med over 8000 kapasitive fingeravtrykk, dersom hver bruker kun hadde lagt inn én finger med 12 delvis avtrykk. Dersom en person har lagt inn avtrykk fra flere fingre, vil sannsynligheten for at en av de genererte fingeravtrykkene matcher være betydelig høyere [110]. Studiet benyttet seg ikke av mobile enheter, men heller av programvare for matching av fingeravtrykk. TouchID benytter seg også av en kapasitiv sensor for å lese fingeravtrykk, og resultatet burde derfor være nokså likt på mobile enheter.

Fingeravtrykklesere på mobile enheter er også sårbare ovenfor mer tradisjonelle angrep. Dersom en person tar på et glass eller en annen blank overflate med litt fettete fingre, legger man ofte igjen fingeravtrykket sitt. Dersom noen skulle få tatt bilde av dette fingeravtrykket, vil de ha mulighet til å gjenskape fingeravtrykket ved hjelp av for eksempel silikon eller lim. Tyske Security Research Labs har demonstrert i en video hvordan man kan lage slike avtrykk basert på et bilde, og dermed lure kapasitive fingeravtrykklesere både på datamaskiner og mobile enheter [111].

Det største problemet med fingeravtrykk og andre biometriske egenskaper, er at man ikke kan bytte fingeravtrykk på samme måte som man kan bytte et passord dersom det skulle komme på avveie. Hvis noen først får tak i et godt bilde av fingeravtrykket eller irisen til et offer, vil offeret aldri være helt trygg. Et fingeravtrykk som har kommet på avveie kan heldigvis kun utnyttes av noen med fysisk tilgang til enheten, og vil derfor beskytte godt mot alle bortsett fra de mest avanserte trusselaktørene.

For å gjøre fingeravtrykklesere sikrere, utvikles det nye metoder for å forsikre seg om at det faktisk er en ekte finger som leses av. Qualcomm har for eksempel utviklet en fingeravtrykkleser som bruker ultralyd til å lage et tredimensjonalt fingeravtrykk [112, 113]. Et enda viktigere tiltak for å sikre mobile enheter, er at iOS og Android ikke lar brukere låse opp enheten med fingeravtrykk rett etter man slår på enheten eller dersom enheten har vært låst de siste 24 til 48 timene [105, 114, 115, side 8]. Dette fører til at en eventuell angriper som prøver å låse opp en enhet ved hjelp av et falsk fingeravtrykk, har liten tid på seg til å gjennomføre angrepet.

Til tross for at sikkerheten i biometriske autentiseringsmetoder på mobile enheter ikke er så veldig god, kan det allikevel være mer sikkert enn å bare bruke en pin-kode. En pin-kode på fire til seks tegn er rask å knekke ved hjelp av et brute-force angrep. Et langt passord med høy entropi sørger for god sikkerhet, men er lite brukervennlig. Sannsynligheten for at mannen i gata vil skrive inn et langt passord hver gang han skal låse opp telefonen sin, er veldig lav. Det er også en risiko for at en angriper benytter seg

av skulder-surfing, en metode for å få tak i passord på ved å se over skulderen til noen som taster det inn, for å få tak i en pin-kode eller et passord.

Mobile enheter har beskyttelse mot brute-force angrep, men på iPhone ser det ut til å være en sårbarhet som gjør det mulig å komme seg forbi denne beskyttelsen. For eksempel finnes GrayKey, som er en liten boks som kan låse opp iPhone-enheter med pin-kode på fire tegn i løpet av omtrent to timer og pin-koder på seks tegn i løpet av tre dager [116]. GrayKey selges kun til politi og andre statlige organer som jobber med etterforskning av kriminalitet, men det er fullt mulig at noen andre kan ha oppdaget eller kommer til å oppdage samme sårbarhet og utnytte denne. Apple har implementert tiltak som skal gjøre det vanskeligere å låse opp enheter med GrayKey, ved at lightning-porten deaktiveres dersom enheten ikke har blitt låst opp de siste syv dagene [117].

En god løsning som sørger for god sikkerhet og brukervennlighet, er bruken av et langt passord kombinert med biometrisk autentisering. Dersom noen skulle få tak i en mobil enhet med et godt passord mens den er avslått, vil det være umulig å knekke krypteringen ved hjelp av dagens teknologi. Samtidig vil det være brukervennlig for brukeren å kunne autentisere seg ved å kun presentere en finger eller se på enheten for å låse den opp.

4.2 Sikkerhet spesielt for Android

Android er verdens mest utbredte operativsystem for smarttelefoner, med en markedsandel på 81,7% av alle solgte smarttelefoner i fjerde kvartal 2016 [118]. Fordi Android er et såpass utbredt operativsystem, er det også et svært ettertraktet mål for personer med onde hensikter. Angrep mot operativsystemet kan også være et bedre mål for eventuelle angripere fordi man også kan benytte seg av sosial manipulering for å få brukere til å installere applikasjoner som ikke tillates i den offisielle applikasjonsbutikken.

4.2.1 Påvirkning av produsent-tilpasning

Så godt som ingen Android-enheter produseres uten noen form for tilpasning fra produsenten. Google sine egne Pixel-modeller kommer med Google sine applikasjoner i form av Google Play, Gmail, Google Maps og mange andre applikasjoner og tjenester fra Google [119]. Android-enheter fra andre produsenter som selger enhetene sine til vestlige land, inkluderer gjerne både Google sine tilpasninger og i tillegg noen egne tilpasninger som Samsung sitt TouchWiz-grensesnitt, Huawei sitt EMUI-grensesnitt og Sony sitt Xperia-grensesnitt [120].

Smarttelefonprodusenter prøver å skille seg så mye som mulig fra konkurrentene slik at det kan fremstå som at de er bedre enn andre. Dette gjør de ved å legge på ekstra funksjonalitet i tillegg til å endre på brukergrensesnittet. Noen eksempler på dette er Samsung sin sikkerhetsløsning som de kaller Knox og Sony sin Stamina modus som skal spare batteri.

I en rapport fra 2013 konkluderes det med at gjennomsnittlig 85,78% av alle applikasjoner og tjenester som var forhåndsinstallert på Android-enhetene de undersøkte var overpriviligerte, og at mesteparten av disse overpriviligerte applikasjonene kom fra produsentens tilpasninger. Det ble det oppdaget at mellom 64,71% og 85% av sårbarhetene i telefonene de undersøkte fra Samsung, HTC, og LG hadde oppstått på grunn av produsentens tilpasninger [121]. Rapporten bemerket at HTC-modellen de testet var merkbart

sikrere enn modellen fra generasjonen før. Mye har nok skjedd de siste 4 årene, men resultatet fra Mobile Pwn2Own konkurransen i 2017 tyder på at tilpasningene til de ulike produsentene fortsatt påvirker sikkerheten til Android.

Ved Mobile Pwn2Own konkurransen i 2017 ble det ikke avdekket noen suksessfulle angrep som også var mulig å gjennomføre på Google sin Pixel-modell eller på enheter som kjører umodifiserte versjoner av Android. Det ble derimot avdekket angrep mot blant annet Samsung Galaxy S8 [122, 123].

Utfordringene til tredjeparts-produsenter av Android-enheter har også hatt positiv påvirkning på Android. Ettersom produsentene har vært veldig dårlige til å gi ut oppdateringer, har tjenester som tidligere var en del av operativsystemet blitt skilt ut til egne tjenester som kan oppdateres direkte gjennom Google Play Store [102]. Slike oppdateringer installeres automatisk hver gang man lader telefonen og er koblet til et WiFi-nettverk, forutsatt at automatiske oppdateringer ikke er skrudd av. Takket være denne endringen, kan noen sårbarheter tettes mye raskere enn dersom tjenestene var en del av operativsystemet.

4.2.2 Diskkryptering på Android

I Android blir dekrypteringsnøkkelen lagret i minnet, og blir værende i minnet så lenge enheten er påslått – selv om brukeren låser enheten. En konsekvens av dette er at en aktør med høy kapabilitet som får tak i enheten mens den fortsatt er påslått, vil gjennomføre et kaldstartsangrep (cold boot attack) for å hente ut dekrypteringsnøkkel. Dette er svært mye vanskeligere å gjennomføre på mobile enheter enn datamaskiner ettersom alle komponentene er veldig små og ikke laget for å kunne tas ut, men det er trolig mulig [124]. Ved bruk av filbasert kryptering i Android vil hver bruker ha en egen krypteringsnøkkel og brukere kan derfor ikke lese hverandres filer [125].

4.2.3 Oppdatering av enheter

Mangel på oppdateringer har lenge vært et problem blant Android-enheter [126, 127, 128]. Produsentene av Android-enheter tjener penger på å selge enheter, men ikke på å gi ut oppdateringer. Det har også vært utfordrende for mange produsenter å gi ut større oppdateringer fordi de i mange tilfeller er avhengige av samarbeid med både systembrikkeprodusenter og mobiloperatører i mange ulike land som vil legge inn sine egne tilpasninger [126].

Tall fra Google som er illustrert i tabell 2, viser at en stor andel Android-enheter kjører en utdatert versjon av Android. Android 6.0, som ble utgitt oktober 2015, og nyere versjoner av Android er fortsatt støttet og mottar sikkerhetsoppdateringer. Eldre versjoner av Android mottar ikke sikkerhetsoppdateringer [129]. Dette tyder på at Android Open Source Project kommer med sikkerhetsoppdateringer på inntil tre år gamle versjoner, noe som er mye lengre enn perioden på 18 måneder som det hittil har vært vanlig at produsentene kommer med oppdateringer til en enhet [130].

Mye tyder på at Android-produsenter har blitt flinkere til å oppdatere enhetene sine. I følge Google mottok 30% flere enheter sikkerhetsoppdateringer i 2017 enn i 2016 [122]. Google er med på å motivere produsentene til å oppdatere enhetene jevnlig ved hjelp av de nye bedrifts-anbefalingene sine[1]. For at Google skal anbefale en smarttelefon for

[1] Android Enterprise Recommended https://www.android.com/enterprise/recommended/

Versjon	Kodenavn	Andel	Sum	Støttet
2.3.3 - 2.3.7	Gingerbread	0.3%	38,6%	Nei
4.0.3 - 4.0.4	Ice Cream Sandwich	0.4%		
4.1.x	Jelly Bean	1.7%		
4.2.x		2.2%		
4.3		0.6%		
4.4	KitKat	10.5%		
5.0	Lollipop	4.9%		
5.1		18.0%		
6.0	Marshmallow	26.0%	61,4%	Ja
7.0	Nougat	23.0%		
7.1		7.8%		
8.0	Oreo	4.1%		
8.1		0.5%		

Tabell 2: Fordeling av versjon blant Android-enheter. Data samlet inn ved å se på versjonen til Android-enheter som besøkte Google Play Store fra den 10. til 16. april 2018. Utgitt av Android Open Source Project under CC BY 2.5 lisensen.

bedrifter, må den motta sikkerhetsoppdateringer innen 90 dager fra de blir utgitt av Android Open Source Project, og den må motta minst én større oppdatering [131].

En endring som er gjort i Android 8 (Oreo) som har gjort det enklere for produsenter å gi ut oppdateringer, er at versjonen har blitt utviklet med et ekstra maskinvareabstraksjonslag. Dette er gjort for å gjøre operativsystemet mer uavhengig av maskinvaren og dermed gjøre smarttelefonprodusentene mer uavhengige av brikkesettprodusentene når de skal gi ut fremtidige oppdateringer [102].

For å beskytte Android-enheter mot nedgraderingsangrep, benyttes versjonsbinding av nøkler i nøkkellageret [132], og fra og med Android 8 benyttes også nedgraderingsbeskyttelse for operativsystemet [133].

4.2.4 Installasjon av applikasjoner

På Android-enheter installeres normalt applikasjoner fra applikasjonsbutikken Google Play. Man kan også installere applikasjoner ved å laste ned en applikasjonspakke-fil (APK-fil) og installere denne direkte. For å kunne installere applikasjoner som ikke kommer fra Google Play, må man inn i innstillinger for å aktivere denne funksjonen, for deretter å ignorere alle advarsler man får når man prøver å installere applikasjonen. Dette er en angrepsvektor som er avhengig av sosial manipulering, men som allikevel er fullt mulig å gjennomføre. Store andeler av angrep som offentliggjøres er avhengige av at brukeren ignorerer advarslene [134].

For å redusere risikoen tilknyttet installasjon av applikasjoner fra andre kilder enn Google Play, har Google utviklet Play Protect, som har mange likhetstrekk med antivirusprogramvare man normalt finner på PC-er [135]. Play Protect scanner gjennom applikasjoner som brukeren prøver å installere, og dersom tjenesten oppdager at en applikasjon er potensielt skadelig[2], vil ikke brukeren få lov til å installere applikasjonen. Google oppgir at andelen potensielt skadelige applikasjoner som ble installert fra andre kilder enn Google Play, ble redusert med over 60% takket være Play Protect [122].

[2]Klassifikasjon av potensielt skadelige applikasjoner: https://source.android.com/security/reports/Google_Android_Security_PHA_classifications.pdf

Biometrisk autentisering på Android-enheter

Hvilke sensorer som er tilgjengelig for biometrisk autentisering på Android-enheter, varierer veldig med både produsent og prisklasse.

Android hadde tidligere en API for ansiktsgjenkjenning kalt Face Unlock. Fordelen med denne autentiseringsmetoden er at den kun krever et front-vendt kamera for å kunne benyttes. Face Unlock hadde store problemer med sikkerheten og slet også med pålitelighet. Det ble eksperimentert med en sjekk om brukeren var levende ved å for eksempel se om brukeren blunket [136], men Face Unlock ble etter hvert fjernet fra Android [137].

Google valgte senere å lansere ansiktsgjenkjenning under det nye navnet Trusted Faces, som en funksjonalitet i Google Smart Lock for Android. Google Smart Lock er ikke en del av Android-operativsystemet, men en Google-tjeneste på lik linje med Google Maps [138]. Trusted Faces har også lav sikkerhet ettersom et vanlig kamera kun kan se i to dimensjoner og dermed ikke kan skille mellom et ansikt og et bilde av et ansikt [139].

Fordi Smart Lock ikke er en del av Android, finnes det heller ingen API som applikasjoner kan benytte seg av for å autentisere brukere ved hjelp av ansiktsgjenkjenning. Dersom man ønsker å autentisere brukere av en applikasjon ved hjelp av ansiktsgjenkjenning, må man derfor benytte seg av tredjeparts API-er. Microsoft tilbyr sin Face API som kan estimere hvor sannsynlig det er at to ansikter tilhører samme person [140], som man deretter kan benytte seg av til å autentisere brukeren. Kairos AR er et annet selskap som tilbyr en API for ansiktsgjenkjenning som blant annet kan prøve å verifisere om et bilde er av samme person som tidligere har blitt registrert [141]. Begge disse tredjeparts API-ene krever at bildene som skal benyttes til autentisering sendes til tjenesteleverandørens systemer, noe som innebærer at man må ha internett-tilkobling for å kunne autentisere seg. I tillegg finnes det ingen måte å benytte seg av ansiktsgjenkjenning til å låse opp nøkler som er lagret i et maskinvarestøttet nøkkellager.

På bakgrunn av alle disse ulempene anser vi ansiktsgjenkjenning med et vanlig kamera som en svært dårlig løsning. På enheter som ikke har noen annen biometrisk sensor, vil vi foreslå å heller benytte seg av en pin-kode eller et passord.

De siste toppmodellene til Samsung er utstyrt med både fingeravtrykkleser, iris-scanner og ansiktsgjenkjenning. De tilbyr også en API for autentisering av brukeren som kalles Samsung Pass. Samsung Pass har en API som lar applikasjonsutviklere be enheten om å autentisere brukeren [142]. Noen ulemper med denne API-en er at den kun er tilgjengelig på Samsungs toppmodeller, og man kan ikke bestemme hva brukeren skal benytte for å autentisere seg. Brukeren kan selv velge om passord, fingeravtrykk eller iris skal benyttes. Man har altså ingen direkte API for å autentisere brukere med iris-scanneren. Iris-scanneren til Samsung er mye sikrere enn ansiktsgjenkjenning med et vanlig kamera, men kan fortsatt lures. Omtrent en måned etter at Samsung Galaxy S8 ble sluppet ut, ble Iris-scanneren lurt av noen medlemmer av den tyske hacke-klubben Chaos Computer Club. Alt de trengte var et fotografi, tatt med et fotoapparat som kan fange opp infrarødt lys fra et par meters avstand unna offeret, en printer og en kontaktlinse [143].

Det er én API for biometrisk autentisering vi vil anbefale til Android-enheter, og det er fingeravtrykk-autentisering. Android har siden versjon 6.0 hatt en standardisert API for fingeravtrykk-autentisering [144], og man kan i tillegg beskytte nøkler lagret i et nøkkellager ved å kreve at brukeren autentiserer seg med fingeravtrykk [145].

4.3 Sikkerhet spesielt for iOS

iOS er verdens nest mest utbredte operativsystem for smarttelefoner, med en markedsandel på 17,9% av alle solgte smarttelefoner i fjerde kvartal 2016 [118]. Fordi alle applikasjoner som skal installeres på en iOS-enhet stort sett må installeres via App Store, er angrepsoverflaten til iOS mindre enn angrepsoverflaten til Android.

4.3.1 Diskkryptering på iOS

iOS sin diskkryptering sørger for et ekstra lag med sikkerhet utover det som er tilgjengelig i Android. Apple benytter seg av en ekstra nøkkel til kryptering og dekryptering som kastes ut av minnet hver gang enheten låses. Ved å kryptere for eksempel bilder, e-post og meldinger med denne nøkkelen, vil disse dataene være veldig godt beskyttet så lenge enheten er låst [105, side 13-15][124], og ikke bare når enheten er avslått slik som på Android-enheter.

4.3.2 Oppdatering av enheter

Fordi Apple har kontroll over både programvaren og maskinvaren til alle enheter, er det enklere å holde enhetene oppdatert. Apple oppgir at de støtter enheter så lenge det er teknisk mulig, og da iOS 11 nylig ble gitt ut i september 2017, fikk iPhone 5S, som ble utgitt 4 år tidligere, også denne oppdateringen. Denne praksisen fører til at de fleste iOS-enheter som fortsatt er i bruk blir oppdatert til nyeste versjon [146].

4.3.3 Installasjon av applikasjoner

For at man skal kunne publisere applikasjoner i App Store, må man først registrere seg som utvikler hos Apple, noe som krever at man identifiserer seg med navn og adresse i tillegg til å betale en avgift på 99 dollar i året. Deretter må applikasjonene man vil publisere i App Store gå gjennom en manuell godkjenning basert på en rekke kriterier. Denne prosessen er svært effektiv for å hindre skadevare i å bli installert på iOS-enheter, men er ikke perfekt. Det har vært avdekket minst 6 tilfeller der skadevare har kommet seg inn i App Store, og ved en av tilfellene skal det ha blitt rapportert om 344 infiserte applikasjoner [147].

Det finnes også en metode for å kunne installere applikasjoner fra andre kilder enn App Store. Apple tilbyr noe de kaller “Developer Enterprise Program”, som er laget for at selskaper skal kunne tilby intern programvare til sine ansatte. Dette benyttes til å distribuere blant annet adware til brukere som ikke tenker på risikoen knyttet til å laste ned applikasjoner fra andre kilder enn App Store [148].

Biometrisk autentisering på iOS-enheter

Apple har i flere år utstyrt iPhone-modellene med en fingeravtrykkleser de kaller TouchID. I 2017 lanserte Apple iPhone X som er utstyrt med en rekke kameraer og sensorer som benyttes til ansiktsgjenkjenning. Apple kaller denne løsningen FaceID, og oppgir at sannsynligheten for at en tilfeldig person kan låse opp enheten din med FaceID er 1/1 000 000, mot 1/50 000 med TouchID [105, side 8].

Det vietnamesiske sikkerhetsselskapet Bkav har allikevel greit å lure FaceID ved hjelp av masker. Alt man trenger for å lure FaceID er et par bilder av offeret tatt fra ulike vinkler, der et bilde av øynene tas med et infrarødt kamera, en 3D-printer for masken og en vanlig printer for øynene [149]. Det skal ha kostet litt under 200 amerikanske dollar

å lage masken, som tilsvarer omtrent 1600 norske kroner. Et slikt angrep er altså mye vanskeligere å gjennomføre enn et angrep mot ansiktsgjenkjenning som kun benytter seg av et vanlig kamera, men fortsatt gjennomførbart for avanserte trusselaktører.

Dersom man utvikler en applikasjon som benytter seg av biometrisk autentisering på en iOS-enhet, vil kallet være uavhengig av maskinvaren som kjører på enheten. Dersom enheten har TouchID vil fingeravtrykk benyttes, og dersom enheten har FaceID vil ansiktsgjenkjenning benyttes. Apple anbefaler at man allikevel utfører en sjekk på hvilken type autentisering som er tilgjengelig, slik at man ikke ber brukere med FaceID om å autentisere seg med TouchID eller omvendt.

4.4 HTTPS

Den enkleste måten å sikre kommunikasjonen mellom applikasjonen og serveren er å benytte seg av HTTPS, som sørger for både konfidensialitet og integritet. Med mindre man har svært høy kompetanse innen kryptografi, er HTTPS og andre kjente og sikre protokoller, sannsynligvis det beste valget. På den måten vet man at protokollen man benytter seg av er velprøvd og ikke inneholder noen grunnleggende sårbarheter [150]. Dersom man ikke ønsker å benytte seg av HTTP, kan man naturligvis benytte seg av TLS til å sikre annen trafikk.

Forutsatt at man velger en sikker chiffreringspakke (cipher suite), er krypteringen som benyttes i HTTPS veldig god. “TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384” og “TLS_DHE_RSA_WITH_AES_256_GCM_SHA384” er eksempler på sikre chiffreringspakker som sørger for Perfect Forward Secrecy. Perfect Forward Secrecy innebærer at det genereres nye nøkler for hver økt, slik at tidligere økter ikke kompromitteres selv om den private nøkkelen kompromitteres [151, 152]. For å teste hvor sikker HTTPS-krypteringen på en server er, kan man benytte seg av Qualys SSL Labs Server Test[3].

4.4.1 Problemer med HTTPS

Det eksisterer en potensielt stor sårbarhet ved HTTPS som man burde implementere tiltak mot dersom man har tenkt å bruke HTTPS for å sikre kommunikasjonen til serveren. For at en server skal kunne autentisere seg for klienten må sertifikatet til serveren være signert av en Certificate Authority. Hvilke CA-er som klienten aksepterer at kan ha signert et sertifikat, er opp til klienten. På nettsiden til Electronic Frontier Foundation sitt prosjekt “SSL Observatory” nevnes det at omtrent 650 CA-er aksepteres av Mozilla og Microsoft [153], og ifølge boka Android Security Internals var det over 100 CA-er som ble akseptert av Android inntil versjon 4.0 [104, side 167].

Dersom kun én av disse aksepterte CA-ene kompromitteres, vil det bli mulig for angripere å autentisere seg som en hvilken som helst annen server. Dette har skjedd gjentatte ganger, enten ved at CA-en feilaktig utsteder et sertifikat til noen som ikke eier domenet de ber om sertifikat til, utsteder et sertifikat til et ugyldig domene (som for eksempel localhost) eller at andre får tak i den private nøkkelen til CA-en [154, 155]. Det mest kjente tilfellet er fra 2011, der den nederlandske CA-en DigiNotar ble kompromittert og utstedte sertifikater for en rekke kjente domener til hackere. Rapporten som ble publisert etter granskningen av hendelsen rapporterte at det var over tre hundre tusen som hadde blitt ofre for angrep som utnyttet denne sårbarheten [156].

[3] SSL Server Test: https://www.ssllabs.com/ssltest/

I tillegg til at CA-er kan kompromitteres av hackere, er det også mulig at statlige organer kan presse CA-er til å utstede falske sertifikater. CA-ene som aksepteres av de største nettleserne er spredt utover 50 land, og sannsynligheten for at minst én av landene benytter seg denne metoden til etterretning er derfor høy [157].

Noen svært effektive tiltak for å hindre denne formen for misbruk av HTTPS, er Certificate Pinning og Certificate Transparency. Disse teknologiene løser problemet på ganske forskjellige måter, og passer derfor best til litt forskjellige bruksområder. Det er også mulig å kombinere Certificate Pinning og Certificate Transparency for best mulig sikkerhet [158]. Ved å benytte seg av HTTPS med certificate pinning og/eller certificate transparency verifisering vil kommunikasjonen mellom applikasjonen og serveren være svært godt sikret.

4.4.2 Certificate pinning

Certificate pinning, som er beskrevet under seksjon 2.2.8 kan være upraktisk å bruke på websider. Dersom CA-en som har signert sertifikatet til et domene blir kompromittert, kan det føre til langvarig tjenestenekt fordi man umiddelbart må bytte CA og få nytt sertifikat, men alle brukere som har låst domenet til det gamle sertifikatet vil bli nektet tilgang til websiden helt frem til utløpsdatoen. Man kan redusere konsekvensen av dette ved å gi pin-en en utløpsdato som er i nær fremtid, men dette vil også gjøre certificate pinning mindre effektivt ettersom brukere som ikke besøker siden jevnlig ikke vil bli beskyttet.

På grunn av risikoen for tjenestenekt, har Google valgt å forelde støtten for certificate pinning i versjon 67 av Chrome. De anbefaler å heller benytte seg av certificate transparency(4.4.3) til websider [159, 160].

På mobilapplikasjoner fungerer certificate pinning mye bedre enn på websider ettersom man kan oppdatere applikasjonene når som helst, og de oppdateres fra en applikasjonsbutikk som er uavhengig av domenet som pin-en eventuelt blokkerer. Man kan altså sette opp pins med utløpsdato mange år frem i tid, og dersom man skulle få behov for å oppdatere hvilke pins som applikasjonen aksepterer, må man bare oppdatere applikasjonen. Forutsatt at brukerne ikke har deaktivert automatiske oppdateringer, vil oppdateringer installeres automatisk når en smarttelefon lader og er koblet til et nettverk som ikke er forbruksmålt. Man kan også benytte seg av push-notifikasjoner og be brukere om å oppdatere applikasjonen før de kan bruke den.

Certificate pinning støttes av Android og er beskrevet i seksjon 5.2.1. iOS har ikke direkte støtte for certificate pinning, men applikasjonsutviklere kan implementere det gjennom biblioteker som TrustKit[4].

4.4.3 Certificate Transparency

Certificate Transparency, som er beskrevet i seksjon 2.2.9, støttes av iOS, og applikasjonsutviklere kan angi om certificate transparency skal håndheves i info.plist-filen til applikasjonen. Apple oppgir at man allikevel trenger støtte for å sjekke og sperre sertifikater, noe som kan gjøres ved hjelp av "Online Certificate Status Protocol (OCSP) stapling" [161].

Android støtter ikke certificate transparency direkte, og det ser heller ikke ut til å være noen biblioteker tilgjengelig, men Google har laget et eksempel for hvordan man

[4]TrustKit: https://github.com/datatheorem/TrustKit

kan implementere verifikasjon av certificate transparency i Java[5]. Ettersom det vil være en god del arbeid å lage produksjonsklar kode for dette, vil det sannsynligvis være bedre å benytte seg av certificate pinning på Android inntil videre.

4.5 Lagring av kryptografiske nøkler

Enhver smarttelefon må ha mulighet til å lagre nøkler for at brukere skal kunne autentisere seg ovenfor tjenester på internett. Det finnes mange ulke måter man kan lagre nøkler på, der de ulike metoden har ulike fordeler og ulemper. I dette kapittelet sammenlignes ulike metoder for å lagre nøkler på.

4.5.1 Lagre nøkler i applikasjonsmappen

Den enkleste metoden er å lagre nøkler i klartekst i applikasjonsmappen. Forutsatt at operativsystemet gjør jobben sin, skal data i applikasjonsmappen være sikret mot at andre applikasjoner som er installert av brukeren får tilgang til det. Selv om ikke andre applikasjoner som er installert av brukeren har tilgang til data, kan fortsatt noen applikasjoner lese data fra applikasjonsmappen og visse applikasjoner som er forhåndsinstallert fra produsenten vil sannsynligvis ha tilgang til dette. Forutsatt at manifestet til applikasjoner ikke oppgir at det ikke er tillatt å ta sikkerhetskopi av applikasjonen, vil enhver med mulighet til å låse opp telefonen også ha mulighet til å hente ut alle data fra applikasjonsmappen.

En litt tryggere versjon av samme løsning er å kryptere nøkkelen med en statisk nøkkel som er hardkodet i applikasjonen. Dersom noen med ferdigheter innen reverse-engineering av applikasjoner skulle ønske å hente ut nøkler fra applikasjonen, vil de enkelt kunne komme seg forbi denne typen beskyttelse.

Ettersom data lagret i applikasjonsmappen kun sikres til en viss grad, anbefales det å heller benytte seg av operativsystemets nøkkellager for lagring av sensitiv informasjon [162]. Dersom man kun vil sikre nøkler, kan man lagre nøkler direkte i nøkkellageret, men dersom man ønsker å sikre større mengder informasjon, kan man lagre en nøkkel i nøkkellageret og benytte seg av denne nøkkelen til å kryptere og dekryptere data som lagres i applikasjonsmappen [163].

4.5.2 Lagre nøkler i et nøkkellager

Den anbefalte praksisen for lagring av nøkler er å benytte seg av operativsystemets nøkkellager [162]. Android sitt nøkkellager kalles Keystore[6] og iOS sitt kalles Keychain[7].

Dersom man ønsker å beskytte nøkler på en enhet som ikke har maskinvarestøttet nøkkellager, eller dersom man ønsker å beskytte nøkler mot aktører med høy kapabilitet og fysisk tilgang til den mobile enheten, må man beskytte nøkkelen med et bruker-angitt passord. Dersom man angir dette vil nøkkellageret benytte dette passordet til å kryptere nøkkelen i nøkkellageret [164, 165]. Brukeren må da angi det samme passordet hver gang nøkkelen skal benyttes. Dersom dette skal implementeres må man benytte seg av passord med høy entropi for å beskytte seg mot brute-force angrep, og det kan være lite brukervennlig å be brukeren om å skrive inn lange passord på mobile enheter. Av den

[5] CTVerifier.java: https://github.com/google/conscrypt/blob/master/platform/src/main/java/org/conscrypt/ct/CTVerifier.java

[6] Android Keystore: https://developer.android.com/training/articles/keystore.html

[7] iOS Keychain: https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_keychain

grunn vil vi kun anbefale bruker-angitte passord for å beskytte nøkler det er svært kritisk at ikke kommer på avveie, eller beskytte nøkler som ikke er lagret i et maskinvarestøttet nøkkellager.

En svakhet knyttet til denne løsningen er at passordet må gå gjennom det usikre miljøet,og kan dermed bli plukket opp av en angriper som har oppnådd root-privilegier [166]. Denne løsningen vil riktig nok beskytte nøkkelen fra en angriper som kun har lese-rettigheter til filsystemet. Det er mulig at obligatorisk adgangskontroll hindrer en angriper med root-privilegier fra å gjennomføre et slikt angrep, men vi fant ingen konkrete bevis på at dette er implementert.

En sikrere metode å beskytte nøkler på er å kreve at brukeren autentiserer seg med biometri hver gang nøkkelen skal benyttes. Grunnen til at dette er sikrere er at den biometriske autentiseringen gjennomføres av det sikre miljøet, og innloggingsdetaljer går dermed aldri gjennom det usikre miljøet. Dette forutsetter naturligvis at nøkkellageret er maskinvarestøttet og ligger i det sikre miljøet.

4.5.3 Maskinvarestøttet nøkkellager

Dersom det skulle eksistere en eller flere alvorlige sårbarheter i programvaren til en mobil enhet, vil en angriper kunne utnytte disse til å ta full kontroll over programvaren på enheten. Både Android og iOS benytter seg av obligatorisk adgangskontroll slik at selv en angriper med root-privilegier ikke burde ha tilgang til alle ressurser på enheten, men man burde ikke anta at dette er nok til å holde seg beskyttet mot alle angrep.

Ved å benytte seg av et nøkkellager som er maskinvarestøttet i stedet for å kun være implementert ved hjelp av programvare, kan man hindre at en angriper får tak i nøklene til tross for at angriperen har root-privilegier siden nøkkellageret kjører i et eget sikkert miljø som ikke stoler på det usikre miljøet.

På Android har man mulighet til å kreve at brukeren autentiserer seg med fingeravtrykk for hver eneste kryptografiske operasjon. Dette gir god sikkerhet ettersom det er TEE-en som både verifiserer at fingeravtrykket er korrekt og gjennomfører den kryptografiske operasjonen [167]. iOS hadde en tilsvarende funksjon tidligere, men denne ble foreldet i iOS 11.3 [168]. iOS har allikevel en lignende funksjon. Man kan kreve at brukeren autentiserer seg med enten Touch ID/Face ID eller låseskjerm-koden sin [169]. Dette øker angrepsoverflaten, men kan være litt mer brukervennlig.

Det er verdt å merke seg at Android sin API kan endre seg i fremtiden. Det eneste kravet som settes på nøkkelen er at brukeren autentiserer seg for hver kryptografiske operasjon. Den eneste metoden for å utføre dette den dag i dag er å autentiserer seg med fingeravtrykk, men dersom Android for eksempel får en API for autentisering ved hjelp av iris, er det mulig at iris også kan brukes til å låse opp nøkkelen [145].

Når man genererer eller lagrer en nøkkel i et nøkkellager, må man angi hva slags type nøkkel og hvilke operasjoner man ønsker å gjennomføre med nøkkelen. I vårt konseptbevis spesifiserte vi at et elliptisk kurve (EC) nøkkelpar skulle genereres og at det skulle benyttes til å signere med SHA-256. Ettersom det maskinvarestøttede nøkkellageret støttet denne typen nøkler og signering, ble nøkkelen generert og lagret i det maskinvarestøttede nøkkellageret. 256 bits EC nøkkelpar er eneste type nøkler som støttes av Apple sin Secure Enclave [170], og ved å benytte seg av samme type nøkkel på Android og iOS, trenger man kun å implementere én felles backend. 256 bits EC nøkkelpar vil derfor

være den mest aktuelle nøkkeltypen i systemer som utvikles for både Android og iOS.

Hva vil det si at et nøkkellager er maskinvarestøttet?

Både iOS og Android støtter å lagre kryptografiske nøkler i et maskinvarestøttet nøkkellager, på engelsk kalt "hardware-backed keystore". Hvordan dette implementeres på ulike enheter varierer litt, men har mange fellestrekk; Man kan opprette to ulike miljøer på enheten, der ett miljø regnes som usikkert og ett miljø regnes som sikkert. I det usikre miljøet kjører det vanlige operativsystemet (Android/iOS) og alle applikasjoner som brukeren installerer. I det sikre miljøet kjøres et eget sikkert operativsystem [171, 172].

For at en nøkkel ikke skal kompromitteres av sårbarheter i det usikre miljøet, må nøkkelen aldri være tilgjengelig utenfor det sikre miljøet. Av den grunn lagres ikke bare nøklene i det sikre miljøet, men de genereres og benyttes også her. Hver gang det usikre miljøet vil gjennomføre en kryptografisk operasjon (kryptere, dekryptere, signere osv..), sendes data til det sikre miljøet der dette miljøet tar seg av den kryptografiske operasjonen og på den måten unngår å gjøre nøkkelen tilgjengelig for det usikrede miljøet [167].

Man kan også importere kryptografiske nøkler til det maskinvarestøttede nøkkellageret, men nøklene vil da gå gjennom det usikrede miljøet. Dersom en angriper har kontroll over enheten under importen, vil angriperen kunne plukke opp nøkkelen. Etter at importen er ferdig og nøkkelen har blitt slettet fra det usikrede miljøet, vil nøkkelen være sikkert lagret.

Om nøklene lagres i et programvarebasert eller maskinvarestøttet nøkkellager er ikke noe applikasjonsutviklere kan bestemme. Operativsystemet vil selv velge å lagre nøklene på sikrest mulig måte. Det vil ofte være et begrenset antall typer nøkler som støttes av det maskinvarestøttede nøkkellageret, og andre typer nøkler vil kun sikres av programvaren [167, 26].

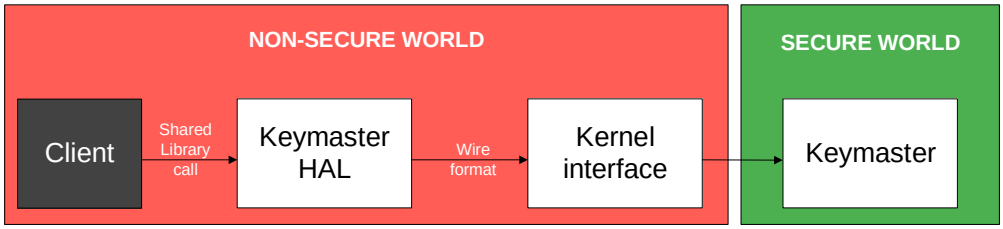
Android

På Android kan man utføre en sjekk på en nøkkel er lagret i et maskinvarestøttet nøkkellager[8]. Denne sjekken kan brukes for å vurdere hvor sikkert nøkkelen er lagret og dermed for eksempel tilpasse levetiden til nøkkelen.

Akkurat hvordan man implementerer maskinvarestøtten på Android vil være avhengig av hvordan produsenten velger å lagre nøklene. Så lenge produsenten skriver nødvendig kode for maskinvareabstraksjonslaget (HAL, Hardware Abstraction Layer) vil alle kall fra applikasjoner fungere som normalt uavhengig av hvordan nøkkellageret er implementert på enheten. Arkitekturen til et maskinvarestøttet nøkkellager er illustrert i figur 2.

Den vanligste måten i implementere maskinvarestøttede nøkkellagre på Android-enheter, er ved hjelp av ARM's TrustZone-teknologi [173, 174]. Man har da gjerne en egen nøkkellager-tjeneste som kjører i TEE-en til den mobile enheten. Det er allikevel mulig at noen produsenter velger å benytte seg av et Secure Element. Ved bruk av et Secure Element vil løsningen ligne svært på Apple sin Secure Enclave.

[8] KeyInfo.isInsideSecurityHardware(): https://developer.android.com/reference/android/security/keystore/KeyInfo#isInsideSecureHardware()



Figur 2: Arkitekturen til et maskinvarestøttet nøkkellager Illustrasjon av Android Open Source Project, utgitt under CC BY 3.0 lisensen. Endret til vektorgrafikk.

ARM TrustZone

TrustZone er en funksjonalitet som kan bygges inn i en ARM-chip for å legge inn maskinvarestøtte for to ulike miljøer. TrustZone er laget som et økonomisk alternativ til å ha en brikke med egen CPU, minne og lagring der TEE kan kjøre. Ettersom TrustZone sørger for god sikkerhet samtidig som det er et veldig økonomisk alternativ, er det ikke overraskende at mange produsenter benytter seg av denne teknologien [173, 174].

I praksis implementeres de ulike miljøene ved hjelp av to flagg (Non-Secure / NS-bits) på systembussen og beskyttede områder av minnet [175]. For mer informasjon om hvordan TrustZone er implementert, anbefaler vi å lese ARM sin egen spesifikasjon[9].

Man kan se på det som en form for maskinvarestøttet viritualisering. Man får ett usikkert miljø der det normale operativsystemet kjører, og et sikkert miljø som kalles TEE (Trusted Execution Enviroment) der sensitive data lagres og behandles [176, 177]. De to miljøene har en begrenset API for å kommunisere med hverandre, og det sikre operativsystemet som kjører i TEE har i de fleste tilfeller svært begrenset med funksjonalitet. Fordi man bygger dette miljøet med begrenset funksjonalitet og med sikkerhet i fokus, er det som oftest veldig sikkert og kan beholde konfidensialiteten og integriteten til data, selv om en angriper skulle få full kontroll over det usikre miljøet [172].

Av alle angrepene på TEE som vi har sett, har alle gått ut på å utnytte sårbarheter i API-en mellom miljøene, og ikke implementasjonen av TrustZone i seg selv.

iOS

Apple har utviklet noe de kaller “Secure Enclave”. Dette er en patentert løsning for et sikret miljø og som finnes på alle av Apple sine systembrikker fra og med A7, som ble brukt i iPhone 5S [105, side 7]. I praksis benyttes ARM’s TrustZone/SecurCore-teknologi for å opprette de ulike miljøene. Secure Enclave kan trolig antas å være litt sikrere ettersom det benyttes en egen prosessor med beskyttelse mot tukling, men denne må også ha en definert API som ikke nødvendigvis er helt sikker [178]. Mye tyder på at det er mulig å komme seg forbi brute-force beskyttelsen og dermed enkelt låse opp telefoner med kort låseskjerm-kode [179, 180].

[9] ARM Security Technology: Building a Secure System using TrustZone Technology: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

ARM SecurCore

ARM SecurCore er prosessortype som er utviklet for å være ekstra vanskelig å fysisk tukle med (Tamper resistant). Ved å legge nøkkellageret på en egen prosessor med beskyttelse mot tukling, får man beskyttelse mot at noen fysisk tukler med den mobile enheten i tillegg til beskyttelsen mot angrep via programvaren. Dersom noen skulle få fysisk tilgang til en mobil enhet, vil det kreve høy kompetanse for å kunne modifisere maskinvaren til å avsløre nøklene som ligger lagret i nøkkellageret, men ved å benytte seg av SecurCore heves dette kompetansekravet enda mer. Merk at ARM beskriver at SecurCore-prosessorene har beskyttelse mot tukling (tamper resistance), og ikke at de er helt beskyttet mot tukling (tamper proof) [181].

Er nøkler i et maskinvarestøttet nøkkellager altså helt sikre?

Det usikre og det sikre miljøet må ha en API for å kommunisere med hverandre slik at det sikre miljøet kan utføre sensitive operasjoner for det usikre miljøet. Dersom denne API-en eller operativsystemet som kjører i det sikre miljøet ikke implementeres korrekt, vil en angriper kunne utnytte eventuelle sårbarheter til å ta kontroll over det sikre miljøet også. Maskinvarestøtten sørger kun for at den eneste måten en angriper kan kommunisere med det sikre miljøet er via den definerte API-en.

En angriper med kontroll over det usikrede miljøet kan styre programflyten

Når man genererer en nøkkel som lagres i et nøkkellager, kan man bestemme en rekke egenskaper for denne nøkkelen. Dersom man ikke bestemmer noen spesielle egenskaper, vil nøkkelen kun sikres ved at den ikke kan hentes ut fra det sikre miljøet. En angriper med full kontroll over det usikrede miljøet kan allikevel bestemme oppførselen til det usikrede miljøet og dermed be nøkkellageret om å utføre en hvilken som helst ønsket kryptografisk operasjon med nøkkelen.

Et slikt angrep kan unngås ved å bestemme et sett med egenskaper i det nøkkelen genereres. Som tidligere beskrevet kan nøkler beskyttes av et bruker-angitt passord eller kreve at brukeren autentiserer seg med fingeravtrykk.

Sårbarheter kan kompromittere nøklene

Mange TEE-implementasjoner har mangelfull beskyttelse, og ettersom angrepsoverflaten stadig økes ved at det legges til flere tjenester som kjøres i TEE (for eksempel DRM), er det nok lenge til man kan regne maskinvarestøttede nøkkellagre som helt sikre. Både TrustZone og Secure Enclave sørger kun for ett sikret miljø, og nøkkellageret må derfor kjøre i samme miljø som andre tjenester som krever ekstra sikkerhet. Man kan allikevel anta at nøkler i et maskinvarestøttet nøkkellager er mye bedre sikret enn nøkler som kun beskyttes av programvaren. For å utnytte sårbarheter i et TEE kreves det som regel at man først har eskalert privilegiene sine i det usikre miljøet. Sannsynligheten for at en angriper finner og klarer å utnytte sårbarheter i begge disse miljøene er liten, men det er mulig [182, 174, 183, 178].

4.5.4 Sammenligning av ulike metoder for å lagre kryptografiske nøkler

I tabell 3 sammenlignes de ulike metodene for å lagre kryptografiske nøkler. Tabellen lister ut de forskjellige metodene og hvor godt de fullfører kriteriene:

- Beskyttelse mot å hentes ut av en angriper med lese-rettigheter til filsystem

Tabell 3: Sammenligning av ulike metoder for å lagre kryptografiske nøkler

	Kriterier				
Metode		Beskyttelse mot å hentes ut av en angriper med lese-rettigheter til filsystem	Beskyttelse mot å hentes ut av en angriper med root-privilegier	Beskyttelse mot at en angriper med root-privilegier bruker nøkkelen til kryptografiske operasjoner	Bruker-vennlighet
	Klartekst i applikasjonsmappen	Svak	Svak	Svak	God
	Kryptert hardkodet nøkkel og lagret i applikasjonsmappen	Svak	Svak	Svak	God
	Kryptert med bruker-angitt passord	Sterk[10]	Svak	Svak	Dårlig
	I nøkkellager uten maskinvarestøtte	Svak	Svak	Svak	God
	I nøkkellager uten maskinvarestøtte, med bruker-angitt passord	Sterk[10]	Svak	Svak	Dårlig
	I nøkkellager med maskinvarestøtte	Sterk	Sterk	Svak	God
	I nøkkellager med maskinvarestøtte, med bruker-angitt passord	Sterk	Sterk	Svak	Dårlig
	I nøkkellager med maskinvarestøtte, med krav om at brukeren autentiserer seg med biometri for hver kryptografiske operasjon	Sterk	Sterk	Sterk[11]	Medium

- Beskyttelse mot å hentes ut av en angriper med root-privilegier
- Beskyttelse mot at en angriper med root-privilegier bruker nøkkelen til kryptografiske operasjoner
- Brukervennlighet

I tabellen kan man se at “I nøkkellager med maskinvarestøtte, med krav om at brukeren autentiserer seg med biometri for hver kryptografiske operasjon” kommer best ut, med tre “sterke” og en “medium”.

[10] Avhenger av entropien til passordet
[11] Avhenger av sikkerheten til den biometriske autentiseringsmetoden

5 Konseptbevis for biometrisk autentisering mot et serversystem

En del av oppgaven fra oppdragsgiver gikk ut på å lage en applikasjon som benytter seg av sterk autentisering opp mot et serversystem. Både Google[1] og Apple[2] har laget eksempel-applikasjoner som autentiserer brukeren med fingeravtrykk, men ingen av eksemplene autentiserer brukeren opp mot et ekte serversystem. Eksemplene opprettet heller ikke autentiserte økter, så dette måtte vi designe og implementere helt fra bunnen av. Vi har derfor hentet litt inspirasjon fra Google sitt eksempel, men har måttet utvikle det meste selv. En del av utviklingsprosessen var å finne alle truslene, så vi utførte trusselmodellering av applikasjonen sammen med utviklingen. Resultatet av dette vises i seksjon 5.6.

5.1 Krav og Design

For å lage en applikasjon som implementerer sterk autentisering mot et serversystem, måtte vi først kartlegge og designe hvordan man implementerer dette. Siden applikasjonen er ment som et konseptbevis, er designet minst like sentralt som selve kildekoden fordi designet kan benyttes til å implementere tilsvarende funksjonalitet på andre plattformer.

5.1.1 Kravspesifikasjon

Som en del av kravspesifikasjonene har vi definert hva applikasjonen skal gjøre og hvordan den skal fungere. Med tanke på at applikasjonen kun skal fungere som et konseptbevis for sterk autentisering med fingeravtrykk og sikkert lagrede nøkler mot et serversystem, har applikasjonen ingen annen funksjonalitet.

Brukstilfeller

Basert på oppgaven fra oppdragsgiver har vi identifisert brukstilfellene som er illustrert i figur 3. Brukstilfellene er beskrevet i detalj i tabell 4, 5 og 6. Vi har også tatt i bruk misbrukstilfeller, som er illustrert i figur 12.

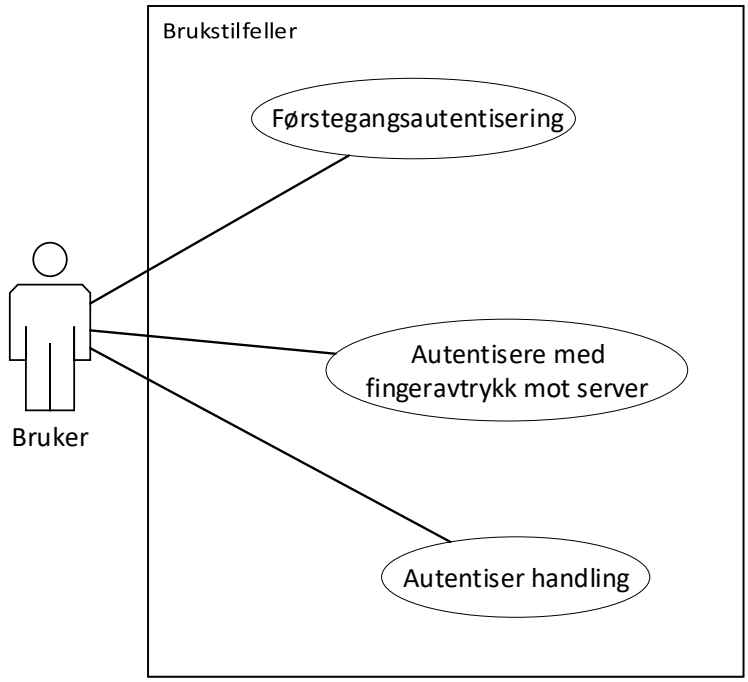
Hvordan er kravene utarbeidet?

De funksjonelle- og operasjonelle kravene er utarbeidet i samarbeid med Eika Gruppen. Sikkerhetskravene har vi selv lagt til for å gi noen krav til hvor sikker autentiseringsmetoden skal være.

- Funksjonelle krav
    - Applikasjonen skal fungere som et konseptbevis for sikker autentisering opp mot et serversystem.

[1] Android AsymmetricFingerprintDialog Sample: https://github.com/googlesamples/android-AsymmetricFingerprintDialog/

[2] KeychainTouchID: Using Touch ID with Keychain and LocalAuthentication: https://developer.apple.com/library/content/samplecode/KeychainTouchID/



Figur 3: Brukstilfeller

Tabell 4: Brukstilfelle for "førstegangsautentisering"

Brukstilfelle:	Førstegangsautentisering
Aktør:	Bruker
Mål:	Autentisere seg for første gang på en enhet
Beskrivelse	En bruker skal autentisere seg for førte gang på en enhet, og enheten knyttes opp mot brukerkontoen til brukeren.
Pre betingelser	Enheten er ikke knyttet opp mot noen bruker. Enheten støtter fingeravtrykkautentisering. Brukeren må ha lagt inn et fingeravtrykk på enheten
Post betingelser	Enheten skal være knyttet opp mot brukeren sin konto, og brukeren skal kunne autentisere seg med fingeravtrykk i fremtiden. Brukeren skal få en autentisert økt og sendes til skjermen der man kan gjennomføre handlinger.
Feilsituasjoner	Dersom enheten ikke støtter fingeravtrykkautentisering, vises en feilmelding. Dersom brukeren ikke har lagt til et fingeravtrykk på enheten, blir brukeren bedt om å legge til et fingeravtrykk. Dersom brukeren mislykkes i å autentisere seg, blir brukeren bedt om å prøve på nytt.

Tabell 5: Brukstilfelle for “autentiser med fingeravtrykk mot server”

Brukstilfelle:	Autentiser med fingeravtrykk mot server
Aktør:	Bruker
Mål:	Det skal opprettes en autentisert økt.
Beskrivelse	Brukeren autentiserer seg med fingeravtrykk og oppretter en økt.
Pre betingelser	Brukeren må ha gjennomført førstegangsautentisering på enheten. Brukeren må ikke ha slettet alle fingeravtrykkene på enheten.
Post betingelser	Brukeren skal få en autentisert økt og sendes til skjermen der man kan gjennomføre handlinger.
Feilsituasjoner	Dersom brukeren har slettet alle fingeravtrykkene på enheten, blir brukeren bedt om å legge til et fingeravtrykk og må gjennomføre førstegangsautentisering på nytt. Dersom brukeren mislykkes i å autentisere seg, blir brukeren bedt om å prøve på nytt.

Tabell 6: Brukstilfelle for “autentiser handling”

Brukstilfelle:	Autentiser handling
Aktør:	Bruker
Mål:	Gjennomføre en handling
Beskrivelse	Brukeren autentiserer en handling ved hjelp av den autentiserte økten.
Pre betingelser	Brukeren har opprettet en økt.
Post betingelser	Brukeren skal få beskjed om at handlingen har blitt gjennomført.
Feilsituasjoner	Dersom handlingen ikke kunne gjennomføres, vil en feilmelding vises til brukeren.

    - Applikasjonen skal bruke fingeravtrykk og sikkert lagrede nøkler for å autentisere brukeren opp mot en server.
    - Når brukeren autentiserer seg med fingeravtrykk, skal det opprettes en økt slik at brukeren ikke må autentisere hvert kall til serveren.

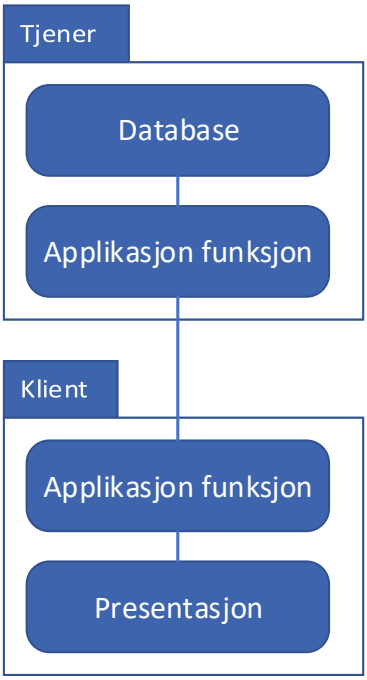
- Operasjonelle krav
    - Applikasjonen skal utvikles for Android
    - Applikasjonen trenger ikke å fungere på eldre Android-versjoner.
- Sikkerhetskrav
    - Nøkler som brukes til å autentisere brukeren skal være svært vanskelig å få tak i for eventuelle angripere - inkludert dersom deler av programvaren på telefonen er sårbar eller angriperen har fysisk tilgang til enheten.
    - Applikasjonen skal sørge for at brukere ikke kan benekte å ha gjennomført handlinger (non-repudiation).
    - Serveren skal kreve at brukeren autentiserer alle kall til serveren.
    - Kommunikasjon mellom applikasjonen og serveren skal være kryptert.
    - Applikasjonen skal i størst mulig grad følge aktuelle standarder for sterk autentisering

5.1.2 Arkitektur

Ved å benytte oss av en lagdelt arkitektur blir det enklere å lage komponenter som følger prinsippene om høy styrke og lave koblinger, som igjen gjør det enklere med gjenbruk og vedlikehold av kode, men med tanke på at oppgaven går ut på å utvikle en applikasjon som autentiserer brukere ovenfor et serversystem, var det også naturlig å velge klient-tjener-modellen som arkitekturmønster.

Vi har derfor valgt en lagdelt klient-tjener-modell som kalles "distribuert funksjon" (figur 4), der klienten tar seg av en stor del av prosesseringen, men kun serveren har tilgang til data. Serveren stoler heller ikke på applikasjonen og brukeren må autentisere seg for å få tilgang til data på serveren.

En nevneverdig ulempe med klient-tjener-modellen, er at modellen er ekstra sårbar ovenfor tjenestenektangrep, siden man har ett enkelt punkt som alle klientene er avhengige av at er tilgjengelig.



Figur 4: Illustrasjon av "distribuert funksjon"

5.1.3 Standarder for sterk autentisering

Siden ett av sikkerhetskravene vi satte opp var at applikasjonen i størst mulig grad skulle følge aktuelle standarder for sterk autentisering, måtte vi først identifisere hvilke standarder som gjelder.

Definisjon av "sterk autentisering"

Oppgavebeskrivelsen nevner at applikasjonen skal benytte «sterk autentisering mot et serversystem». Vi har basert oss på EU-byrået European Banking Association (EBA) sin definisjon av «sterk autentisering». EBA definerer sterk bruker-autentisering på følgende

måte [184, side 11]:

"a procedure based on the use of two or more of the following elements – categorised as knowledge, ownership and inherence: i) something only the user knows, e.g. static password, code, personal identification number; ii) something only the user possesses, e.g. token, smart card, mobile phone; iii) something the user is, e.g. biometric characteristic, such as a fingerprint. In addition, the elements selected must be mutually independent, i.e. the breach of one does not compromise the other(s). At least one of the elements should be non-reusable and non-replicable (except for inherence), and not capable of being surreptitiously stolen via the internet. The strong authentication procedure should be designed in such a way as to protect the confidentiality of the authentication data."

Ved bruk av nøkler som er lagret i et maskinvarestøttet nøkkellager og fingeravtrykk som brukes til å låse opp denne nøkkelen, vil autentiseringen være definert som «sterk autentisering» fordi det kreves at man både har den mobile enheten til brukeren og fingeravtrykket til brukeren for å autentisere seg.

FIDO

FIDO er en organisasjon som arbeider for å lage et standardisert økosystem for autentisering som skal sikre brukervennlighet og sikkerhet. FIDO alliansen har over 250 medlemmer, som inkluderer mange av verdens fremste IT-selskaper [185, 186].

FIDO UAF (FIDO Universal Authentication Framework) er et rammeverk for sterk autentisering som benytter seg av nøkkelpar og signering av utfordringer for å autentisere brukeren. FIDO alliansen har publisert et dokument som beskriver arkitekturen til UAF[3], og følgende illustrasjoner hentet fra arkitektur-oversikten gir en god oversikt over hvor brukeren registreres (figur 5) og autentiseres (figur 6).

Registrering av applikasjon: Det opprettes et nytt nøkkelpar som er spesifikt for brukeren og den gjeldende applikasjonen. Den offentlige nøkkelen til brukeren overføres til autentiseringsserveren og knyttes til brukerens konto.

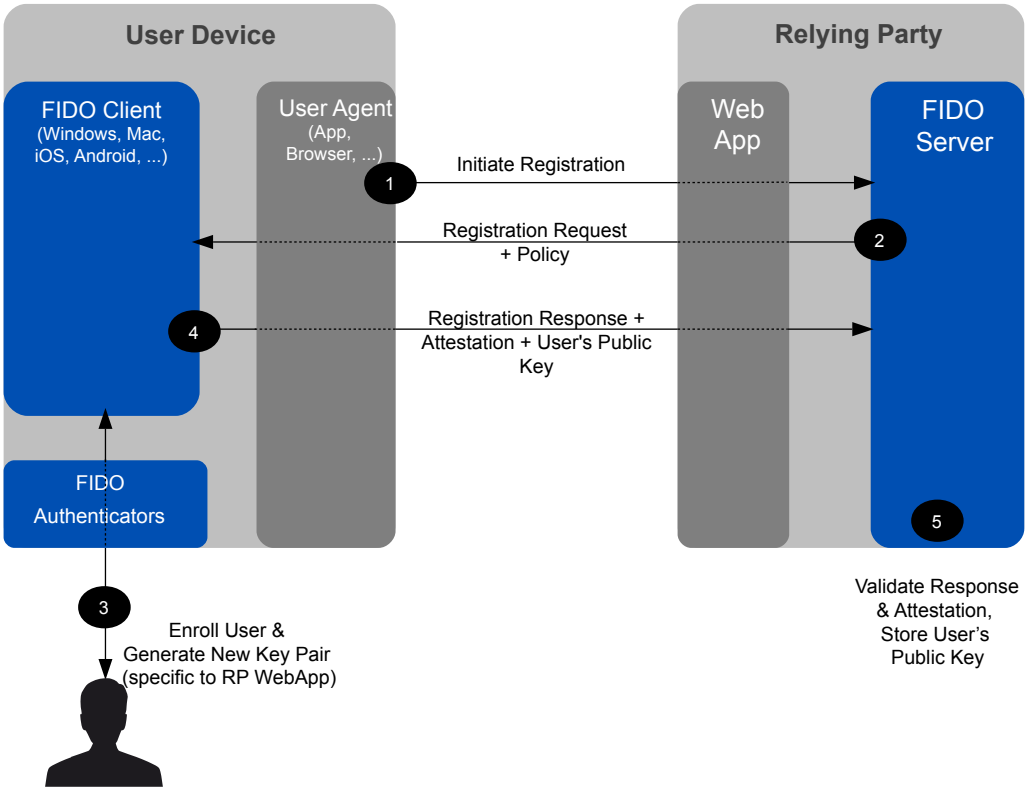
Autentisering av bruker: Merk at denne autentiseringsmetoden autentiserer kun ett enkelt kall. Applikasjonen starter autentiseringen og mottar en utfordring fra serveren. Brukeren må deretter autentisere seg ovenfor den mobile enheten sin for å låse opp den private nøkkelen som brukes til å signere utfordringen og deretter sende denne signerte utfordringen til serveren som verifiserer utfordringen med brukerens offentlige nøkkel.

Hvordan brukeren autentiserer seg ovenfor den mobile enheten sin, er ikke en del av standarden, men noen mulige metoder er biometrisk autentisering eller en pin-kode.

5.1.4 Autentisering for iOS

Det er verdt å merke seg at man ikke kan kreve at brukeren autentiserer seg med fingeravtrykk for å låse opp en nøkkel lagret i maskinvarestøttet nøkkellager på iPhone (avviklet i iOS 11.3). Med iOS kan man derimot kreve at nøkkelen låses opp ved at brukeren autentiserer seg med enten fingeravtrykk eller skjermlås-koden til enheten. Angrepsoverflaten vil bli litt større siden en eventuell angriper kun trenger å få tak i enten skjermlås-koden eller fingeravtrykket til brukeren, men autentiseringen regnes uansett som sterk autentisering fordi man minst må ha noe brukeren har og noe brukeren vet eller fingeravtrykket til brukeren.

[3] UAF arkitekturen https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html#fig-fido-uaf-high-level-architecture



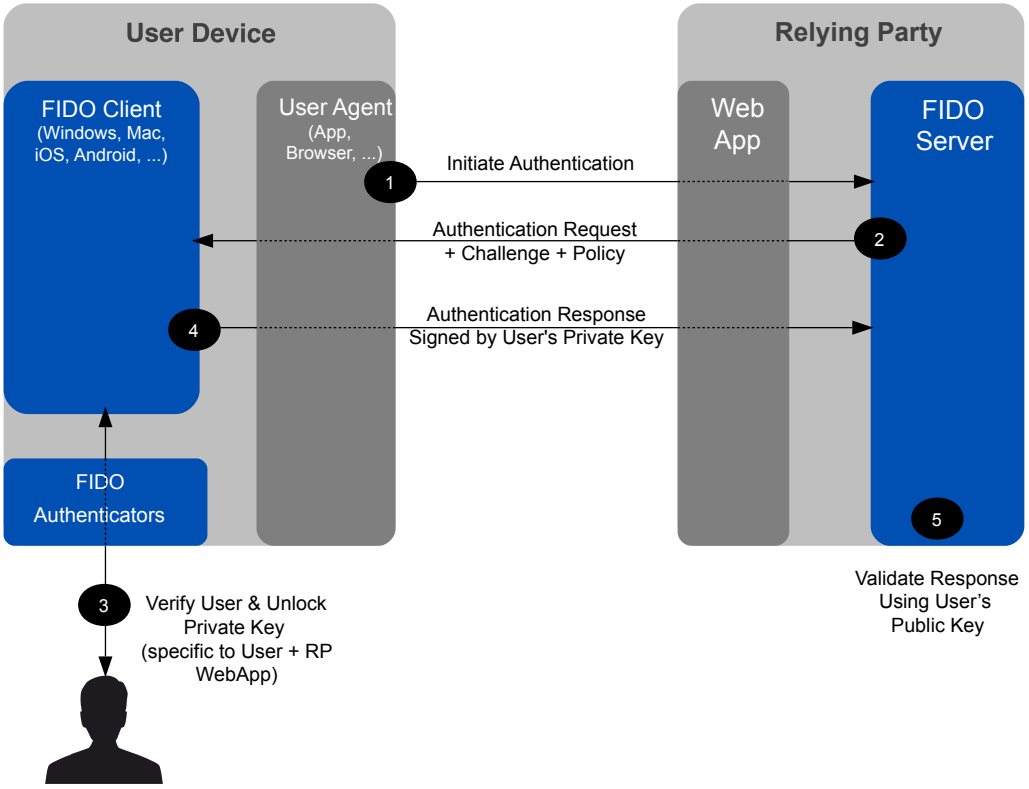
Figur 5: FIDO UAF registrering [3]. Endret font og nedskalert fra original.

iPhone har også mindre funksjonalitet når det kommer til typen nøkler som kan lagres i det maskinvarestøttede nøkkellageret. iPhone kan kun lagre private nøkler for nøkkelpar i nøkkellageret [170]. Av den grunn er det mer praktisk å benytte asymmetriske nøkler på Android også, slik at man ikke trenger å benytte seg av ulike signerings- og verifiseringsmetoder avhengig av hva slags enhet brukeren har. Ved bruk av symmetriske nøkler ville også angrepsoverflaten blitt større fordi den hemmelige nøkkelen hadde vært lagret både på den mobile enheten og på serveren.

5.1.5 Autentiseringsmetode for autentiserte økter

Etter å ha gått gjennom FIDO UAF og lest en sikkerhetsanalyse av protokollen, har vi kommet frem til at en liknende autentiseringsmetode vil være ideell for å opprette autentiserte økter. Analysen konkluderer med at det er mulig for en angriper å autentisere seg som en annen bruker, men dette forutsetter at angriperen har mulighet til å endre på programvaren som kjører på brukerens enhet [187]. Vi har ikke adoptert UAF-protokollen direkte, da dette fort ville vært for stort omfang for denne oppgaven, og det eksisterer noen utfordringer for å implementere denne på vanlige mobile enheter [188]. Få mobile enheter har maskinvarestøtte for nøkkel-attestering [189], så man får ikke benyttet hele UAF-protokollen på de fleste enhetene heller. En av Qualcomm sine nyeste fingeravtrykklesere til mobile enheter, “Snapdragon Sense ID”, har støtte for FIDO UAF, men krever en API fra operativsystemet for å kunne benyttes [190].

Fordi oppdragsgiver ønsket at vi skulle lage et konseptbevis som opprettet autentiserte økter, ble første mål å designe en autentiseringsmetode som benytter seg av private



Figur 6: FIDO UAF autentisering [3]. Endret font og nedskalert fra original.

nøkler som lagres i et maskinvarestøttet nøkkellager og i tillegg bruker fingeravtrykk-autentisering for å opprette autentiserte økter.

Normalt vil økter håndteres ved hjelp av en type øktnøkkel/token som serveren utsteder for brukeren, og som brukeren autentiserer seg med resten av økten. Ulempen med dette er at en slik token kun kan lagres i applikasjonens data-mappe. Dersom den mobile enheten skulle bli infisert av ondsinnet programvare som oppnår privilegier som tillater at programvaren kan hente ut data fra andre applikasjoners mapper, vil denne øktnøkkelen kunne hentes ut og brukes av angriperen.

Vi benytter oss derfor av et eget øktnøkkelpar som genereres og lagres i det maskinvarestøttede nøkkellageret til enheten, på samme måte som det vanlige nøkkelparet som brukeren normalt ville brukt til å autentisere ett enkelt kall (Som i FIDO sin illustrasjon). De eneste forskjellene mellom det opprinnelige nøkkelparet og øktnøkkelparet, er at den private øktnøkkelen har mye kortere levetid, men til gjengjeld krever den ikke at brukeren autentiserer seg for å låse opp nøkkelen.

Noe annet som skiller seg fra FIDO sin registrering, er at vi benytter oss av en ekstra autentiseringsmetode. Denne ekstra autentiseringsmetoden fungerer som «hovednøkkel»/master key, og lar brukeren knytte en mobil enhet opp mot brukerkontoen sin på serveren. Grunnen til dette er at den private nøkkelen som lagres i nøkkellageret vil slettes dersom brukeren fjerner skjermlåsen på enheten sin, og dersom denne nøkkelen var det eneste som brukeren kunne autentisere seg med, ville dette ført til at brukeren aldri kunne gjenopprettet kontoen sin (uten en eller annen form for “bakdør”).

Denne autentiseringsmetoden må også være sterk ettersom den vil fungere som hovednøkkel. I vår applikasjon har vi kun brukt grunnleggende autentisering fordi denne autentiseringen er utenfor omfanget av oppgaven.

I sekvensdiagrammet for "førstegangsautentisering" (figur 7) ser man hvordan man autentiserer en bruker for første gang på en mobil enhet. Det første som gjøres er at man genererer to nøkkelpar. Ett permanent nøkkelpar og ett nøkkelpar som kun skal gjelde for en økt. Deretter signerer man den offentlige øktnøkkelen, men for å kunne signere noe med den private nøkkelen må man autentisere brukeren med fingeravtrykk. Ved å autentisere brukeren med fingeravtrykk i dette skrittet, får man både signert den offentlige øktnøkkelen og bekreftet at den aktive brukeren av enheten også er personen som har lagt inn fingeravtrykket sitt.

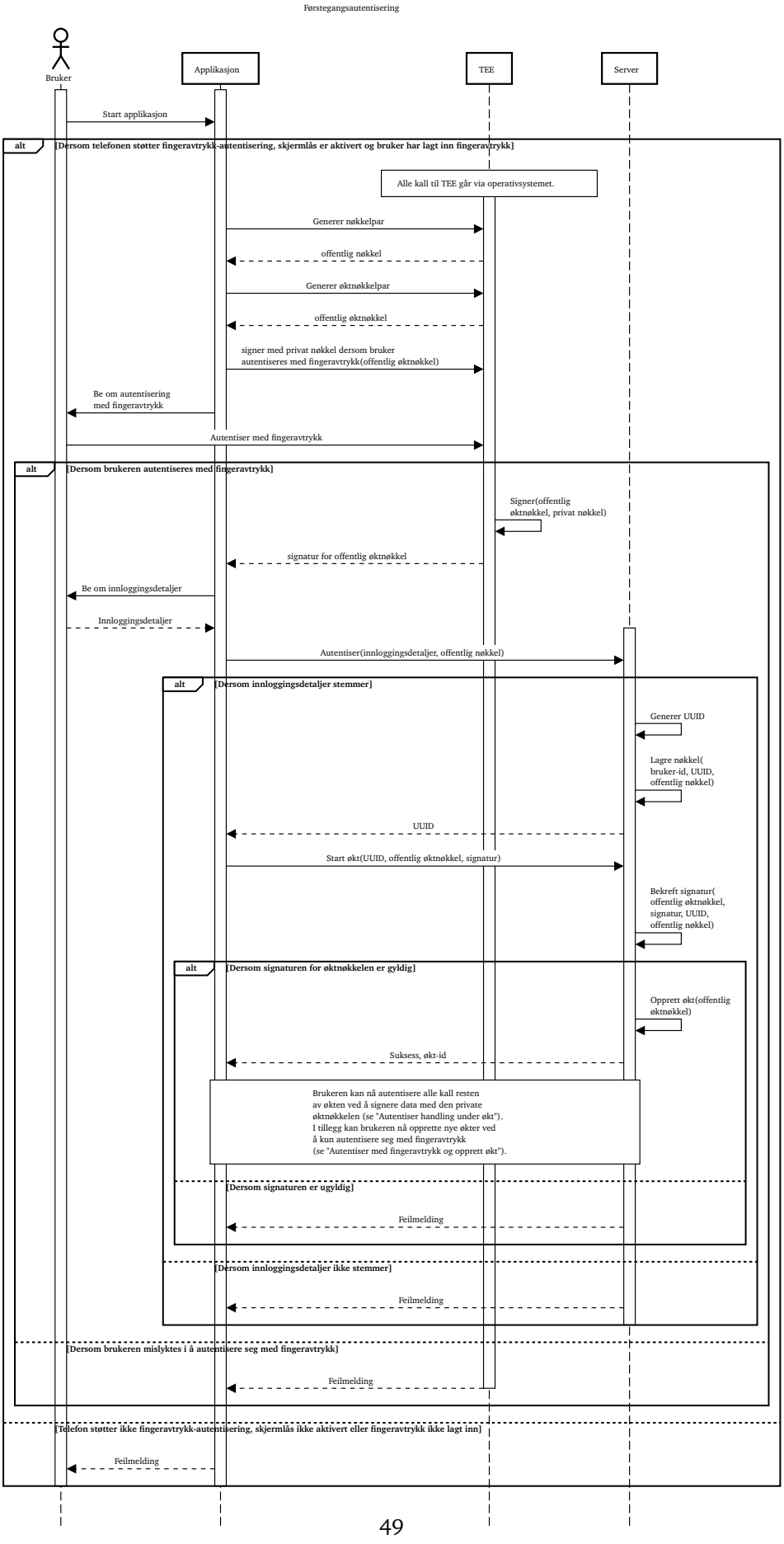
Etter å ha generert to nøkkelpar og signert den offentlige øktnøkkelen, må brukeren autentisere seg ovenfor serveren på en eller annen metode. Sammen med innloggingsdetaljene som skal autentisere brukeren sendes den permanente offentlige nøkkelen. Dersom innloggingsdetaljene stemmer, lagres den offentlige nøkkelen på serveren og knyttes opp mot brukeren som autentiserte seg.

Serveren genererer også en unik id (UUID) for nøkkelen som sendes til klienten. Denne unike id-en vil applikasjonen bruke hver gang den vil angi hvilken nøkkel som serveren skal benytte for å verifisere signaturen. Det er benyttet UUID for hver nøkkel slik at en bruker kan knyttes opp mot flere nøkler dersom man skulle ha flere mobile enheter. Det er også en fordel å benytte seg av UUID fremfor et tall som inkrementerer med en verdi for hver gang man legger til en ny, fordi man unngår å avsløre antall brukere av tjenesten, og det gjør det vanskeligere for en eventuell angriper å gjennomføre automatiske angrep mot samtlige brukere.

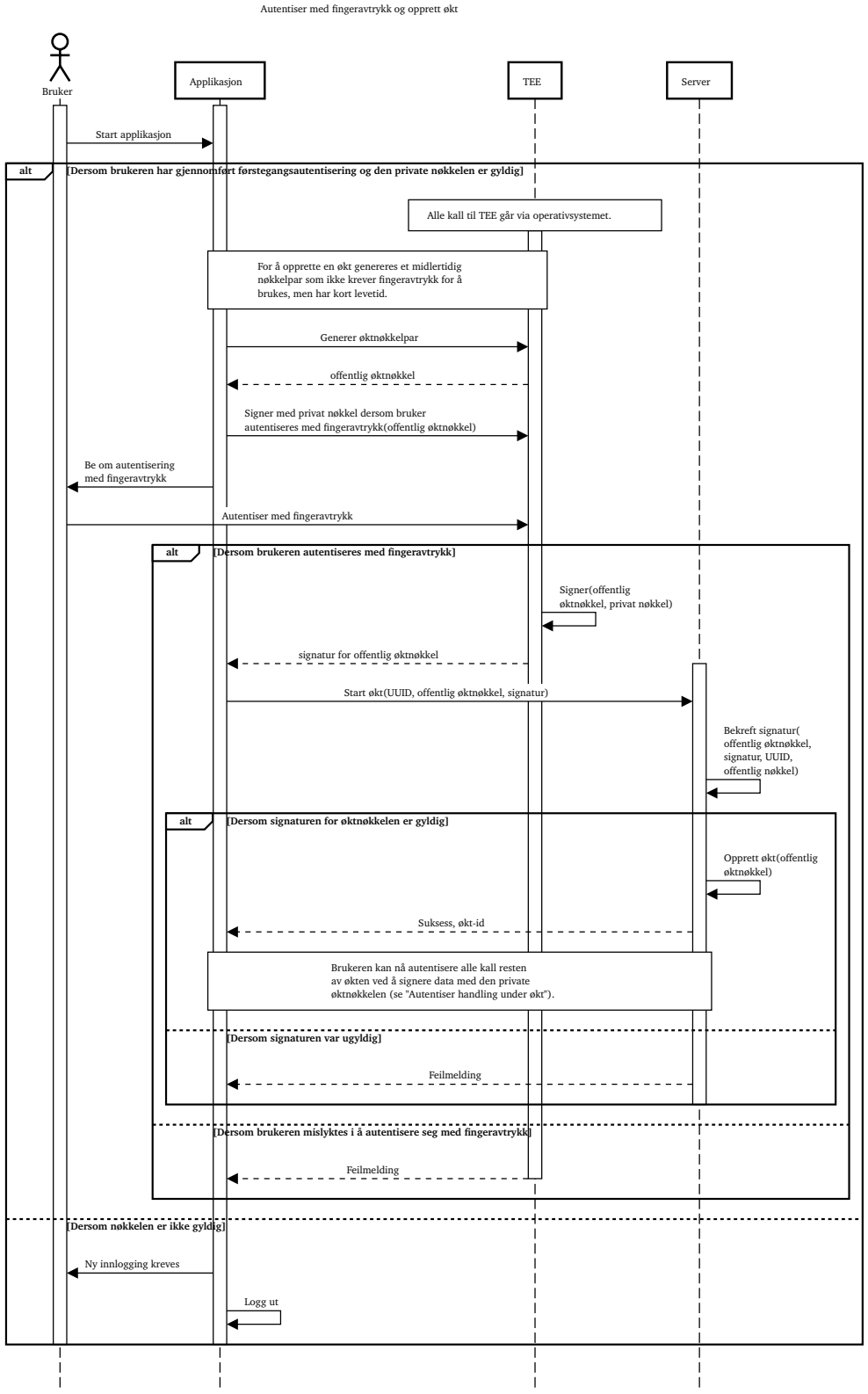
Etter at applikasjonen har mottatt en UUID for nøkkelen, lagres dette unna. Applikasjonen sender deretter en forespørsel om å starte en økt til serveren. Det som sendes med som parametere til kallet er UUID som identifiserer nøkkelen, den offentlige øktnøkkelen og signaturen av den offentlige øktnøkkelen, som har blitt gjennomført med den permanente private nøkkelen. Serveren kan deretter verifisere signaturen av den offentlige øktnøkkelen ved å benytte seg av den permanente offentlige nøkkelen. Dersom signaturen stemmer vet serveren at nøkkelen kommer fra den samme enheten som brukeren autentiserte seg med innloggingsdetaljene fra. Serveren knytter deretter den offentlige øktnøkkelen opp mot UUIDen til den permanente offentlige nøkkelen og angir et utløpstidspunkt i nær fremtid. En økt-id returneres til applikasjonen slik at den kan angi hvilken UUID og økt hvert kall gjelder for.

Etter at brukeren har gjennomført førstegangsautentisering har man en autentisert økt slik at brukeren kan gjennomføre handlinger og autentisere hvert kall. Dersom man har gjennomført førstegangsautentisering på enheten tidligere, trenger man kun å generere et nytt øktnøkkelpar og signere dette på tilsvarende måte som i førstegangsautentiseringen. Den eneste forskjellen er at man ikke trenger å generere et permanent nøkkelpar eller å angi innloggingsdetaljene (hovednøkkelen). Denne fremgangsmåten er illustrert i sekvensdiagrammet "Autentiser med fingeravtrykk og opprett økt" (figur 8).

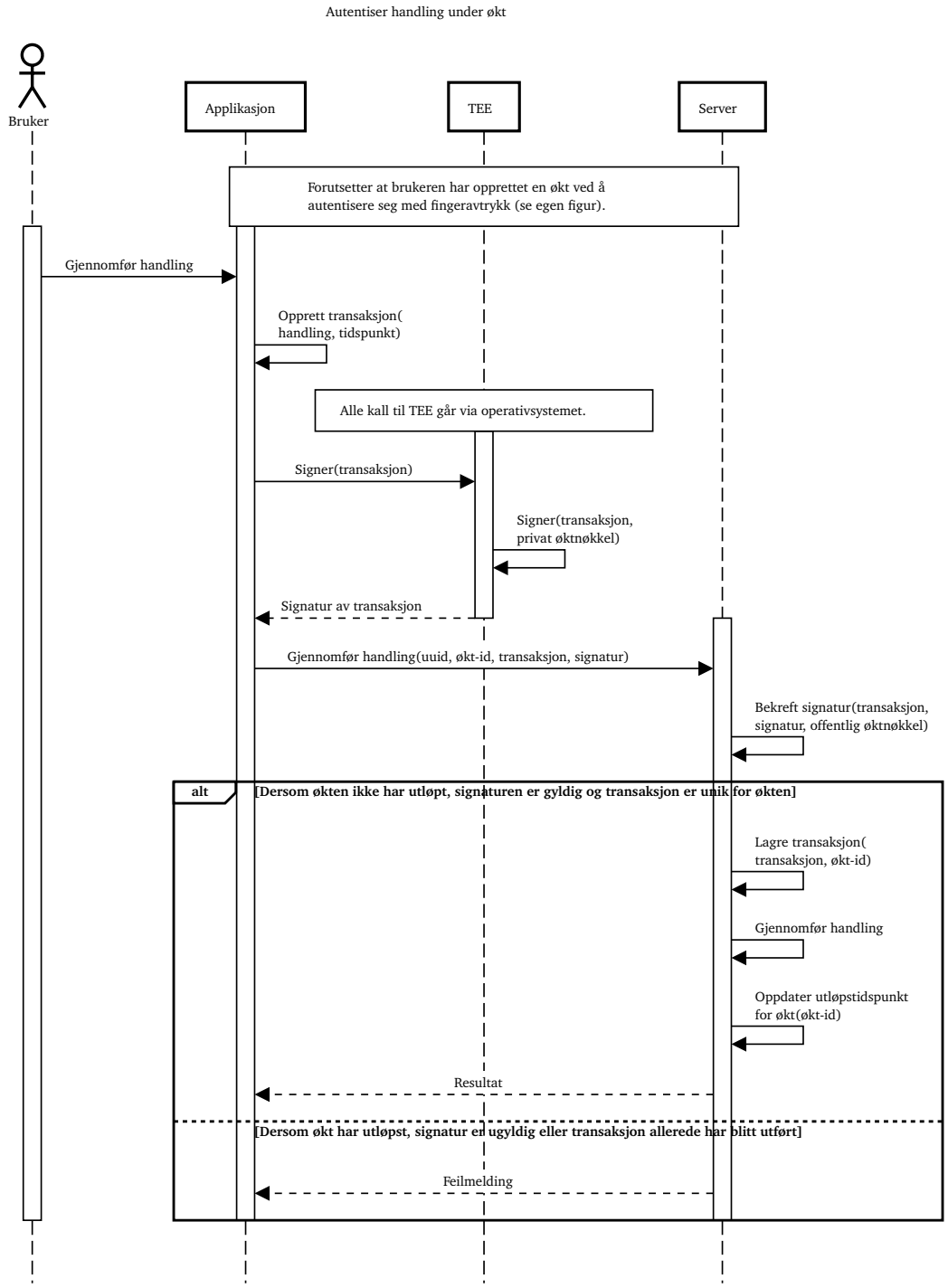
Sekvensdiagrammet "Autentiser handling under økt" (figur 9) viser hvordan en bruker kan gjennomføre en handling under en økt og autentisere denne handlingen ovenfor serveren. Dette forutsetter selvfølgelig at man har en aktiv økt som ble opprettet som vist



Figur 7: Et sekvensdiagram som illustrerer hvordan førstegangsautentiseringen fungerer



Figur 8: Et sekvensdiagram som illustrerer hvordan autentisere med fingeravtrykk og opprette en økt



Figur 9: Et sekvensdiagram som illustrerer hvordan autentisere en handling under en økt

i tidligere sekvensdiagrammer.

Første handlingen applikasjonen utfører er å opprette en transaksjon, et objekt som består av en handling og et tidspunkt. Denne transaksjonen signeres deretter med den private øktnøkkelen og transaksjonen sendes til serveren sammen med signaturen. Applikasjonen må også sende med UUID og økt-id for at serveren skal vite hvilket nøkkelpar den skal bruke til å verifisere signaturen. Når transaksjonen kommer frem til serveren, må serveren sjekke tre ting; Økten må ikke ha utløpt, signaturen må være gyldig og transaksjonen må ikke ha forekommet tidligere i økten. Sistnevnte sjekk er for å hindre replay-angrep. Dersom disse tre tingene var gyldige, kan serveren lagre unna transaksjonen og gjennomføre handlingen. For å kunne ansvarliggjøre brukeren for sine handlinger, må alle transaksjoner og signaturer for handlinger som brukeren kan finne på å benekte, lagres også etter at økten har utløpt.

I tillegg til disse sekvensdiagrammene som illustrerer autentiseringen i applikasjonen, har vi laget noen alternative sekvensdiagrammer (Figur 10) som viser hvordan man kan autentisere brukeren ovenfor en server uten at man oppretter en økt. Denne modellen kan utvides til å autentisere brukeren og opprette økt ved å utstede en vanlig øktnøkkel (for eksempel SAML token eller Json Web Token). Dette vil være litt mindre sikkert enn bruk at nøkler lagret i et maskinvarestøttet nøkkellager, men vil være bedre egnet til SSO. Det siste sekvensdiagrammet (Figur 11) illustrerer hvordan man kan autentisere en enkelttransaksjon ved hjelp av fingeravtrykk. Dette er metoden som ligner mest på FIDO UAF-protokollen.

5.1.6 Sikkerhet ved autentiseringsmetoden

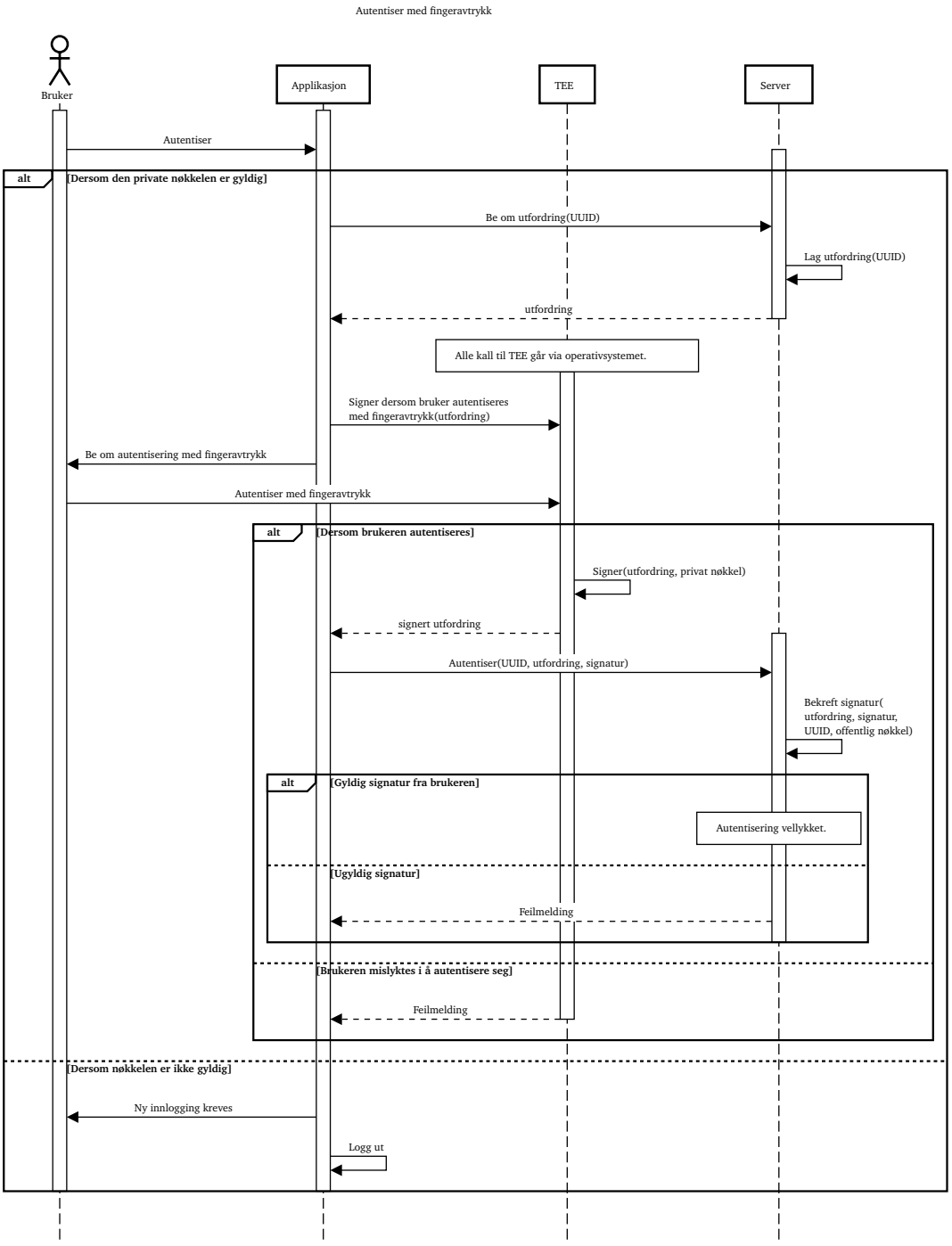
Ved bruk av asymmetriske nøkler som signerer data, vil autentiseringsmetoden sørge for integritet (data har ikke blitt modifisert etter at signaturen ble generert) og autentisering av brukeren (data er sendt av brukeren som besitter den private nøkkelen).

Etter en grundig analyse har vi derimot kommet frem til at det er én nøkkel som vi ikke bekrefter integriteten til i applikasjonslaget: Den permanente offentlige nøkkelen. Dersom man ønsker å bekrefte integriteten til denne på applikasjonslaget, kan man legge til en nøkkelutveksling på applikasjonslaget. Den symmetriske nøkkelen fra denne utvekslingen kan brukes til å generere en HMAC av den offentlige nøkkelen. Denne symmetriske nøkkelen kan også brukes til å kryptere innloggingsdetaljene til brukeren.

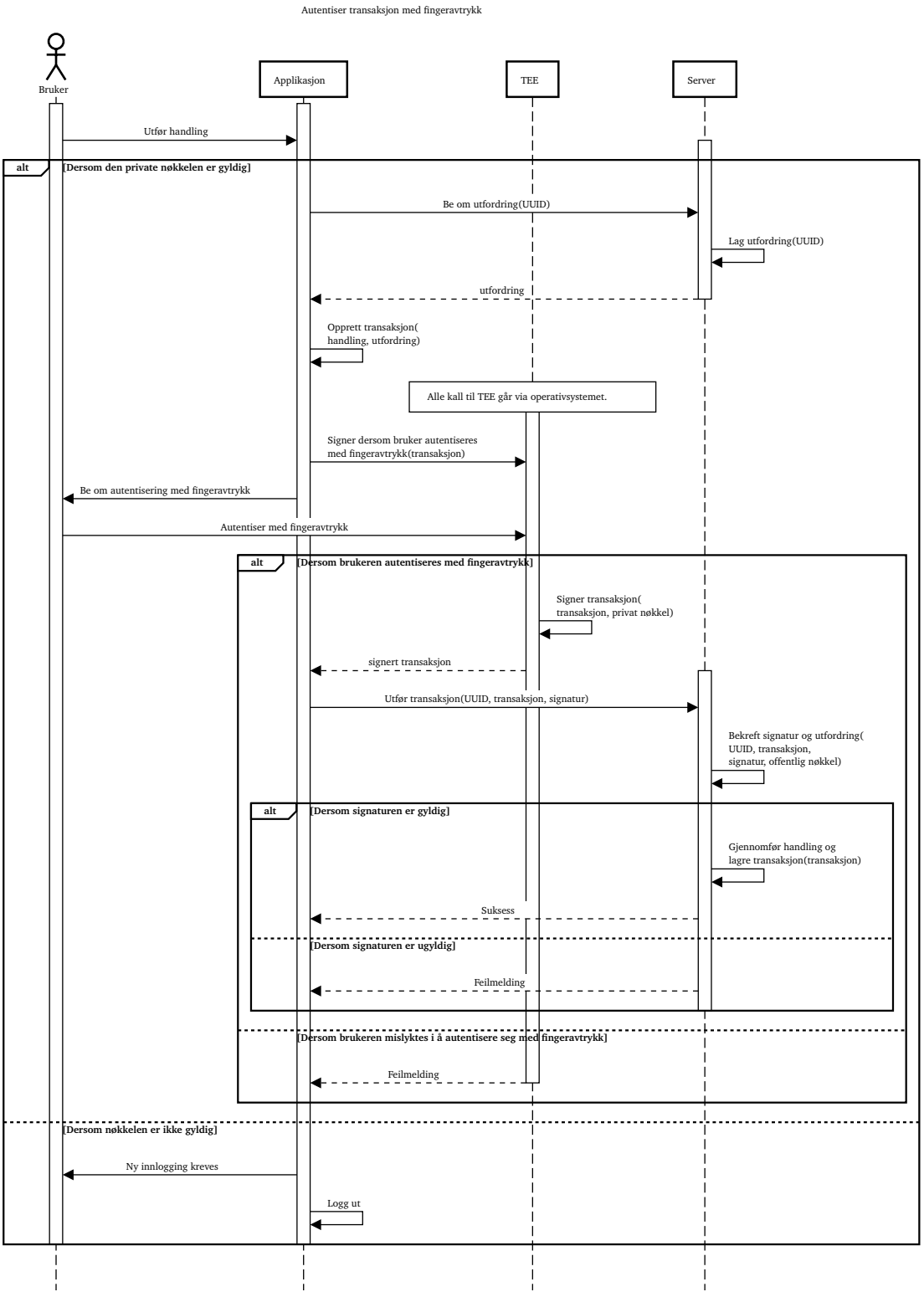
Vi antar at en av grunnene til at Apple ikke inkluderer en slik API er at de mener HTTPS er godt nok. Dersom noen skulle ha mulighet til å endre på den offentlige nøkkelen før den når transport-laget eller etter den har forlatt transport-laget, er det høy sannsynlighet for at de også vil har mulighet til å endre på nøkkelen på applikasjonslaget også.

Når det gjelder konfidensialitet, så er det kun innloggingsdetaljene som brukes til førstegangsautentisering (som er utenfor omfanget av oppgaven) som må krypteres av data som sendes til serveren. De offentlige nøklene kan kun brukes til å verifisere signaturene og regnes derfor ikke som sensitive.

Ved bruk av HTTPS, som vi har satt opp at applikasjonen krever, vil HTTPS sørge for konfidensialitet, integritet og autentisering av serveren under transport. Normalt sørger også HTTPS for beskyttelse mot replay-angrep, men TLS 1.3 (som nylig ble en foreslått standard på lik linje med TLS 1.2 [191], støtter 0-RTT [192] som effektivt fjerner



Figur 10: Et sekvensdiagram som illustrerer hvordan man autentiserer seg med fingeravtrykk, uten å opprette en økt



Figur 11: Et sekvensdiagram som illustrerer hvordan autentisere en transaksjon med fingeravtrykk

denne beskyttelsen. En angriper kan også gjennomføre et replay-angrep på applikasjonslaget dersom en sårbar enhet infiseres. Applikasjonen har derfor implementert replay-beskyttelse.

Diffie-Hellman Key Exchange (DHKE) i applikasjonslaget støttes kun av iOS ved å kompilere og linke OpenSSL-biblioteket manuelt, noe Apple normalt fraråder utviklere å gjøre med mindre man absolutt trenger det [193]. På Android er DHKE enklere å implementere, men vi har prøvd å unngå bruk av funksjonalitet som ikke er tilgjengelig eller anbefalt for iOS, ettersom omtrent halvparten av smarttelefonene i Norge har iOS [194]. En ekstra nøkkel-utveksling vil i tillegg føre til at applikasjonen kan virke tregere fordi man må gjennomføre en ekstra nøkkelutveksling før applikasjonen og serveren kan utveksle data.

Et bedre alternativ for å sikre integriteten til den permanente offentlige nøkkelen uten å måtte legge til en egen nøkkelutveksling, er å bruke en del av innloggingsdetaljene som kun er kjent for serveren og brukeren. Dersom man for eksempel brukte tidsbaserte engangspassord som en ekstra faktor, vil man kunne benytte HMAC for å både sikre integriteten til den permanente offentlige nøkkelen og autentisere brukeren. Forutsatt at man velger en sikker autentiseringsmetode til førstegangsautentisering, vil dette være den mest praktiske måten å sikre integriteten til den permanente offentlige nøkkelen. Passord passer dårlig til dette ettersom mange brukere har samme passord på ulike tjenester og serveren derfor kun burde lagre en hash av passordet. Vi har ikke implementert integritetssjekk for den offentlige nøkkelen fordi denne HMAC-beregningen vil avhenge av hvordan man implementerer autentiseringen som brukes i førstegangsautentiseringen.

5.1.7 Autentiseringsmetode for førstegangsautentisering

Autentiseringsmetoden som autentiserer brukeren til å begynne med, kan ikke benytte seg av fingeravtrykk eller nøkler i maskinvarestøttet nøkkellager, og er derfor utenfor omfanget av oppgaven. Vi vil allikevel komme med noen anbefalinger rundt hva slags autentiseringsmetode som burde brukes.

Autentiseringen i applikasjonen er kun så sterk som det svakeste leddet. Dersom autentiseringen som brukes til førstegangsautentisering (som også regnes som hovednøkkel) er svak, vil det ikke hjelpe om fingeravtrykk-autentiseringen er sterk. Denne autentiseringen må derfor også være sterk, som beskrevet i seksjon 5.1.3. Dersom passord skal være en av faktorene som brukes i denne autentiseringen, anbefaler vi OWASP sine tips til passord-krav[4]. Det eneste retningslinjen vi ville endret er å fjerne maksgrensen på 128 tegn. Brukerne burde få bruke passord på minst 256 tegn slik at de enkelt kan lage passordfraser. Så lenge man benytter seg av en nyere nøkkel-deriveringsfunksjon skal hele nøkkelen inkluderes i utregningen av passordhashen. For å svarteliste dårlige passord kan man blant annet svarteliste de mest vanlige passordene (som blant annet kan finnes på [5]) eller svarteliste alle passord som noen sinne har vært inkludert i et datainnbrudd [6]. NIST sin veiledning for digitale identiteter, som ble publisert 22. juni 2017, anbefaler å svarteliste sistnevnte [195].

[4] OWASP sine tips til passord-krav https://www.owasp.org/index.php?title=Authentication_Cheat_Sheet&oldid=228922#Implement_Proper_Password_Strength_Controls

[5] Liste over de mest vanlige passordene https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt

[6] Passord som har vært inkludert i datainnbrudd https://haveibeenpwned.com/Passwords

En enkel og sikker faktor nummer to er å implementere tidsbaserte engangspassord der brukerne kun trenger å laste ned en applikasjon som for eksempel Authy, Google Authenticator eller Microsoft Authenticator. Denne faktoren er også sårbar ovenfor phishing, men krever kun at brukerne laster ned en gratis applikasjon. Sikre autentiseringsmetoder sammenlignes i seksjon 3.5.3.

5.1.8 Beskyttelse mot replay-angrep

Den viktigste grunnen til å lagre nøkler i et maskinvarestøttet nøkkellager er at nøkkelen er beskyttet selv dersom en angriper skulle få root-privilegier på den mobile enheten. Dersom en angriper får root-privilegier til enheten, vil de derimot ha mulighet til å plukke opp pakker som allerede er signert. For at man skal være helt sikker på at brukeren har autentisert seg med fingeravtrykk, må brukeren signere ulik data hver gang. Dette trenger man en nonce/utfordring til. Det viktigste er at serveren vet helt sikkert at akkurat det som ble signert aldri har blitt mottatt før. Merk at dette ikke gir beskyttelse mot eventuelle angripere som klarer å endre på data før det sendes til TEE-en for å signeres.

Vi har illustrert to ulike metoder for å beskytte mot replay-angrep. I "Autentiser handling under økt" (figur 9) legges det nåværende tidspunktet til på transaksjonen før den signeres. Dersom det allerede har blitt mottatt en identisk transaksjon (med samme handling, tidspunkt og øktnummer), vil kallet ignoreres. Dersom transaksjonen ikke allerede er mottatt og signaturen er gyldig, vil serveren lagre unna transaksjonen og gjennomføre handlingen. Dette forutsetter at alle transaksjoner og alle øktnøkler blir lagret så lenge den tilhørende nøkkelen er gyldig. Det er mulig at man vil lagre alle transaksjoner og nøkler etter dette, men dette er et valg man må ta basert på om det er viktigst å spare lagringsplass (som uansett er relativt billig) eller om man vil beholde autentiseringsdata for alle handlingene brukeren har utført for å kunne holde brukeren ansvarlig for alle handlinger.

Merk at "Autentiser med fingeravtrykk og opprett økt" (figur 8) signerer kun en øktnøkkel. Denne øktnøkkelen må derfor lagres på serveren så lenge den tilhørende permanente nøkkelen eksisterer for å beskytte mot replay-angrep. Man kan spare lagringsplass ved å benytte seg av utfordringer som beskrevet under, men lagring av transaksjoner og nøkler er raskere.

"Autentiser med fingeravtrykk" (figur 10) og "Autentiser transaksjon med fingeravtrykk" (figur 11) går ut på at applikasjonen ber serveren om en utfordring (som kun skal forekomme én gang, for eksempel en nonce) hver gang brukeren skal autentisere seg. Serveren genererer da en utfordring, lagrer den i databasen og sender den til klienten. Klienten genererer deretter en signatur av denne utfordringen dersom brukeren autentiserer seg med fingeravtrykk og sender denne til serveren. Serveren sammenligner så den signaturen med utfordringen i databasen. Dersom signaturen stemmer er brukeren autentisert.

5.2 Implementasjon

Det første vi gjorde for å implementere denne applikasjonen var å se gjennom Google sin anbefalte metode for bruk av fingeravtrykk til å autentisere brukere[7]. Vi baserte imple-

[7] Android AsymmetricFingerprintDialog Sample: https://github.com/googlesamples/android-AsymmetricFingerprintDialog/

mentasjonen av genereringen, lagringen og uthentingen av nøklene på dette, og tilpasset det til denne applikasjonen.

5.2.1 Utvalgte kodesnutter

Vi har valgt å ta med noen utvalgte kodesnutter i rapporten som viser det vi mener er utfordrende eller sentrale deler av implementasjonen.

Overføring av nøkler til server

En av de største utfordringene som oppsto var at Java produserte binære nøkler med DER-enkoding[8], men autentiseringsserveren ville ha nøklene på PEM-format[9]. Vi testet mange av mulighetene for å endre formatet til nøkkelen, men fant til slutt ut at den beste løsningen var å manuelt kode om nøkkelen til PEM-format. Dette kunne heldigvis gjøres ganske enkelt ved å kode om nøkkelen til Base64-format og legge til nødvendig tekst før og etter nøkkelen. Implementasjonen av dette er illustrert i listing 5.1.

Listing 5.1: Koding av nøkkel til PEM-format

```
 try {
 //Setter opp pemOktKey på et format som kan leses av server
  pemOktKey = "-----BEGIN␣PUBLIC␣KEY-----\n"
  +android.util.Base64.encodeToString(fNokkel
  .getCertificate("OktNokkel").getPublicKey()
  .getEncoded(), android.util.Base64.DEFAULT)
  +"-----END␣PUBLIC␣KEY-----";
} catch (KeyStoreException e) {
  MainActivity.visFeilMelding("En␣feil␣har␣oppstått",
     FingerprintActivity.this);
}
```

Verifisering av signatur

Å bekrefte signaturer utført av den mobile enheten var også nokså utfordrende. Etter en stund fant vi ut at dette kunne gjennomføres nokså enkelt ved hjelp av OpenSSL-funksjonene i PHP. I listing 5.2 henter vi ut den permanente offentlige nøkkelen til brukeren fra databasen, dekoder signaturen fra Base64 til binærdata, og deretter sjekker vi om signaturen av den offentlige øktnøkkelen er gyldig. Dersom OpenSSL returnerte at signaturen var gyldig, vil serveren deretter lagre den offentlige øktnøkkelen og dermed kunne autentisere brukerens kall resten av økten ved å gjennomføre en lignende sjekk.

Listing 5.2: Verifisering av signatur

```
$sql = 'SELECT␣offentlig_nokkel␣FROM␣nokkel␣WHERE␣uuid␣=␣?';     // Henter PubKey
$sth = $this->dbh->prepare($sql); // Nøkkelen som øktnøkkelen skal være signert av
$sth->execute([$uuid]);
$rad = $sth->fetch(PDO::FETCH_ASSOC);
if($rad !== null) {
  $nokkel = openssl_get_publickey($rad['offentlig_nokkel']);     // Importerer nøkkel
}
else {
  $retur['suksess'] = false;
```

[8] DER-enkoding: https://en.wikipedia.org/w/index.php?title=X.690&oldid=820700186#DER_encoding

[9] PEM-format: https://en.wikipedia.org/w/index.php?title=Privacy-enhanced_Electronic_Mail&oldid=837833336

```
  $retur['feilmelding'] = 'Fant␣ikke␣offentlig␣␣nøkkel';
  return $retur;                // Returnerer feilmelding hvis PubKey ikke ble funnet
}

$binSignatur = base64_decode($signatur);            // Base64 -> Binær signatur
$res = openssl_verify($offentligOktnokkel, $binSignatur, $nokkel, OPENSSL_ALGO_SHA256);
if($res !== 1) {                                    // Dersom signaturen ikke stemmer
  if($res === -1) {                                 // Dersom en alvorlig feil oppstod,
    $retur['openssl_error'] = openssl_error_string(); // returner OpenSSL feilmelding
  }
  $retur['suksess'] = false;
  $retur['feilmelding'] = 'Ugyldig␣signatur';
  openssl_free_key($nokkel);
  return $retur;                // Returnerer feil dersom signaturen var ugyldig
}
else {                          // Dersom signauren er gyldig
  openssl_free_key($nokkel);    // Frigjør OpenSSL nøkkel-objektet og fortsett
}
```

Lagring av ikke-konfidensielle data

En annen utfordring under utviklingen var å beholde data, som for eksempel UUID, etter at brukeren lukker applikasjonen. Vi løste dette ved bruk av et SharedPreferences objekt, som peker til en fil som holder nøkkel-verdi par. Denne har ganske tunge operasjoner, men kan fint brukes i dette tilfellet hvor det ikke er mange verdier som skal ligge der [196]. Sikkerheten til SharedPreferences er den samme som alle andre Unix-baserte filer, så hvis noen har root-tilgang til mobiltelefonen kan de enkelt se innholdet i filen. Derfor har vi bare lagret nødvendig informasjon som ikke er spesielt sensitiv hvis den skulle komme i andre sine hender.

For å kommunisere med server tok vi i bruk Volley-biblioteket til Google. Dette biblioteket gir oss gode muligheter til å tilpasse det til eget bruk, og er laget for høy hastighet [197]. Ved å oppgi alle POST parametrene vi skal sende til serveren i en hashmap (assosiativ array), håndterer Volley resten av forespørselen når vi har lagt den inn i køen. Dette er illustrert i listing 5.3.

Listing 5.3: Volley tar imot parametere med en hashmap

```
public RegistrerForesporsel(String brukernavn, String passord, Response
    .Listener<String> listener) {
    super(Method.POST, MainActivity.HandlingsURL, listener, null);
    parametere = new HashMap<>();
    parametere.put("epost", brukernavn);
    parametere.put("passord", passord);
    parametere.put("registrer", "true");
}
```

Sikring av kommunikasjon mellom applikasjon og server

Sikkerheten i autentiseringsmetoden er avhengig av at kommunikasjonen mellom applikasjonen og serveren er sikker. HTTPS sørger allerede for god sikkerhet i form av konfidensialitet og integritet, men vi har implementert noen ekstra tiltak for å gjøre autentiseringen enda sikrere. En av disse tiltakene er “Certificate Pinning”, som er beskrevet i seksjon 4.4.2.

I manifestet til applikasjonen spesifiserte vi at applikasjonen skulle benytte seg av en

nettverkssikkerhets-konfigurasjonsfil[10]. I denne konfigurasjonsfilen inkluderte vi koden som vises i listing 5.4 for å pinne sertifikatene som benyttes av serveren.

Vi valgte å både pinne det aktive sertifikatet og sertifikatet til CA-en som signerte sertifikatet. På den måten vil kommunikasjonen fortsatt fungere dersom serveren får nytt sertifikat fra samme CA. Det er egentlig overflødig å spesifisere et sertifikat som er signert av sertifikatet til en CA man også spesifiserer, ettersom alle sertifikater som er signert av et den CA-en også vil være gyldige. Vi valgte å ta med begge sertifikatene allikevel for å illustrere at man kunne spesifisere flere sertifikater.

For å sørge for at en applikasjon kun benytter seg av HTTPS og all trafikk sendes kryptert, kan man også spesifisere dette i samme fil ved å angi attributten "cleartextTrafficPermitted" med verdien "false" . Dette sørger for at man ikke ved et uhell benytter seg av kall som kommuniserer i klartekst med serveren.

Listing 5.4: Nettverkssikkerhets-konfigurasjonsfil

```
<network-security-config>
  <!-- Tillat kun kryptert trafikk -->
  <domain-config cleartextTrafficPermitted="false">
    <!-- Instillingene skal gjelde for følgende domener -->
    <domain includeSubdomains="true">ntnu.no</domain>

    <!-- Sett utløpsdato for pins -->
    <pin-set expiration="2024-11-18">
      <!-- Pin sertifikater som skal tillates (en av sertifikatene i
           sertifikat-kjeden til serveren må ha en av følgende hasher)
             -->
      <pin digest="SHA-256">1b2/cPyvQcgOZfdIOHfTbqqJvmE/rfTtVPDduhkeBHQ
          =</pin>
      <pin digest="SHA-256">8651wEkMkH5ftiaLp57oqmx3KHTFzDgp7ZeJXR0ToBs
          =</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

5.2.2 GUI

For å bygge utseendet til applikasjonen har vi brukt Android Studio sin Layout Editor. Denne gjør det enkelt å søke opp forskjellige komponenter og legge de rett inn i applikasjonen med click-and-drag. Når man er fornøyd med utseendet, har man også fått ferdig generert XML-kode, så man kan ta i bruk komponentene med en gang.

5.3 Kvalitetssikring

Vi brukte statisk analyse for å kvalitetssikre applikasjonen. Dette har blitt gjort for all kildekode ved hjelp av Android Studio sitt innebygde verktøy, "Lint". Lint sjekker og varsler om den finner problemer relatert til sikkerhet, ytelse og kodestandarder [198].

5.4 Dokumentasjon

Det har blitt skrevet dokumentasjon for alle klasser og funksjoner i Javadoc format, sammen med noen inline kommentarer i koden hvor det er nødvendig. Koden til selve applikasjonen ble deretter kjørt gjennom Javadoc for generering av dokumentasjonen, og

[10]Network security config: https://developer.android.com/training/articles/security-config

serverkoden ble kjørt gjennom phpdoc for tilsvarende dokumentasjon. Dokumentasjonen ligger også i Github-repositoriet[11].

5.5 Kildekode

Kildekoden til applikasjonen ligger i følgende offentlige repository på Github:
https://github.com/ntnu-rgb/android-sikker-fingeravtrykk-autentisering.
En ferdigbygd apk-fil er tilgjengelig fra:
https://github.com/ntnu-rgb/android-sikker-fingeravtrykk-autentisering/releases.

5.6 Trusselmodellering

Vi har gjennomført trusselmodellering for å analysere sikkerheten til applikasjonen og autentiseringsmetoden. Som del av trusselmodelleringen har vi benyttet oss av følgende modeller som hjelper oss med å identifisere angrepsvektorer og sårbarheter: Misbrukstilfeller, dataflytdiagram, angrepstrær. Vi har benyttet oss av Microsoft Threat Modeling Tool for å identifisere trusler som ikke ble oppdaget gjennom den manuelle trusselmodelleringen. Etter å ha identifisert trusler benyttet vi oss av CVSS for å finne alvorlighetsgraden til hver trussel.

5.6.1 Misbrukstilfeller

Vi har benyttet misbrukstilfeller, som vist i figur 12, for å illustrere funksjonaliteten i applikasjonen, og hva ulike trusselaktører kan være ute etter å gjøre som truer denne funksjonaliteten.

De hvite boblene illustrerer brukstilfeller – de vanlige operasjonene som applikasjonen skal brukes til. De røde boblene illustrerer misbrukstilfeller – hvordan en ondsinnet aktør kan være ute etter å misbruke applikasjonen.

5.6.2 Dataflytdiagram (DFD)

Vi har laget et dataflytdiagram, som vist i figur 13, for å illustrere hvordan data beveger seg i systemet, og hvor de ulike tillitsgrensene er. Vi har identifisert følgende tillitsgrenser:

Mellom TEE og operativsystemet, mellom operativsystemet og applikasjonen, og mellom applikasjonen og serveren. Det er ved en av disse grensene en eventuell angriper sannsynligvis vil være ute etter å angripe systemet. Alle prosesser som kommuniserer med en annen prosess med et annet tillitsnivå, må sikres med for eksempel inputvalidering og autentisering.

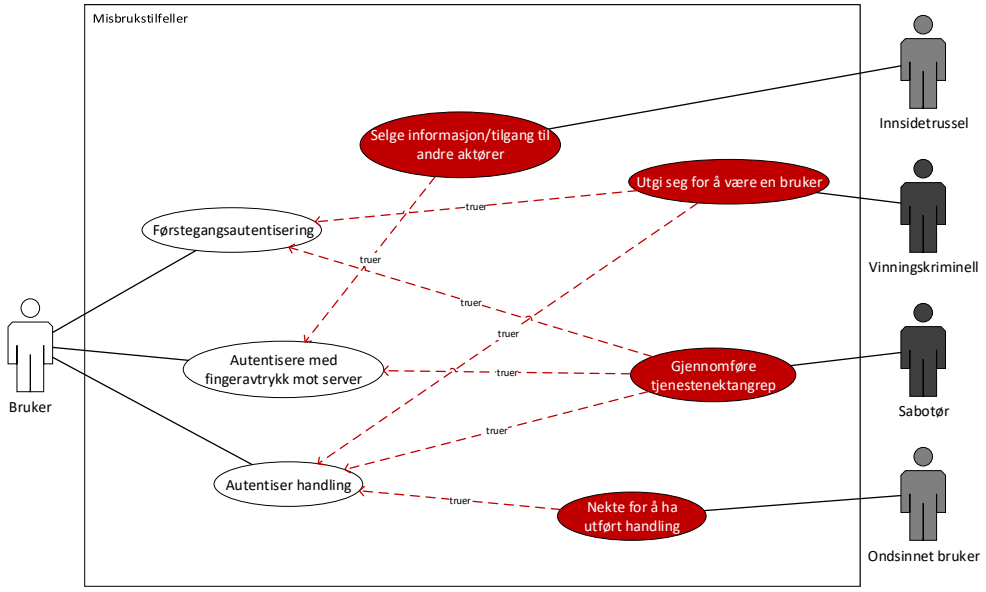
5.6.3 Angrepstrær

Angrepstrærne illustrerer de forskjellige angrepsvektorene en aktør kan benytte for å oppnå målet sitt.

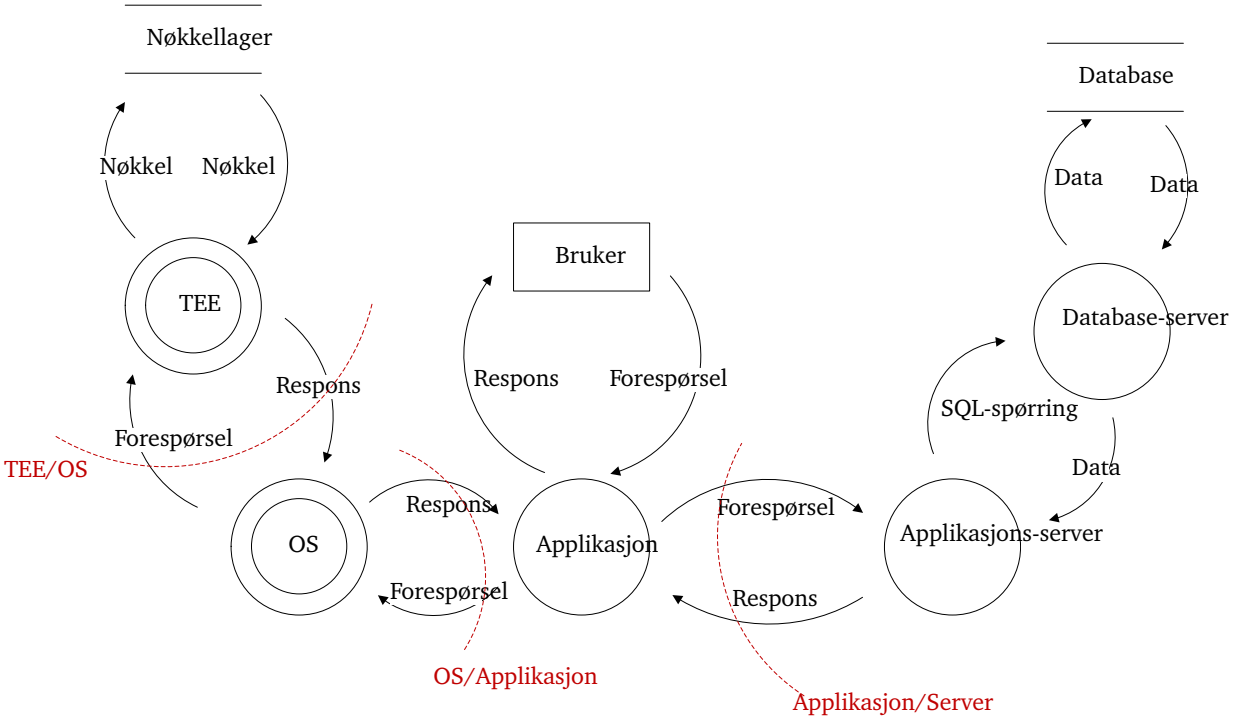
Utgi seg for å være en bruker

Det er mange forskjellige måter en ondsinnet aktør kan utgi seg for å være en legitim bruker. I angrepstreet (figur 14) har vi modellert de tre måtene det er mest sannsynlig at

[11] Dokumentasjon: https://github.com/ntnu-rgb/android-sikker-fingeravtrykk-autentisering/tree/master/docs



Figur 12: Misbrukstilfeller



Figur 13: Dataflytdiagram

en ondsinnet aktør kan oppnå dette målet på.

Bruk av forfalsket fingeravtrykk

Dersom man skal benytte seg av forfalsket fingeravtrykk for å utgi seg for å være brukeren, må man først og fremst få tak i den mobile enheten til brukeren. Mobile enheter har også beskyttelse som skal gjøre det vanskeligere å låse opp andre brukeres enheter. Dersom enheten ikke har vært i bruk de siste 24 til 48 timene, vil man måtte bruke vanlig kode/passord for å låse opp mobilen. Dersom man får tak i den mobile enheten til brukeren og det er under 24 timer siden enheten har vært i bruk, kan man benytte seg av et forfalsket fingeravtrykk for å utgi seg for å være brukeren. Mulige måter å lure en fingeravtrykk-leser er diskutert i seksjon 4.1.6.

Kapre privat nøkkel

For å kunne gjøre dette må den ondsinnede aktøren klare å finne og utnytte en sårbarhet i både operativsystemet og TEE.

Kapre innloggingsdetaljene til brukeren

Hvor enkelt eller vanskelig det vil være å kapre innloggingsdetaljene til brukeren, vil avgjøres av hvordan den overordnede autentiseringen implementeres.

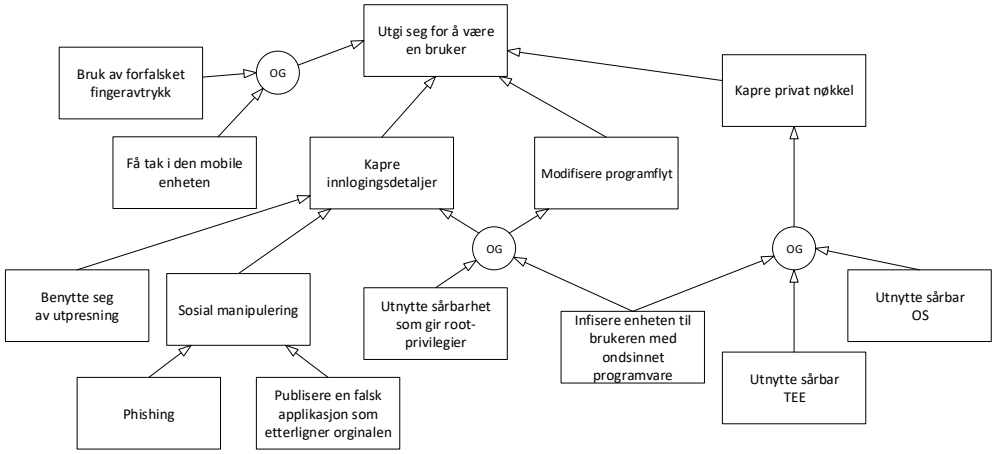
En ondsinnet aktør kan prøve å benytte seg av utpressing eller sosial manipulering for å få brukeren til å gi fra seg innloggingsdetaljene. Forutsatt at den tekniske sikkerheten er god, vil den enkleste måten å få tilgang til brukerens konto være å lure brukeren til å gi fra seg kontodetaljene. Dersom aktøren klarer å infisere enheten til brukeren med ondsinnet programvare og klarer å utnytte en sårbarhet som gir root-privilegier, vil aktøren trolig kunne plukke opp innloggingsdetaljene til brukeren. Dette forutsetter at angriperen klarer å infisere enheten før brukeren gjennomfører førstegangsautentisering, ettersom dette er de eneste nøklene som brukes til autentisering av brukeren og som befinner seg utenfor TEE.

En av de mest brukte metodene for å kapre innloggingsdetaljer, er sosial manipulering, der Phishing er en vanlig angrepsvektor. Fingeravtrykk-autentiseringen er sikret mot Phishing ved at fingeravtrykket og de tilhørende private nøklene ikke kan snappes opp av en eventuell angriper. Den overordnede autentiseringsmetoden som brukes til å autentisere brukeren og lagre den offentlige nøkkelen på serveren, vil derimot kunne være sårbar ovenfor Phishing, som beskrevet i seksjon 3.5.1.

En ondsinnet aktør kan også finne på å utvikle en applikasjon som etterligner den originale applikasjonen og som brukes til å samle inn innloggingsdetaljene til uoppmerksomme brukere. Normalt skal ikke dette aksepteres av Apple eller Google inn i applikasjonsbutikkene, men det kan være lurt å holde utkikk etter slike etterligninger. En sterk autentiseringsmetode som er immun mot Phishing, kan også hindre dette.

Modifisere programflyt

Ved å modifisere programflyt, kan man sørge for at applikasjonen signerer andre transaksjoner enn brukeren ønsket å gjennomføre. Dette er en svært komplisert form for angrep som krever at en avansert trusselaktør infiserer brukerens mobile enhet og utnytter en sårbarhet som gir skadevaren root-rettigheter, for å deretter modifisere programflyten uten at brukeren fatter mistanke. Denne angrepsvektoren er svært usannsynlig at benyttes, og kan effektivt hindres ved å holde enheten sin oppdatert og ved å unngå å laste ned potensielt skadelige applikasjoner.



Figur 14: Et angrepstre som illustrerer hvordan en aktør kan angi seg for å være en bruker

Selge informasjon/tilgang til andre aktører

Å selge informasjon eller tilgang til systemet kan kun gjøres av en eller flere på innsiden av systemet. Som vist i figur 15, må aktøren ha tilgang til å lese/skrive til databasen eller mulighet til å endre kildekoden. Økonomiske problemer eller utpressing kan ofte være årsaken til at noen vil selge informasjon eller tilgang. Det kan være andre grunner som ikke er lagt til i dette angrepstreet som for eksempel å hevne seg på arbeidsgiveren dersom de føler at de har blitt urettferdig behandlet eller lignende.

Gjennomføre tjenestenektangrep

Det er mange mulige måter en ondsinnet aktør kan gjennomføre tjenestenektangrep på, som vist i 16. Hvor enkelt og hvor effektivt angrepet er, vil avhenge av om aktøren har tilgang til innsiden av systemet eller ikke.

Logisk bombe

Dersom den ondsinnede aktøren er en på innsiden av systemet, kan vedkommende utføre tjenestenektangrep ved å legge til en form for logisk bombe.

Distribuert tjenestenekt/refleksjonsangrep mot applikasjonsserver

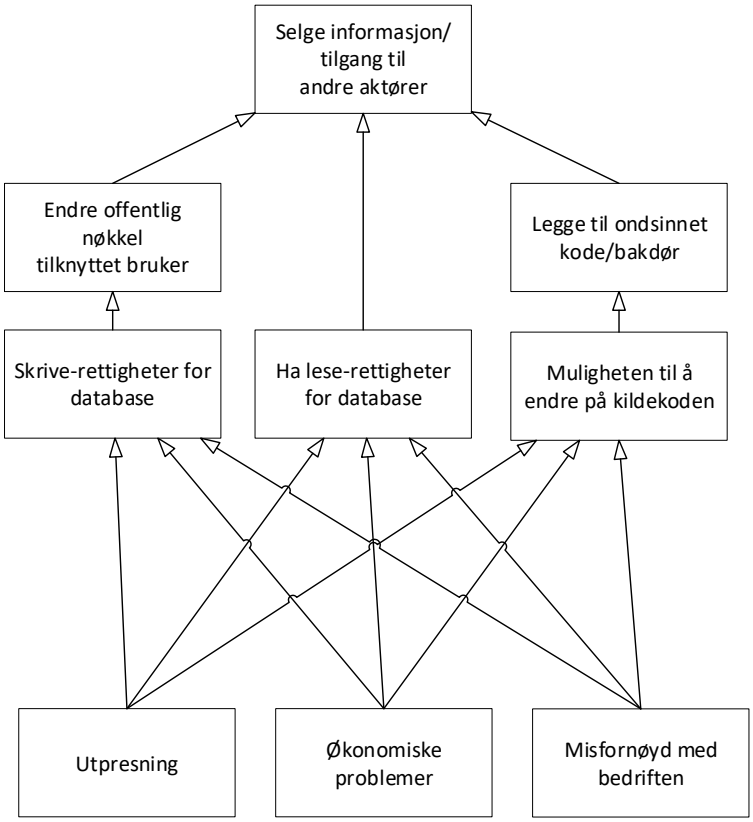
Ved å utføre et tjenestenektangrep som sender store mengder data til applikasjonsserveren, kan man overbelaste serveren og hindre legitime brukere i å utføre handlinger.

Tjenestenekt mot klient

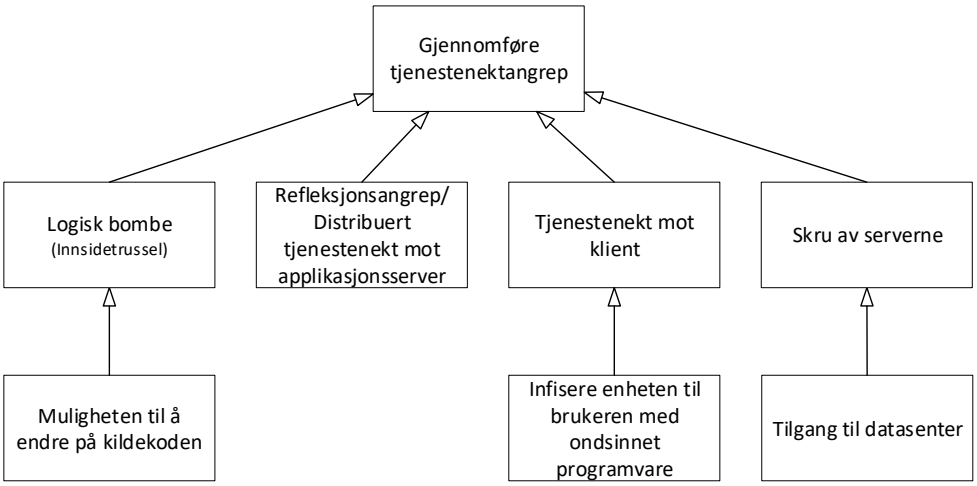
Dersom det eksisterer en sårbarhet i operativsystemet til den mobile enheten som gjør det mulig å gjennomføre tjenestenektangrep mot den mobile enheten til en bruker, vil en eventuell angriper kunne hindre brukeren i å benytte seg av enheten og dermed hindre brukeren i å benytte seg av applikasjonen.

Nekte for å ha utført handling

Nekte for å ha utført handling har svært få metoder under seg for å oppnå målet sitt, og har derfor ikke noe eget angrepstre. Denne handlingen baserer seg på at en bruker kan påstå at vedkommende ikke gjennomførte en spesifikk handling. For å kunne ansvarliggjøre brukeren må man derfor lagre alle signaturer og handlinger som brukeren har gjennomført. Vi har designet autentiseringsmetoden slik at en bruker ikke kan benekte



Figur 15: Et angrepstre som illustrerer hvordan en aktør kan selge informasjon/tilgang til andre aktører



Figur 16: Et angrepstre som illustrerer hvordan en aktør kan gjennomføre tjenestenektangrep

at vedkommende har gjennomført en handling, forutsatt at brukeren ikke har gitt fra seg innloggingsdetaljene til andre, eller har latt andre personer legge inn fingeravtrykket sitt på den mobile enheten sin. Man kan allikevel ansvarliggjøre brukeren ved å skrive at brukeren ikke får lov til å gjøre en av disse handlingene i betingelsene for bruk av tjenesten (ToS).

5.6.4 CVSS

For å identifisere hvor alvorlige de ulike truslene er, har vi regnet ut CVSS score for hver trussel.

Common Vulnerability Scoring System (CVSS) er en standard for klassifisering av sårbarheter. Man fyller inn en rekke egenskaper som for eksempel angrepsvektoren og påvirkningen som trusselene har på konfidensialiteten og integriteten til systemet. Resultatet er en verdi mellom null og ti som beskriver alvorlighetsgraden til sårbarheten, der ti er det mest alvorlige. I tillegg til å beskrive sårbarheten kan man også legge til tids- og miljømessige parametre for å beregne alvorlighetsgraden på nåværende tidspunkt i aktuelt miljø.

CVSS er en av OWASP sine anbefalte metoder for beregning av risiko tilknyttet trusler mot applikasjoner [199]. CVSS er i hovedsak utviklet for å regne ut alvorlighetsgraden til sårbarheter, men er også egnet til å regne ut alvorlighetsgraden til trusler ettersom en sårbarhet som gjør det mulig å realisere en trussel vil ha samme alvorlighetsgrad som trusselen.

OWASP anbefaler primært en metode som kalles DREAD for regne ut risikoen tilknyttet en programvaretrussel [199], men vi har valgt CVSS ettersom CVSS gjør det enkelt å tilpasse alvorlighetsgraden til sitt eget miljø. Ettersom vi utvikler et konseptbevis vil det trolig være nyttig å kunne tilpasse alvorlighetsgraden til sitt eget miljø.

For å regne ut alvorlighetsgraden benyttet vi oss av NIST sin CVSSv3 kalkulator[12]. For hver trussel vi har identifisert i seksjon 5.7, har vi også oppgitt CVSS base score, som er den grunnleggende alvorlighetsgraden uten noen tids- eller miljømessige faktorer.

For å identifisere hvor alvorlig truslene ville vært i sitt eget system kan man ta utgangspunkt i beregningene våre som er oppgitt i vedlegg C og fylle inn "Environmental Score Metrics". Metoden vi har benyttet oss av for å regne ut CVSS score presenteres også her.

5.7 Trusler

Etter å ha identifisert hvordan ulike aktører kan være ute etter å misbruke autentiseringsmetoden og hvordan aktørene kan oppnå dette, har vi bearbeidet denne informasjonen til konkrete trusler som truer sikkerheten til autentiseringsmetoden.

Noen trusler er identifisert ved hjelp av Microsoft SDL Threat Modeling Tool, som er beskrevet under seksjon 5.7.1.

5.7.1 Microsoft SDL Threat modeling tool

Microsoft SDL Threat Modeling Tool[13] ble tatt i bruk for å identifisere flest mulige trusler. Verktøyet tok dataflytdiagrammet vårt som input og utførte STRIDE per interaksjon på

[12] CVSS Calculator: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
[13] SDL TMT: https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

Tabell 7: Identifiserte trusler med CVSS-verdi

Trussel	CVSS
Den private nøkkelen eller CA-en som har signert sertifikatet kompromitteres	8,4
En hacker kan infisere en enhet med ondsinnet programvare og utnytte en sårbarhet i TEE og operativsystemet kan gjøre det mulig å hente ut private nøkler.	6,4
En hacker kan infisere en enheten med ondsinnet programvare og utnytte en sårbarhet som gir root-rettigheter for å endre programflyten i programmet og dermed utføre handlinger på vegne av brukeren.	6,4
Programvarefeil kan føre til at systemet ikke fungerer eller fungerer på en uønsket måte	5,9
En person med fysisk tilgang til den mobile enheten kan ta i bruk et forfalsket fingeravtrykk	5,7
En sabotør kan infisere enheten til brukeren med ondsinnet programvare slik at applikasjonen ikke kan kjøres	3,5

modellen for å identifisere potensielle trusler. For at verktøyet skulle gi nøyaktige resultater, måtte vi fylle inn detaljer om hvordan de ulike komponentene kommuniserte ved å angi attributter på interaksjonene.

Den største utfordringen ved bruk av dette verktøyet var å oversette detaljene i våre modeller til attributter som programmet forstår. Et eksempel er at verktøyet skulle vite om man hadde implementert beskyttelse mot replay-angrep. Vi benytter oss av unike nøkler og tidsstempler, men den nærmeste attributten i verktøyet var “other canary”.

Av truslene som verktøyet identifiserte, var det kun noen få trusler vi så på som relevante, trolig fordi applikasjonen implementerte noen typer kontroller som ikke enkelt kunne representeres med en av de forhåndsdefinerte attributtene til verktøyet. De truslene vi så på som relevante er tatt med i listen over identifiserte trusler i seksjon 5.7.2.

5.7.2 Identifiserte trusler

Vi har delt opp de identifiserte truslene i tre ulike kategorier: Trusler knyttet til applikasjon og autentisering, trusler knyttet til metode for førstegangsautentisering og operasjonelle trusler. For hver av truslene har vi beregnet CVSS-verdien, og hele utregningen for hver CVSS-verdi ligger i vedlegg C.

Trusler knyttet til applikasjon og autentisering

Trusler knyttet til applikasjon og autentisering kan kun sikres ved å forbedre sikkerheten på den mobile enheten eller designet til autentiseringsmetoden. Dette innebærer kun autentiseringsmetoden vi har beskrevet med nøkkelpar som låses opp med fingeravtrykkautentisering, og ikke autentiseringsmetoden som benyttes til førstegangsautentisering. Truslene knyttet til applikasjonen og autentiseringsmetoden vises i tabell 7.

Trusler knyttet til metode for førstegangsautentisering

Dersom man velger en svak metode for førstegangsautentiering, for eksempel kun brukernavn og passord, vil det oppstå noen trusler som ikke ellers ville vært til stede. Vi har identifisert noen trusler som kan oppstå dersom man velger en svak autentiseringsmetode til førstegangsautentisering. Truslene knyttet til metode for førstegangsautentisering vises i tabell 8.

Tabell 8: Identifiserte trusler for førstegangsautentisering

Trussel	CVSS
Innloggingsdetaljene kan bli kapret ved hjelp av phishing	8,3
En bruker kan bli lurt til å laste ned en falsk applikasjon som etterligner originalen for å kapre innloggingsdetaljene	7,1
En hacker kan infisere en enhet med ondsinnet programvare som utnytter en sårbarhet som gir root-rettigheter for å få tak i innloggingsdetaljene	7,1

Tabell 9: Identifiserte operasjonelle trusler

Trussel	CVSS
En sabotør kan gjennomføre et distribuert tjenestenektangrep mot serverene	7,5
En ansatt med tilgang til nettverket der webserveren og databaseserveren kommuniserer, kan endre data i sanntid	6,5
En bruker kan nekte for å ha gjennomført en handling dersom vedkommende delte innloggingsdetaljene sine med andre	6,5
En person med skriverettigheter til databasen kan endre den offentlige nøkkelen tilknyttet en bruker og selge tilgang til andre aktører	5,8
En person med tilgang til kildekoden kan legge inn ondsinnet kode eller en bakdør for å selge tilgang til andre aktører	5,7
En person med tilgang til datasenteret kan skru av serverene	4,9
En person med leserettigheter til databasen kan selge informasjon til andre aktører	4,4
En ansatt med muligheten til å endre koden kan legge inn en logisk bombe	3,9

Operasjonelle trusler

Applikasjonen og autentiseringsmetoden må ikke bare designes sikkert, men systemet må også implementeres og driftes på en sikker måte for å beskytte konfidensialiteten, integriteten og tilgjengeligheten til systemet. De operasjonelle truslene vises i tabell 9.

5.7.3 Foreslåtte tiltak

For å senke risiko tilknyttet truslene, kan man gjennomføre en rekke tiltak for å redusere sannsynligheten for at truslene realiseres. Vi har kommet frem til at følgende tiltak vil være svært nyttige å implementere.

- Benytt en sterk autentiseringsmetode (se 5.1.3) til førstegangsautentisering. Gjerne en autentiseringsmetode som gjør det vanskelig å gjennomføre phishing.
- Skriv at det ikke er tillatt å dele innloggingsdetaljene sine med andre i vilkårene for tjenesten.
- Lag en hendelseshåndteringsplan.
    - Utform en policy som sørger for at hendelseshåndteringsplanen vil bli utformet, at de som jobber med hendelseshåndtering har de nødvendige ressursene, at håndteringen øves på og at planen jevnlig oppdateres.
    - Planlegg hvem som skal gjøre hva ved ulike hendelser.
    - Loggfør alle hendelser.
- Sørg for at infrastrukturen er robust og skalerbar.
- Inngå samarbeid med en aktør som tilbyr tjenester tilknyttet beskyttelse mot DDoS-angrep.

- Jevnlig gjennomfør en felles kvalitetssjekk og gjennomgang av kode.
- Lag grundige automatiske tester.
- Benytt versjonskontroll for kildekoden.
- Benytt prinsippet om minste privilegium.
- Loggfør alle spørringer mot databasen som ikke kommer fra applikasjonen.
- Sørg for at systemet feiler på en sikker måte.

6 Konklusjon

I de foregående kapitlene har vi presentert resultatene fra prosjektet. I dette kapittelet vil vi prøve å analysere disse resultatene og arbeidet til gruppen, for å deretter gi et konkret svar på problemstillingen.

6.1 Diskusjon rundt resultater

En del av denne rapporten presenterer tidligere publisert informasjon og deler av rapporten benytter seg av kartlegging, vurdering og oppsummering av tidligere publisert informasjon. Vi mener resultatene i rapporten er veldig nyttig ettersom denne tidligere publiserte informasjonen var spredt over svært mange kilder og er nå samlet i denne rapporten. En rekke rapporter ga allerede god oversikt over sårbarhetene i SS7, men vår oppgave har tilføyd og drøftet informasjon rundt sikkerheten i norsk mobilinfrastruktur i tillegg til å lage en oversikt over hvordan aktører kan få tilgang til SS7.

NIST hadde allerede frarådet bruk av mobilnettverket til autentisering, men de har ikke gitt noen begrunnelse for hvorfor dette frarådes. Vi benyttet oss av informasjonen vi bearbeidet om sikkerheten i mobilinfrastrukturen til å trekke vår egen konklusjon basert på informasjonen vi samlet inn. Videre har vi også vurdert fordeler og ulemper med alternative kodebærere for bruk i Norge.

På delen om sikkerhet i mobilautentisering har vi kartlagt mye eksisterende informasjon om hvordan nøkkellagre er implementert og sikret. Vi har analysert fordeler og ulemper ved ulike metoder for lagring av nøkler og autentisering på mobile enheter, og deretter brukt dette til implementasjon av en applikasjon som autentiserer brukere mot et serversystem på en sikker måte. Applikasjonen baserte seg i stor grad på kode som Google har gitt ut, men vi har måttet designe store deler av applikasjonen på nytt for gjøre det mulig å opprette autentiserte økter. Vi har også måttet implementere databasen og serveren fra bunnen av ettersom Google sitt eksempel ikke kommuniserte med en ekte server.

6.2 Kritikk av oppgaven

Oppgaven har gjengitt resultater fra rapporter om sikkerhet i mobilinfrastrukturen, men resultatet fra disse rapportene gjenspeiler ikke nødvendigvis situasjonen i dag eller situasjonen blant norske mobiloperatører. Det har blitt implementert mange tiltak for å beskytte mot angrep via SS7, men det har ikke vært mulig å finne ut effekten av disse tiltakene. Vi skulle gjerne testet dagens sikkerhet i mobilinfrastrukturen, men har ikke hatt ressursene til dette på grunn av omfanget til oppgaven eller mulighet til dette av juridiske årsaker. Dette vil si at oppgaven kan gir feil bilde av situasjonen. Under seksjon 6.3 har vi derfor foreslått å undersøke effektiviteten til de implementerte tiltakene.

Sikkerhet knyttet til autentisering med SMS er ikke bare avhengig av sikkerheten i infrastrukturen, men også avhengig av beskyttelse mot sosial manipulering. Ettersom SIM-bytte fremstår som en trussel uansett hvor godt SS7 sikres, vil vi ikke anbefale å

bruke SMS til autentisering.

Applikasjonen vi utviklet kunne trolig hatt et ekstra lag med beskyttelse. Den krypterer kun data under transport med HTTPS og ikke i selve applikasjonen. Vi kom frem til at det ville ført til lite ekstra sikkerhet i forhold til ressursene det ville krevd å implementere. Dersom man skulle benyttet seg av en ekstra nøkkelutveksling ville det også tatt lengre tid for applikasjonen å koble seg til serveren. Det er mulig at vi allikevel burde prioritert å implementere dette, ettersom at målet var å implementere sikrest mulig autentisering.

6.3 Fremtidig arbeid

Ettersom effektiviteten til tiltakene i mobilinfrastrukturen er ukjent, vil vi anbefale at dette undersøkes. En slik undersøkelse vil trolig kreve samarbeid med en mobiloperatør. Det er også mulig at effektiviteten av disse tiltakene allerede er testet, men at resultatene ikke er publisert. Dersom dette er tilfellet, vil vi oppfordre til at disse resultatene publiseres for å gi et realistisk bilde av dagens sikkerhet i mobilinfrastrukturen.

Dersom det blir vanligere med maskinvarestøtte for nøkkelattestasjon i smarttelefoner, vil det også være nyttig å bygge inn en slik sjekk i applikasjonen for å sjekke at nøkkelen er lagret i et maskinvarestøttet nøkkellager og at nøkkelen ikke har blitt hentet ut fra det sikre miljøet der den ble generert. Det kan også være aktuelt å undersøke mulighetene til å implementere autentisering etter FIDO UAF protokollen.

6.4 Evaluering av gruppens arbeid

Vi mener at vi har jobbet godt med prosjektet og oppnådd tilfredsstillende resultater. Arbeidet har avviket en god del fra GANTT-skjemaet og prosjektplanen i vedlegg D, men vi har besvart problemstillingen og dekket hele oppgaven fra oppdragsgiver.

Et problem som førte til at fremgangen har avviket fra planen, er at vi brukte litt for mye tid på å undersøke mobil sikkerhet som ikke kan knyttes til direkte til sikkerhet i mobilinfrastruktur eller mobilautentisering. Vi beregnet også for lite tid til designet og trusselmodelleringen av applikasjonen.

Vi er fornøyde med resultatene våre og har lært mye om sikkerhet i mobilinfrastruktur, sikkerhet i mobile enheter og utvikling av applikasjoner til Android. Vi synes derfor prosjekt som arbeidsform har vært veldig lærerikt og motiverende. Vi har klart å fordele arbeidet godt ved hjelp av prosjektsyringsverktøyet Trello og jevnlige gruppemøter, og gruppearbeidet har dermed fungert godt.

6.5 Konklusjon

Problemstillingen gikk ut på å vurdere sikkerhet knyttet til bruk av SMS som autentiseringsmetode og sikkerheten i autentiseringsmetoder på mobile enheter. Gjennom en rekke rapporter har vi kommet frem til at den mobile infrastrukturen har alvorlige sårbarheter som gjør at engangskoder på SMS er dårlig egnet til autentisering. Vi har også kommet frem til at norske operatører gjør mye for å sikre infrastrukturen og har implementert tiltak i henhold til anbefalingene fra relevante bransjeorganisasjoner. Av den grunn vil vi anbefale å benytte seg av andre kodebærere som for eksempel programvarebaserte kodebærere eller sikkerhetsnøkler. SMS som kodebærer vil allikevel være sikrere

enn kun ett trinn i autentiseringen, men burde kun brukes dersom man ikke har mulighet til å benytte seg av en sikrere kodebærer.

Lagring av nøkler og autentisering med de lagrede nøklene kan gjøres veldig sikkert ved hjelp av nøkkellagre, og krav om at nøkler kun kan benyttes dersom brukeren autentiserer seg med fingeravtrykk eller andre biometriske autentiseringsmetoder. De fleste Android- og iOS-enhetene kommer med et eget sikret miljø som brukes til sensitive operasjoner, som for eksempel lagring av kryptografiske nøkler. Mange implementasjoner av et slikt sikkert miljø har inneholdt sårbarheter, men takket være mange lag med sikkerhet og obligatorisk adgangskontroll, er selv alvorlige sårbarheter vanskelig å utnytte.

Vi har utviklet et konseptbevis som implementerer sterk autentisering opp mot et serversystem, der en privat nøkkel kan benyttes til å signere data hvis brukeren autentiserer seg med fingeravtrykk. Den tilhørende offentlige nøkkelen overføres til serveren som kan verifisere at en transaksjon var signert av brukerens private nøkkel og dermed autentisere brukeren. Autentisering med slike nøkler kan ikke regnes som en permanent autentiseringsmetode ettersom den private nøkkelen vil slettes dersom man fjerner alle fingeravtrykkene sine eller fjerner låseskjermen. For at brukere ikke skal miste kontoene sine, må man benytte seg av en annen, sikker og mer permanent autentiseringsmetode som “hovednøkkel”.

Bibliografi

[1] Eika Gruppen AS. Om oss - Eika. https://www2.eika.no/eikagruppen/om-oss. (Besøkt Januar 2018).

[2] Eika Gruppen AS. Sikkerhet i mobilinfrastruktur/autentisering. Publisert under Innkomne oppgaveforslag for BIS3900, vår 2018.

[3] FIDO Alliance. FIDO UAF architectural overview. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html. (Besøkt Mars 2018).

[4] Telia. Hvor trygg er mobilen din – egentlig? https://e24.no/betalt-innhold/bak-tallene/hvor-trygg-er-mobilen-din-egentlig/23700089. (Besøkt Januar 2018).

[5] Davis, J. Two factor auth. https://twofactorauth.org/. (Besøkt Januar 2018).

[6] Thompson, I. Januar 2018. Who's using 2fa? sweet fa. less than 10% of gmail users enable two-factor authentication. http://www.theregister.co.uk/2018/01/17/no_one_uses_two_factor_authentication/. (Besøkt Januar 2018).

[7] Worstall, T. Mars 2013. More people have mobile phones than toilets. https://www.forbes.com/sites/timworstall/2013/03/23/more-people-have-mobile-phones-than-toilets/#702349dd6569. (Besøkt Januar 2018).

[8] Jørgenrud, M. Mars 2016. Mystisk sårbarhetstest slo ut telenor mobilnett. https://www.digi.no/artikler/mystisk-sarbarhetstest-slo-ut-telenor-mobilnett/348263. (Besøkt april 2018).

[9] GSM Association. Full members. https://www.gsma.com/membership/who-are-our-gsma-members/full-membership/. (Besøkt April 2018).

[10] Telenor ASA. Utlandspriser. https://www.telenor.no/privat/mobil/utlandet/#tab1=1. (Besøkt April 2018).

[11] Nohl, K. Desember 2014. Mobile self-defense. https://events.ccc.de/congress/2014/Fahrplan/system/attachments/2493/original/Mobile_Self_Defense-Karsten_Nohl-31C3-v1.pdf. (Besøkt april 2018).

[12] F5 bidragsytere. Diameter protocol. https://f5.com/glossary/diameter-protocol/. (Besøkt april 2018).

[13] Kim Gibbons. September 2017. Gsma guidelines for diameter firewall. https://www.netnumber.com/gsma-guidelines-diameter-firewall/. (Besøkt april 2018).

[14] Store Norske Leksikon. Oktober 2011. Imsi. https://snl.no/IMSI. (Besøkt april 2018).

[15] Store Norske Leksikon. August 2015. Imsi-fanger. https://snl.no/IMSI-fanger. (Besøkt april 2018).

[16] Myren, S. K. August 2015. Imsi fanger. *Store norske leksikon*. URL: https://snl.no/IMSI-fanger.

[17] Brombach, H. Desember 2014. Imsi-catchere har blitt brukt i årevis. https://www.digi.no/artikler/imsi-catchere-har-blitt-brukt-i-arevis/288006. (Besøkt april 2018).

[18] Jørgenrud, M. Juli 2017. – Dette baner vei for helt nye IMSI-catchere. 3G- og 4G/LTE-mobiler kan overvåkes. https://www.digi.no/artikler/dette-baner-vei-for-helt-nye-imsi-catchere-3g-og-4g-lte-mobiler-kan-overvakes/398405. (Besøkt april 2018).

[19] Nohl, K. Juli 2010. Attacking phone privacy. https://srlabs.de/wp-content/uploads/2010/07/Attacking.Phone_.Privacy_Karsten.Nohl_1-1.pdf. (Besøkt mai 2018).

[20] Chris Hoffman. August 2013. Sandboxes explained: How they're already protecting you and how to sandbox any program. https://www.howtogeek.com/169139/sandboxes-explained-how-theyre-already-protecting-you-and-how-to-sandbox-any-program/. (Besøkt april 2018).

[21] Android Source bidragsytere. Encryption. https://source.android.com/security/encryption/. (Besøkt april 2018).

[22] Technopedia bidragsytere. Biometric security. https://www.techopedia.com/definition/6203/biometric-security. (Besøkt april 2018).

[23] bidragsytere, I. Public key cryptography. https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.13/gtps7/s7pkey.html. (Besøkt april 2018).

[24] Maria Bartnes. Februar 2018. Https. https://snl.no/HTTPS. (Besøkt april 2018).

[25] Google. What is certificate transparency? https://www.certificate-transparency.org/what-is-ct. (Besøkt april 2018).

[26] Android Open Source Project. Features. https://source.android.com/security/keystore/features. (Besøkt mai 2018).

[27] Jørgenrud, M. Februar 2016. – et ondsinnet angrep mot telenor ville hatt samme konsekvens. https://www.digi.no/artikler/et-ondsinnet-angrep-mot-telenor-ville-hatt-samme-konsekvens/320604. (Besøkt april 2018).

[28] Positive Technologies. 2016. Primary security threats for SS7 cellular networks. https://www.ptsecurity.com/upload/ptcom/SS7-VULNERABILITY-2016-eng.pdf. (Besøkt Januar 2018).

[29] Evolved Intelligence SS7 signalling firewall selected by European tier 1 mobile operator group. https://www.gsma.com/membership/evolved-intelligence-ss7-signalling-firewall-selected-european-tier-1-mobile-operator-g (Besøkt April 2018).

[30] PT Telecom Attack Discovery. https://www.ptsecurity.com/ww-en/products/tad/. (Besøkt April 2018).

[31] The Communications Security, Reliability and Interoperability Council V – Working Group 10. Legacy systems risk reductions final report. https://www.fcc.gov/files/csric5-wg10-finalreport031517pdf. (Besøkt April 2018).

[32] European Union Agency For Network and Information Security. Signalling Security in Telecom SS7/Diameter/5G. https://www.enisa.europa.eu/publications/signalling-security-in-telecom-ss7-diameter-5g. (Besøkt April 2018).

[33] Jørgenrud, M. Mai 2016. Hemmelige tiltak gjør at teleoperatørene innfører mer datalagring. https://www.digi.no/artikler/hemmelige-tiltak-gjor-at-teleoperatorene-innforer-mer-datalagring/347928. (Besøkt april 2018).

[34] The Communications Security, Reliability and Interoperability Council V – Working Group 10. Legacy Systems and Services Risk Reduction Status Update. https://www.fcc.gov/files/csric5-wg10-presentation031517pptx. (Besøkt April 2018).

[35] NKOM. Risikovurdering av ekomsektoren. https://www.nkom.no/aktuelt/rapporter/_attachment/29084?_ts=15c9b3cff27. (Besøkt April 2018).

[36] Teknologiskifte i Telenors Infrastruktur. https://www.nkom.no/aktuelt/nyheter/_attachment/6895?_ts=13dd8a76e8f. (Besøkt April 2018).

[37] Nasjonal Sikkerhetsmyndighet. Falske basestasjoner - hvordan redusere risiko. https://www.nsm.stat.no/aktuelt/falske-basestasjoner-hvordan-redusere-risiko/. (Besøkt april 2018).

[38] TorProject bidragsytere. Tor: Overview. https://www.torproject.org/about/overview.html.en. (Besøkt april 2018).

[39] Brandom, R. Juni 2017. For 500, this site promises the power to track a phone and intercept its texts. https://www.theverge.com/2017/6/13/15794292/ss7-hack-dark-web-tap-phone-texts-cyber-crime. (Besøkt april 2018).

[40] LIFARS. Juni 2017. Hidden tor website offers phone snooping service for 500. https://lifars.com/2017/06/

hidden-tor-website-offers-phone-snooping-service-500/. (Besøkt april 2018).

[41] Technologies, P. 2017. Bitcoin wallet hacked via sms interception. https://www.youtube.com/watch?v=mLh1Nmqa6OM. (Besøkt april 2018).

[42] Telenor. Mvno. https://www.telenorwholesale.no/produkter/mvno/. (Besøkt april 2018).

[43] Zhilenko, A. Oktober 2016. How much does it cost to set up a data only mvno (mobile virtual network operator)? what is the required equipment? https://www.quora.com/How-much-does-it-cost-to-set-up-a-data-only-MVNO-Mobile-Virtual-Network-Operator-What- (Besøkt april 2018).

[44] Khandelwal, S. Mai 2017. Real-world ss7 attack — hackers are stealing money from bank accounts. https://thehackernews.com/2017/05/ss7-vulnerability-bank-hacking.html. (Besøkt april 2018).

[45] Tanriverdi, H. & Zydra, M. Mai 2017. Schwachstelle im mobilfunknetz: Kriminelle hacker räumen konten leer. http://www.sueddeutsche.de/digital/it-sicherheit-schwachstelle-im-mobilfunknetz-kriminelle-hacker-raeumen-konten-leer-1.3486504. (Besøkt april 2018).

[46] Kaspersky Lab. August 2016. Cybercriminals recruit insiders to attack telecoms providers. https://www.kaspersky.com/about/press-releases/2016_cybercriminals-recruit-insiders-to-attack-telecoms-providers. (Besøkt april 2018).

[47] Weaver, N. April 2016. Nick asks the nsa: Signaling system 7 (ss7). https://www.lawfareblog.com/nick-asks-nsa-signaling-system-7-ss7. (Besøkt april 2018).

[48] Mackey, A. & Kalia, A. Januar 2016. Companies should resist government pressure and stand up for free speech. https://www.eff.org/deeplinks/2016/01/companies-should-resist-government-pressure-and-stand-free-speech. (Besøkt april 2018).

[49] Crocker, A. Desember 2015. No matter what the fbi says, compromising encryption is a technical issue. https://www.eff.org/deeplinks/2015/12/no-matter-what-fbi-says-compromising-encryption-technical-issue. (Besøkt april 2018).

[50] Fox-Brewster, T. Mai 2016. For 20m, these israeli hackers will spy on any phone on the planet. https://www.forbes.com/sites/thomasbrewster/2016/05/31/ability-unlimited-spy-system-ulin-ss7/#707ef12e63fa. (Besøkt april 2018).

[51] E24. April 2016. Skulle ikke fått nettet til å falle sammen. https://e24.no/digital/telenor/signaler-fra-luxembourg-slo-ut-telenor-nettet-testene-skulle-ikke-faatt-nettet-til-aa-f23660244. (Besøkt april 2018).

[52] Kibar, O. Februar 2015. Sirklet inn fra utlandet. https://www.dn.no/magasinet/2015/02/13/2219/Teknologi/sirklet-inn-fra-utlandet. (Besøkt april 2018).

[53] Skjetne, O. L. Mai 2017. Fn-ekspert advarer mot storbritannias masseovervåkning. https://www.vg.no/nyheter/utenriks/i/84XnE/fn-ekspert-advarer-mot-storbritannias-masseovervaakning. (Besøkt april 2018).

[54] Rasch, J. S. Juli 2013. Le monde: Frankrike driver med enorm dataovervåkning. https://www.dagbladet.no/nyheter/le-monde-frankrike-driver-med-enorm-dataovervakning/62427305. (Besøkt april 2018).

[55] VG. Desember 2013. Nsa overvåker flere hundre millioner mobiltelefoner. https://www.vg.no/nyheter/utenriks/i/Ov23q/nsa-overvaaker-flere-hundre-millioner-mobiltelefoner. (Besøkt april 2018).

[56] Johansen, P. A., Færaas, A., & Bleikelia, M. Juli 2013. Disse landene kan overvåke din nett- og telefonbruk. https://www.aftenposten.no/norge/i/dd1VO/Disse-landene-kan-overvake-din-nett--og-telefonbruk. (Besøkt april 2018).

[57] Branigan, T. Januar 2010. State owned china mobile is world's biggest mobile phone operator. https://www.theguardian.com/business/2010/jan/11/china-mobile-telecomms. (Besøkt april 2018).

[58] Wikipedia bidragsytere. 2018. China mobile. https://en.wikipedia.org/w/index.php?title=China_Mobile&oldid=832665001. (Besøkt april 2018).

[59] Wikipedia bidragsytere. 2018. Mobily. https://en.wikipedia.org/w/index.php?title=Mobily&oldid=826103361. (Besøkt april 2018).

[60] Store Norske Leksikon. Januar 2017. Telenor ASA. https://snl.no/Telenor_ASA. (Besøkt april 2018).

[61] Johansen, P. A., Foss, A. B., & Hager-Thoresen, F. Januar 2015. Mobilovervåkingen: Metoderapport til skup-prisen 2014. *Aftenposten*. URL: https://www.skup.no/rapporter/2014/mobilovervakingen.

[62] Nohl, Karsten. 2014. Mobile self-defense. https://events.ccc.de/congress/2014/Fahrplan/system/attachments/2493/original/Mobile_Self_Defense-Karsten_Nohl-31C3-v1.pdf. (Besøkt april 2018).

[63] bidragsytere, T. Mobile switching center (msc). https://www.techopedia.com/definition/8448/mobile-switching-center-msc. (Besøkt april 2018).

[64] bidragsytere, W. Mobile application part. https://en.wikipedia.org/w/index.php?title=Mobile_Application_Part&oldid=827012243. (Besøkt mai 2018).

[65] Jensen, K. Juni 2016. Improving ss7 security using machine learning techniques. https://brage.bibsys.no/xmlui/bitstream/id/440836/KJensen_2016.pdf. (Besøkt Januar 2018).

[66] Ghadialy, Z. Juli 2004. Camel: An introduction. https://www.3g4g.co.uk/Tutorial/ZG/zg_camel.html. (Besøkt april 2018).

[67] Oliver, I. e. a. 2016. User location tracking attacks for lte networks using the interworking functionality. http://dl.ifip.org/db/conf/networking/networking2016/1570236202.pdf. (Besøkt april 2018).

[68] Káčer, M. & Langlois, P. Juli 2017. Ss7 attacker heaven turns into riot: How to make nation-state and intelligence attackers' lives much harder on mobile networks. https://www.blackhat.com/docs/us-17/wednesday/us-17-Kacer-SS7-Attacker-Heaven-Turns-Into-Riot-How-To-Make-Nation-State-And-Intelliger pdf. (Besøkt april 2018).

[69] Jørgenrud, M. Mars 2016. «ingen kan fullgodt hindre misbruk» av sårbare protokoller fra 70-tallet. https://www.digi.no/artikler/ingen-kan-fullgodt-hindre-misbruk-av-sarbare-protokoller-fra-70-tallet/348218. (Besøkt april 2018).

[70] Foss, A. B. Januar 2018. Politiet brukte falske basestasjoner i 107 tilfeller i fjor – kraftig vekst fra året før. https://www.aftenposten.no/norge/i/zLrGr9/Politiet-brukte-falske-basestasjoner-i-107-tilfeller-i-fjor--kraftig-vekst-fra-aret-for (Besøkt april 2018).

[71] Færaas, A. Desember 2013. Nsa knekker kryptering som hindrer mobilavlytting. https://www.aftenposten.no/norge/i/3jXKq/NSA-knekker-kryptering-som-hindrer-mobilavlytting. (Besøkt februar 2018).

[72] Cichonski, J. & Franklin, J. April 2015. Lte security – how good is it? https://www.rsaconference.com/writable/presentations/file_upload/tech-r03_lte-security-how-good-is-it.pdf. (Besøkt april 2018).

[73] Deutsche Telekom AG. Desember 2013. Devices that support the a5/3 encryption standard. https://www.telekom.com/en/media/media-information/archive/devices-that-support-the-a5-3-encryption-standard-360316. (Besøkt april 2018).

[74] Kvalheim, F. J. Februar 2016. Har det dukket opp et nytt symbol på mobilen din? https://www.tek.no/artikler/har-det-dukket-opp-et-nytt-symbol-pa-mobilen-din/276823. (Besøkt april 2018).

[75] Jørgenrud, M. Mai 2012. Huawei nekter for ulovlig dumping. https://www.digi.no/artikler/huawei-nekter-for-ulovlig-dumping/200020. (Besøkt april 2018).

[76] Rossen, E. Mars 2014. Boikott biter ikke på huawei. https://www.digi.no/artikler/boikott-biter-ikke-pa-huawei/287309. (Besøkt april 2018).

[77] Messmer, E. Nov 08 2012. Huawei security chief: We can help keep u.s. safe from 'net threats. *Network World (Online)*. Copyright - Copyright 2012 Network World, Inc. All Rights Reserved; People - Purdy, Andy; Borg, Scott; Streufert, John; Last updated - 2016-03-14; SubjectsTermNotLitGenreText - Purdy, Andy; Borg, Scott; Streufert, John. URL: https://search.proquest.com/docview/1151061172.

[78] Al Bawaba. Oct 09 2012. Analysts blame politics, not security, for huawei, zte allegations. *Computer News Middle East*. Copyright - Copyright Al Bawaba (Middle East) Ltd. Oct 9, 2012; Last updated - 2012-10-10. URL: https://search.proquest.com/docview/1095405836.

[79] Jørgenrud, M. Februar 2014. Pst advarer igjen mot huawei. https://www.digi.no/artikler/pst-advarer-igjen-mot-huawei/287012. (Besøkt april 2018).

[80] Rossen, E. Oktober 2012. Telenor: – huawei er en risiko vi tar. https://www.digi.no/artikler/telenor-huawei-er-en-risiko-vi-tar/292485. (Besøkt april 2018).

[81] Jørgenrud, M. Mai 2013. Sterkt imot granskning av huawei. https://www.digi.no/artikler/sterkt-imot-granskning-av-huawei/288763. (Besøkt april 2018).

[82] Rossen, E. Mars 2014. Eu dropper sak mot huawei. https://www.digi.no/artikler/eu-dropper-sak-mot-huawei/287315. (Besøkt april 2018).

[83] Greenberg, A. Juni 2016. So hey you should stop using texts for two-factor authentication. https://www.wired.com/2016/06/hey-stop-using-texts-two-factor-authentication/. (Besøkt april 2018).

[84] Google. Google 2-trinns bekreftelse. https://www.google.com/landing/2step/. (Besøkt April 2018).

[85] Larson, T. September 2013. Two-step vs. two-factor authentication - is there a difference? https://security.stackexchange.com/a/41981. (Besøkt april 2018).

[86] Krebs, B. Juni 2012. Attackers hit weak spots in 2-factor authentication. https://krebsonsecurity.com/2012/06/attackers-target-weak-spots-in-2-factor-authentication/. (Besøkt april 2018).

[87] Aizomai, M., Josang, A., McCullagh, A., & Foo, E. Dec 2008. Strengthening sms-based authentication through usability. In *2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*, 683–688. doi:10.1109/ISPA.2008.57.

[88] Statistisk sentralbyrå. Norsk mediebarometer: Andel som har tilgang til ulike medier og elektroniske tilbud i hjemmet (prosent), etter medietype, statistikkvariabel og år. http://www.ssb.no/statbank/sq/10005654/. (Besøkt april 2018).

[89] medienorge & Kantar TNS. Andel som har smarttelefon. http://www.medienorge.uib.no/statistikk/medium/ikt/379. (Besøkt april 2018).

[90] Knudsen, E. Februar 2017. 99,6 prosent av mobilmarkedet er nå eid av to operativsystem. https://www.tek.no/artikler/na-har-android-og-ios-tatt-nesten-i-hele-i-mobilmarkedet/376773. (Besøkt april 2018).

[91] Statistisk sentralbyrå. September 2017. Bruk av ikt i husholdningene. https://www.ssb.no/ikthus. (Besøkt april 2018).

[92] Brombach, H. Oktober 2014. Logg inn på google med usb-pinne. https://www.digi.no/artikler/logg-inn-pa-google-med-usb-pinne/287913. (Besøkt april 2018).

[93] Yubico. Fido u2f. https://www.yubico.com/solutions/fido-u2f/. (Besøkt april 2018).

[94] FIDO Alliance. Desember 2016. Case study series: Google security keys work. https://fidoalliance.org/case-study-series-google-security-keys-work/. (Besøkt april 2018).

[95] Proshop AS. Yubico yubikey neo. https://www.proshop.no/Kabinett-tilbehoer-kjoelepasta-mv/Yubico-YubiKey-NEO/2649451. (Besøkt april 2018).

[96] Dustin Norway AS. Yubico yubikey u2f. https://www.dustinhome.no/product/5010932411/yubikey-u2f. (Besøkt april 2018).

[97] Griffith, E. Februar 2018. Two-factor authentication: Who has it and how to set it up. https://www.pcmag.com/feature/358289/two-factor-authentication-who-has-it-and-how-to-set-it-up. (Besøkt april 2018).

[98] Futsæter, K.-A. Januar 2017. Nå har 99 prosent av alle mellom 12 og 49 år en smarttelefon. https://www.medier24.no/artikler/na-har-99-prosent-av-alle-mellom-12-og-49-ar-en-smarttelefon/366987. (Besøkt april 2018).

[99] Wikipedia bidragsytere. April 2018. Time-based one-time password algorithm. https://en.wikipedia.org/w/index.php?title=Time-based_One-time_Password_algorithm&oldid=835911425#Weaknesses_and_vulnerabilities. (Besøkt april 2018).

[100] Dunn, J. E. November 2016. Hacker used password resets to break into 1,050 university email accounts. https://nakedsecurity.sophos.com/2016/11/08/hacker-used-password-resets-to-break-into-1050-university-email-accounts/. (Besøkt april 2018).

[101] The MITRE Corporation. Cwe-640: Weak password recovery mechanism for forgotten password. https://cwe.mitre.org/data/definitions/640.html. (Besøkt april 2018).

[102] Kralevich, N. Juli 2017. Honey, i shrunk the attack surface. https://www.blackhat.com/docs/us-17/thursday/us-17-Kralevich-Honey-I-Shrunk-The-Attack-Surface-Adventures-In-Android-Security-Harden pdf. (Besøkt april 2018).

[103] Zovi, D. D. & Miller, C. 2012. ios security internals. https://www.rsaconference.com/writable/presentations/file_upload/mbs-402.pdf. (Besøkt februar 2018).

[104] Elenkov, N. 2014. *Android Security Internals: An In-Depth Guide to Android's Security Architecture*. No Starch Press, San Francisco, CA, USA, 1st edition.

[105] Apple. Januar 2018. iOS security: iOS 11. https://www.apple.com/business/docs/iOS_Security_Guide.pdf. (Besøkt april 2018).

[106] Android Open Source Project. Implementing dm-verity. https://source.android.com/security/verifiedboot/dm-verity. (Besøkt april 2018).

[107] Qualcomm Technologies, Inc. Secure boot and image authentication technical overview. https://www.qualcomm.com/documents/secure-boot-and-image-authentication-technical-overview. (Besøkt april 2018).

[108] SAMSUNG. Platform security. http://developer.samsung.com/tech-insights/knox/platform-security. (Besøkt april 2018).

[109] Friedman, A. Desember 2017. Qualcomm had the highest share of the smartphone soc market during q3 2017, followed by apple. https://www.phonearena.com/news/Qualcomm-had-the-highest-share-of-the-smartphone-SoC-market-during-Q3-2017-followed-by-id101167. (Besøkt april 2018).

[110] Roy, A., Memon, N., & Ross, A. Sept 2017. Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems. *IEEE Transactions on Information Forensics and Security*, 12(9), 2013–2025. doi:10.1109/TIFS.2017.2691658.

[111] Security Research Labs GmbH. September 2013. Breakthrough 3d fingerprint authentication with snapdragon sense id. https://srlabs.de/bites/spoofing-fingerprints/. (Besøkt april 2018).

[112] Triggs, R. Februar 2018. How fingerprint scanners work: optical, capacitive, and ultrasonic variants explained. https://www.androidauthority.com/how-fingerprint-scanners-work-670934/. (Besøkt april 2018).

[113] Goel, V. April 2017. That fingerprint sensor on your phone is not as safe as you think. https://www.nytimes.com/2017/04/10/technology/fingerprint-security-smartphones-apple-google-samsung.html. (Besøkt april 2018).

[114] Google. That fingerprint sensor on your phone is not as safe as you think. https://support.google.com/nexus/answer/6285273#unlock_device. (Besøkt april 2018).

[115] GadgetGuideOnline.com. Changes on using fingerprint to unlock Galaxy S6 in Android Marshmallow update for Galaxy S6, S6 edge and S6 edge+. https://gadgetguideonline.com/galaxys6/samsung-galaxy-s6-guides/changes-on-using-fingerprint-to-unlock-galaxy-s6-in-android-marshmallow-update/. (Besøkt april 2018).

[116] Reed, T. Mars 2018. GrayKey iPhone unlocker poses serious security concerns. https://blog.malwarebytes.com/security-world/2018/03/graykey-iphone-unlocker-poses-serious-security-concerns/. (Besøkt april 2018).

[117] Størbu, M. K. Mai 2018. Slik gjør apple det vanskeligere for «iphone-åpneren». https://itavisen.no/2018/05/09/slik-gjor-apple-det-vanskeligere-for-iphone-apneren/. (Besøkt mai 2018).

[118] Gartner. Februar 2017. Gartner says worldwide sales of smartphones grew 7 percent in the fourth quarter of 2016. https://www.gartner.com/newsroom/id/3609817. (Besøkt april 2018).

[119] Lopez, N. Oktober 2017. Stock android is no longer the best android. https://thenextweb.com/opinion/2017/10/05/stock-android-is-no-longer-the-best-version-of-android/. (Besøkt april 2018).

[120] Rutnik, M. Mars 2018. What is stock android? https://www.androidauthority.com/what-is-stock-android-845627/. (Besøkt april 2018).

[121] Wu, L., Grace, M., Zhou, Y., Wu, C., & Jiang, X. 2013. The impact of vendor customizations on android security. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, 623–634, New York, NY, USA. ACM. URL: http://doi.acm.org/10.1145/2508859.2516728, doi:10.1145/2508859.2516728.

[122] Kleidermacher, D. Mars 2019. Android security 2017 year in review. https://security.googleblog.com/2018/03/android-security-2017-year-in-review.html. (Besøkt april 2018).

[123] Childs, D. November 2017. The results – mobile pwn2own 2017 day two. https://www.thezdi.com/blog/2017/11/2/the-results-mobile-pwn2own-2017-day-two. (Besøkt april 2018).

[124] Green, M. November 2016. The limitations of android N encryption. https://blog.cryptographyengineering.com/2016/11/24/android-n-encryption/. (Besøkt april 2018).

[125] Android Open Source Project. File-based encryption. https://source.android.com/security/encryption/file-based. (Besøkt april 2018).

[126] Hoffman, C. Desember 2017. Why your android phone isn't getting operating system updates and what you can do about it. https://www.howtogeek.com/129273/why-your-android-phone-isnt-getting-operating-system-updates-and-what-you-can-do-about- (Besøkt april 2018).

[127] Raphael, J. Februar 2017. The ugly truth behind android's upgrade problem. https://www.computerworld.com/article/3175067/android/android-upgrade-problem.html. (Besøkt april 2018).

[128] Raphael, J. Februar 2018. Android upgrade report card: Grading the manufacturers on oreo. https://www.computerworld.com/article/3257607/android/android-upgrade-report-card-oreo.html. (Besøkt april 2018).

[129] Android Open Source Project. April 2018. Android security bulletin—april 2018. https://source.android.com/security/bulletin/2018-04-01. (Besøkt april 2018).

[130] Google. How long your google play edition device will get updates. https://support.google.com/nexus/answer/4457705?hl=en#googleplayedition_devices. (Besøkt april 2018).

[131] Android enterprise recommended requirements. https://www.android.com/enterprise/recommended/requirements/. (Besøkt april 2018).

[132] Android Open Source Project. Version binding. https://source.android.com/security/keystore/version-binding. (Besøkt april 2018).

[133] Hager, R. September 2017. Android oreo feature spotlight: Rollback protection, a new part of verified boot, won't allow you to start a downgraded os. https://www.androidpolice.com/2017/09/05/android-oreo-feature-spotlight-changes-verified-boot-wont-allow-start-downgraded-os/. (Besøkt april 2018).

[134] Computerworld. August 2017. Mingis on tech: Android vs ios - which is more secure? https://www.youtube.com/watch?v=uE31gOo4xAU. (Besøkt mars 2018).

[135] Cunningham, E. Mai 2017. Keeping you safe with google play protect. https://blog.google/products/android/google-play-protect/. (Besøkt april 2018).

[136] Devine, R. Juni 2012. Face unlock in jelly bean gets a 'Liveness check'. https://www.androidcentral.com/face-unlock-jelly-bean-gets-liveness-check. (Besøkt april 2018).

[137] Duino, J. August 2015. Trusted face in lollipop, explained. https://www.androidcentral.com/face-unlock-explained. (Besøkt april 2018).

[138] Google. Smart lock. https://get.google.com/smartlock/. (Besøkt april 2018).

[139] Nguyen, D. Februar 2009. Your face is NOT your password. https://www.blackhat.com/presentations/bh-dc-09/Nguyen/BlackHat-DC-09-Nguyen-Face-not-your-password-slides.pdf. (Besøkt april 2018).

[140] Microsoft. Face API. https://azure.microsoft.com/en-us/services/cognitive-services/face/. (Besøkt april 2018).

[141] AR, Inc., K. Identity, emotions, and demographics - all in one place. https://www.kairos.com/features. (Besøkt april 2018).

[142] SAMSUNG. Samsung pass. https://www.samsung.com/us/support/owners/app/samsung-pass. (Besøkt april 2018).

[143] Chaos Computer Club. Mai 2017. Hacking the samsung galaxy s8 irisscanner. https://media.ccc.de/v/biometrie-s8-iris-en. (Besøkt april 2018).

[144] Android Open Source Project. Android 6.0 APIs. https://developer.android.com/about/versions/marshmallow/android-6.0.html#fingerprint-authentication. (Besøkt april 2018).

[145] Android Open Source Project. Android keystore system. https://developer.android.com/training/articles/keystore.html#UserAuthentication. (Besøkt mars 2018).

[146] Apple. Januar 2018. App store. https://developer.apple.com/support/app-store/. (Besøkt april 2018).

[147] Finkle, J. September 2015. Apple's iOS app store suffers first major attack. https://www.reuters.com/article/us-apple-china-malware/apples-ios-app-store-suffers-first-major-attack-idUSKCN0RK0ZB20150920. (Besøkt april 2018).

[148] Newman, L. H. Desember 2016. Hack brief: Beware the spammy pokemon go apps being pushed to millions of iPhones. https://www.wired.com/2016/09/hack-brief-beware-spammy-pokemon-go-apps-pushed-millions-iphones/. (Besøkt april 2018).

[149] Bkav Corporation. November 2017. Bkav's new mask beats face id in twin way": Severity level raised, do not use face id in business transactions. http://www.bkav.com/dt/top-news/-/view_content/content/103968/bkav%EF%BF%BDs-new-mask-beats-face-id-in-twin-way-severity-level-raised-do-not-use-face-id-in-bus (Besøkt april 2018).

[150] Cox, J. Desember 2015. Why you don't roll your own crypto. https://motherboard.vice.com/en_us/article/wnx8nq/why-you-dont-roll-your-own-crypto. (Besøkt april 2018).

[151] Nasjonal sikkerhetsmyndighet. Mars 2015. Sikring av windows tls. https://www.nsm.stat.no/globalassets/dokumenter/veiledninger/systemteknisk-sikkerhet/u-03_sikring_av_windows_tls.pdf. (Besøkt mai 2018).

[152] Bernat, V. November 2011. Tls & perfect forward secrecy. https://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy. (Besøkt mai 2018).

[153] Electronic Frontier Foundation. Desember 2015. The eff ssl observatory. https://www.eff.org/observatory. (Besøkt april 2018).

[154] Hoffman, C. Februar 2014. 5 serious problems with https and ssl security on the web. https://www.howtogeek.com/182425/5-serious-problems-with-https-and-ssl-security-on-the-web/. (Besøkt april 2018).

[155] Turner, P. Oktober 2012. Ca compromise discussion: Preparing for and responding to certification authority compromise and fraudulent certificate issuance. https://csrc.nist.gov/csrc/media/projects/forum/documents/2012/october-2012_fcsm_pturner.pdf. (Besøkt april 2018).

[156] Galperin, E., Schoen, S., & Eckersley, P. September 2011. A post mortem on the iranian diginotar attack. https://www.eff.org/deeplinks/2011/09/post-mortem-iranian-diginotar-attack. (Besøkt april 2018).

[157] Electronic Frontier Foundation. Countries with CAs. https://www.eff.org/files/countries-with-cas.txt. (Besøkt april 2018).

[158] Wei, X. & Wolf, M. 2017. A survey on https implementation by android apps: Issues and countermeasures. *Applied Computing and Informatics*, 13(2), 101 – 117. URL: http://www.sciencedirect.com/science/article/pii/S2210832716300722, doi:https://doi.org/10.1016/j.aci.2016.10.001.

[159] Palmer, C. Oktober 2017. Intent to deprecate and remove: Public key pinning. https://groups.google.com/a/chromium.org/d/msg/blink-dev/he9tr7p3rZ8/eNMwKPmUBAAJ. (Besøkt mai 2018).

[160] Medley, J. April 2017. Deprecations and removals in chrome 67. https://developers.google.com/web/updates/2018/04/chrome-67-deps-rems. (Besøkt mai 2018).

[161] Apple Inc. Cocoa keys. https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW58. (Besøkt april 2018).

[162] Nolan, G. Mars 2015. Where is the best place to store a password in your android app. https://www.androidauthority.com/where-is-the-best-place-to-store-a-password-in-your-android-app-597197/. (Besøkt april 2018).

[163] Android Open Source Project. Security tips. https://developer.android.com/training/articles/security-tips.html#StoringData. (Besøkt mars 2018).

[164] Android Open Source Project. KeyStore.PasswordProtection. https://developer.android.com/reference/java/security/KeyStore.PasswordProtection.html. (Besøkt mars 2018).

[165] Apple Inc. applicationPassword. https://developer.apple.com/documentation/security/secaccesscontrolcreateflags/1617930-applicationpassword. (Besøkt mars 2018).

[166] Gilles. Januar 2011. How do i read from /proc/$pid/mem under linux? https://unix.stackexchange.com/a/6302. (Besøkt mai 2018).

[167] Android Open Source Project. Android Keystore System. https://developer.android.com/training/articles/keystore.html. (Besøkt mars 2018).

[168] Apple Inc. touchIDCurrentSet. https://developer.apple.com/documentation/security/secaccesscontrolcreateflags/1618004-touchidcurrentset. (Besøkt mars 2018).

[169] Apple Inc. userPresence. https://developer.apple.com/documentation/security/secaccesscontrolcreateflags/1392879-userpresence. (Besøkt mars 2018).

[170] Apple Inc. Storing Keys in the Secure Enclave. https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_secure_enclave. (Besøkt mars 2018).

[171] Android Open Source Project. Trusty tee. https://source.android.com/security/trusty/. (Besøkt april 2018).

[172] GlobalPlatform. Globalplatform made simple guide: Trusted execution environment (tee) guide. https://www.globalplatform.org/mediaguidetee.asp. (Besøkt april 2018).

[173] SAMSUNG. Device-side security: Samsung pay, TrustZone, and the TEE. http://developer.samsung.com/tech-insights/pay/device-side-security. (Besøkt april 2018).

[174] Rosenberg, D. August 2014. Reflections on trusting TrustZone. https://www.blackhat.com/docs/us-14/materials/us-14-Rosenberg-Reflections-on-Trusting-TrustZone.pdf#7. (Besøkt april 2018).

[175] ARM Limited. 2009. ARM security technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf#34. (Besøkt april 2018).

[176] ARM Limited. ARM trustzone. https://developer.arm.com/technologies/trustzone. (Besøkt april 2018).

[177] Wikipedia bidragsytere. 2018. ARM architecture. https://en.wikipedia.org/w/index.php?title=ARM_architecture&oldid=836864244#Security_extensions. (Besøkt april 2018).

[178] Mandt, T., Solnik, M., & Wang, D. August 2018. Demystifying the secure enclave processor. https://www.blackhat.com/docs/us-16/materials/us-16-Mandt-Demystifying-The-Secure-Enclave-Processor.pdf. (Besøkt april 2018).

[179] Smedsrud, A. B. Mars 2018. Får plass i lomma og åpner alle verdens iphoner. https://www.tek.no/artikler/far-plass-i-lomma-og-apner-alle-verdens-iphones/432916. (Besøkt mars 2018).

[180] Reed, T. Mars 2018. Graykey iphone unlocker poses serious security concerns. https://blog.malwarebytes.com/security-world/2018/03/graykey-iphone-unlocker-poses-serious-security-concerns/. (Besøkt mars 2018).

[181] ARM Limited. Tamper-resistant processors. https://www.arm.com/products/processors/securcore. (Besøkt april 2018).

[182] Shen, D. August 2015. Exploiting trustzone on android. https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android-wp.pdf. (Besøkt mars 2018).

[183] Beniamini, G. Juli 2017. Trust issues: Exploiting TrustZone TEEs. https://googleprojectzero.blogspot.no/2017/07/trust-issues-exploiting-trustzone-tees.html. (Besøkt april 2018).

[184] European Banking Association. Desember 2014. Final guidelines on the security of internet payments. https://www.eba.europa.eu/documents/10180/934179/EBA-GL-2014-12+%28Guidelines+on+the+security+of+internet+payments%29_Rev1. (Besøkt mars 2018).

[185] FIDO Alliance. Press room. https://fidoalliance.org/about/press-room/. (Besøkt april 2018).

[186] FIDO Alliance. Members: Bringing together an ecosystem. https://fidoalliance.org/participate/members-bringing-together-ecosystem/. (Besøkt april 2018).

[187] Hu, K. & Zhang, Z. December 2016. Security analysis of an attractive online authentication standard: Fido uaf protocol. *China Communications*, 13(12), 189–198. doi:10.1109/CC.2016.7897543.

[188] Bidragsytere til UAF rammeverket. UAF - Universal Authentication Framework. https://github.com/eBay/UAF/blob/master/README.md. (Besøkt Mars 2018).

[189] Android Open Source Project. Key Attestation. https://developer.android.com/training/articles/security-key-attestation.html. (Besøkt Mars 2018).

[190] Qualcomm Technologies, Inc. Mars 2015. Breakthrough 3d fingerprint authentication with snapdragon sense id. https://www.qualcomm.com/news/onq/2015/03/02/breakthrough-3d-fingerprint-authentication-snapdragon-sense-id. (Besøkt april 2018).

[191] Internet Engineering Steering Group. Mars 2018. Protocol Action: 'The Transport Layer Security (TLS) Protocol Version 1.3' to Proposed Standard (draft-ietf-tls-tls13-28.txt). https://www.ietf.org/mail-archive/web/ietf-announce/current/msg17592.html. (Besøkt Mars 2018).

[192] Sullivan, N. Mars 2017. Introducing Zero Round Trip Time Resumption (0-RTT). https://blog.cloudflare.com/introducing-0-rtt/#whatsthecatch. (Besøkt Mars 2018).

[193] Apple Inc. 2014. Transmitting Data Securely. https://developer.apple.com/library/content/documentation/Security/Conceptual/cryptoservices/SecureNetworkCommunicationAPIs/SecureNetworkCommunicationAPIs.html#//apple_ref/doc/uid/TP40011172-CH13-SW3. (Besøkt Mars 2018).

[194] Hellum, C., Nyman, H., & Hauger, K. K. Januar 2016. Apple har kontroll på «halve» Medie-Norge. https://kampanje.com/tech/2016/01/apple-har-kontroll-pa-halve-medie-norge/. (Besøkt Mars 2018).

[195] Grassi, P. A., Fenton, J. L., Newton, E. M., Perlner, R. A., Regenscheid, A. R., Burr, W. E., & Richer, J. P. Juni 2017. Digital Identity Guidelines. https://pages.nist.gov/800-63-3/sp800-63b.html#-5112-memorized-secret-verifiers. (Besøkt Mars 2018).

[196] Android Developers. Sharedpreferences. https://developer.android.com/reference/android/content/SharedPreferences. (Besøkt mars 2018).

[197] Android Open Source Project. Volley overview. https://developer.android.com/training/volley/. (Besøkt mars 2018).

[198] Android Developers. 2014. Improve your code with lint. https://developer.android.com/studio/write/lint. (Besøkt april 2018).

[199] OWASP bidragsytere. Juli 2017. Threat risk modeling: Cvss. https://www.owasp.org/index.php?title=Threat_Risk_Modeling&oldid=231638. (Besøkt april 2018).

A Ordforklaringer

Adware	Software som gir inntekt til utvikleren ved å vise frem reklame
Android	Mobilt operativsystem som er eid av Google
API	Application Programming Interface (programmeringsgrensesnitt). Et sett av definerte metoder for kommunikasjon mellom forskjellige komponenter
Bakdør	En metode for å forbipassere autentiseringen eller krypteringen i et datasystem
BankID	Elektronisk identifikasjon
Basestasjon	Radiosender som binder sammen telefonsentralen med mobiltelefoner
Biometri	Autentisering basert på noe man er, for eksempel et fingeravtrykk. Biologisk mønster.
Bootloader	Lite program som sørger for at operativsystemet blir lastet inn under oppstart
Brute-force angrep	Prøve hvert eneste passord frem til man finner det riktige
CAMEL	Signaliseringsprotokoll brukt ved roaming, og lar hjemmenettet følge anrop utført av abonnenten
Click-and-drag	Bevege elementer ved å klikke på de og flytte de til ønsket posisjon
Dekrypteringsnøkkel	Brukt for å endre data som er kryptert med tilhørende krypteringsnøkkel tilbake til klartekst
DHKE	Diffie-Helman Key Exchange. Brukes for å sikkert utveksle nøkler over et offentlig nettverk
Domene	Et område på internett
DRM	Digital Rights Management (digital rettighetsadministrasjon). Brukt for å begrense bruk av informasjon som er opphavsrettsbeskyttet
FaceID	System for ansiktsgjenkjenning utgitt av Apple
Falsk positiv	Falsk alarm
Hacker	I denne konteksten defineres hacker som en innbruddstyv som prøver å finne og utnytte sårbarheter for å få tilgang til konfidensiell informasjon eller systemer.
HAL	Maskinvareabstraksjonslaget. Et lag som emulerer platformspesifikke detaljer slik at man ikke må kommunisere direkte med maskinvaren
Hash	Informasjon som har blitt kjørt gjennom en matematisk funksjon som ikke kan reverseres.
Hendelseshåndteringsplan	En plan for hvordan man skal håndtere uønskede hendelser
HLR	Database som inneholder data om abonnentene (som for eksempel IMSI-nummeret).
HMAC	Hash-based Message Authentication Code. Brukes for å verifisere integritet og til autentisering.
HTTPS	En sikrere version av HTTP protokollen, laget for sikker kommunikasjon over et datanettverk.
IP	Internet Protocol. En protokoll som opererer på nettverkslaget for å rute datapakker.
IP-adresse	En unik identifikator som tildeles en enhet som bruker IP nettverket.
iOS	Operativsystem utviklet av Apple
Kaldstartsangrep	Angrep hvor angriperen har fysisk tilgang til datamaskinen og utfører en kaldstart

	for å uthente krypterinsnøklene. Dette er utførbart siden deler av minne er lesbart i noen sekunder etter datamaskinen er slått av.
Kernel	Kjernen til operativsystemet, med full kontroll over systemet
Kodebærer	En gjenstand eller tjeneste som gjør det mulig å hente ut engangskoder
Kryptering	Endrer klartekst ved en matematisk funksjon så bare personer med dekrypteringsnøkkelen kan lese det
Krypteringsnøkkel	Nøkkel brukt i krypteringsmetoden for å låseklarteksten
MAC	Adgangskontroll hvor operativsystemet bestemmer tilgangen brukeren har til et objekt
Man-in-the-middle-angrep	Et angrep hvor angriperen plasserer seg mellom to parter, og fanger opp informasjonen som blir sendt mellom de.
MAP	En protokoll i SS7 protokoll-settet som opererer på applikasjonslaget, og som noder i GSM-, UTMS- og GPRS-nettverk bruker for å kommunisere med hverandre.
Misbrukstilfelle	Metode for å finne sikkerhetskrav til en operasjon ved å modellere hva som kan gå galt
MSC	Mobile Switching Centre. Basestasjoner kobler seg opp mot et MSC, og all mobiltrafikk blir sendt gjennom MSC for switching.
Nedgraderingsangrep	Angrep hvor man tvinger systemet til å nedgradere til en tidligere versjon, for å finne flere sårbarheter som kan utnyttes
NFC-leser	En sensor som leser av NFC-brikker
Nonce	Et vilkårlig nummer som bare kan bli brukt én gang
Pakke-svitsjing	Gruppering av data inn i pakker med rutingsinformasjon som sendes over et nettverk
Phishing	Et angrep hvor angriperen utgir seg for å være en annen (for eksempel en bank) for å få målet til å gi fra seg sensitiv informasjon, som for eksempel et passord.
Policy	En type regler
Refleksjonsangrep	En form for tjenestenektangrep der man utnytter enkelte protokoller som svarer med mye mer data enn det mottas. Ved å sende forespørsler med offeret sin IP-adresse som avsenderavdresse til mange servere som benytter seg av en slik protokoll, vil store mengder data sendes til offeret.
Rainbow-table	En forhåndsberegnet tabell for å reversere hashfunksjoner
Replay-angrep	Et angrep hvor en gyldig dataoverføring blir sendt på nytt av en angriper for å kunne utgi seg for å være den originale senderen
Repository	Oppbevaringssted for kildekode
Reverse-engineering	Plukke fra hverandre et ferdig produkt for å finne alle detaljene på hvordan det fungerer
Roaming	Bruke et mobilnettverk utenfor hjemmenettets geografiske område
Rootet	En Android-enhet med mulighet til å starte applikasjoner med root-privilegier.
Root	Administrasjonsbrukerkontoen i unix-baserte systemer
Sandboxing	Sikkerhetsmekanisme som separerer kjørende programmer
Sertifikat	Elektronisk legitimasjon
Signering	Signere data til bruk som bevis på autentisitet
SS7	Et sett av signaliseringsprotokoller for telefonnettverket
SSO	Single Sign-On. Lar brukeren bruke ett sett av innloggingsdetaljer for å logge på flere applikasjoner
Systembuss	En komponent som kobler forskjellige komponenter sammen og frakter data mel-

	lom de
Systemprosess	Et program under utførelse i systemet
TEE	Trusted Execution Environment. Et sikkert miljø for operasjoner som må utføres adskilt fra det usikre miljøet.
Tjenestenektangrep	Et angrep hvor målet er å hindre brukere fra å få tilgang til systemer eller ressurser
TLS	Kryptografisk protokoll for sikker kommunikasjon på Internett
To-faktor autentisering	Autentiseringsmetode hvor man må oppgi to forskjellige bevis på at man eier kontoen som man prøver å logge inn på
ToS	Vilkår for bruk
TouchID	System for fingeravtrykksgjenkjenning utgitt av Apple
Trusselmodellering	Analysering av sikkerheten i en applikasjon, med fokus på truslene
Virtualisering	Bruk av en virtuell versjon av et system. Kan for eksempel brukes for å kjøre flere virtuelle operativsystemer på en maskin.

B Relevant kommunikasjon

B.1 E-post fra Henning Lunde

Henning Lunde, kommunikasjonsdirektør i Telia besvarte noen spørsmål rundt sikkerhet i Telia sitt nettverk. Svarene fra Lunde er lagt inn på samme linje som spørsmålene. Svaret på alle spørsmålene var altså "Ja".

Fra: henning.lunde@telia.no
Sendt: tirsdag 10. april 2018 kl. 11.39
Til: Sturla Høgdahl Bae
Emne: Re: Sikkerhet i mobilnettverket til Telia

Hei Sturla.

Det er nokså teknisk, komplisert og sensitivt, men dersom du kun ønsker det korte svaret så har jeg skrevet inn det under.

Mvh Henning

Fra: Sturla Høgdahl Bae <sturlaba@stud.ntnu.no>
Dato: fredag 6. april 2018 14.14
Til: Henning Lunde <henning.lunde@telia.no>
Emne: Sikkerhet i mobilnettverket til Telia

Hei,

Jeg jobber for tiden med en bacheloroppgave om sikkerhet i mobil infrastruktur og ville satt stor pris på om du kunne svare på noen spørsmål knyttet til sikkerheten i mobilnettet til Telia.

Jeg har kommet over to artikler der jeg er usikker på hvor godt sikret mobilnettet til Telia er sammenlignet med andre norske operatører. Den første er en artikkel publisert på digi.no den 14. mars 2016, der Telenor, Telia og Ice ble spurt om sikkerhet knyttet til SS7 (https://www.digi.no/artikler/ingen-kan-fullgodt-hindre-misbruk-av-sarbare-protokoller-fra-70-tallet/348218).

Kontaktpersonene fra både Telenor og Ice nevner at de har implementert eller implementerer tiltakene som anbefales av GSMA og Nkom, men ut i fra artikkelen ser det ikke ut til at Telia har innført noen slike tiltak. Derfor ønsker jeg gjerne å vite om Telia har implementert noen tiltak for å sikre mobilnettet sitt mot angrep over SS7. Jeg har full forståelse ovenfor at tiltakene kan være konfidensielle, så det går fint om du kun kan svare på det første spørsmålet.

- Har Telia implementert tiltak for å sikre mobilnettet sitt mot angrep over SS7? Ja
- Har Telia implementert tiltak som er foreslått fra GSMA og/eller Nkom? Ja

I tillegg vil jeg gjerne bekrefte om den planlagte implementasjonen av A5/3-kryptering har blitt Gjennomført. I en artikkel i Aftenposten 11. desember 2013 (https://www.aftenposten.no/norge/i/3jXKq/NSA-knekker-kryptering-som-hindrer-mobilavlytting) nevnes følgende: «*Netcom: Har bare A5/1 i 2G-nettet sitt, men planlegger å rulle ut A5/3-kryptering i 2G-nettet sitt fra og med januar 2014.»*

- Har A5/3-kryptering i 2G-nettet blitt rullet ut nå? Ja, ble fullført januar 2014.

Med vennlig hilsen,
Sturla Høgdahl Bae
Student ved Bachelor i Informasjonssikkerhet, Norges teknisk-naturvitenskapelige universitet

C CVSS Scores

C.1 CVSS Utregninger

- En sabotør kan infisere enheten til brukeren med ondsinnet programvare slik at applikasjonen ikke kan kjøres. CVSS 3,5: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:L/UI:R/S:U/C:N/I:N/A:L
- En hacker kan infisere en enhet med ondsinnet programvare og utnytte en sårbarhet i TEE og operativsystemet kan gjøre det mulig å hente ut private nøkler. CVSS 6,4: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:N
- En person med fysisk tilgang til den mobile enheten kan ta i bruk et forfalsket fingeravtrykk. CVSS 5,7: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:P/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N
- En hacker kan infisere en enheten med ondsinnet programvare og utnytte en sårbarhet som gir root-rettigheter for å endre programflyten i programmet og dermed utføre handlinger på vegne av brukeren. CVSS 6,4: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:N
- Den private nøkkelen eller CA-en som har signert sertifikatet kompromitteres. CVSS 8,4: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:L
- Programvarefeil kan føre til at systemet ikke fungerer eller fungerer på en uønsket måte. CVSS 5,9: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L
- Innloggingsdetaljene kan bli kapret ved hjelp av phishing. CVSS 8,3: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:L
- En bruker kan bli lurt til å laste ned en falsk applikasjon som etterligner originalen for å kapre innloggingsdetaljene. CVSS 7,1: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:L
- En hacker kan infisere en enhet med ondsinnet programvare som utnytter en sårbarhet som gir root-rettigheter for å få tak i innloggingsdetaljene. CVSS 7,1: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:L
- En sabotør kan gjennomføre et distribuert tjenestenektangrep mot serverene. CVSS 7,5: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
- En person med tilgang til datasenteret kan skru av serverene. CVSS 4,9: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:N/A:H
- En ansatt med muligheten til å endre koden kan legge inn en logisk bombe. CVSS 3,9: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:P/

AC:H/PR:H/UI:N/S:U/C:N/I:N/A:H

- En person med skriverettigheter til databasen kan endre den offentlige nøkkelen tilknyttet en bruker og selge tilgang til andre aktører. CVSS 5,8: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:P/AC:H/PR:H/UI:N/S:U/C:N/I:N/A:H
- En person med leserettigheter til databasen kan selge informasjon til andre aktører. CVSS 4,4: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N
- En person med tilgang til kildekoden kan legge inn ondsinnet kode eller en bakdør for å selge tilgang til andre aktører. CVSS 5,7: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:N
- En bruker kan nekte for å ha gjennomført en handling dersom vedkommende delte innloggingsdetaljene sine med andre. CVSS 6,5: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N
- En ansatt med tilgang til nettverket der webserveren og databaseserveren kommuniserer kan endre data i sanntid. CVSS 5,6: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:L/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:N

C.2 Metode for utregning av CVSS score

For å illustrere hvordan alvorlighetsgraden er beregnet har vi brukt en av truslene som eksempel.

"Innloggingsdetaljene kan bli kapret ved hjelp av phishing" har angrepsvektoren "nettverk" ettersom angriperen ikke trenger å være i nærheten av offeret. Angrepet har "lav" kompleksitet ettersom phishing er relativt enkelt å gjennomføre. Nødvendige privilegier er satt til "ingen" fordi man ikke trenger å ha noen privilegier til systemet for å gjennomføre et phising-angrep. For å kunne gjennomføre et phishing angrep er bruker-interasjon "påkrevd", men angriperen vil ikke få høyere privilegier enn privilegiene til offeret, og omfanget er derfor "uendret".

De siste faktorene består av hvordan angrepet påvirker konfidensialiteten, integriteten og tilgjengeligheten. Her har vi satt opp at angrepet har "høy" påvirkning på konfidensialitet og integritet ettersom angriperen får tilgang til all informasjon som er lagret om offeret i tillegg til mulighet til å endre på offerets detaljer. Påvirkningen på tilgjengeligheten satt vi til "lav" ettersom brukeren normalt vil kunne fortsette å benytte seg av tjenesten. En angriper kan endre innloggingsdetaljene og dermed hindre offeret i å benytte seg av tjenesten, men dersom dette skulle skje vil offeret trolig kontakte kundestøtte og få gjenopprettet full kontroll over kontoen.

Etter å ha angitt disse attributtene i NIST sin CVSSv3 kalkulator[1] var den resulterende "CVSS Base Score" på 8,3.

[1] NIST CVSSv3 kalkulator med utfylte vektorer: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:L

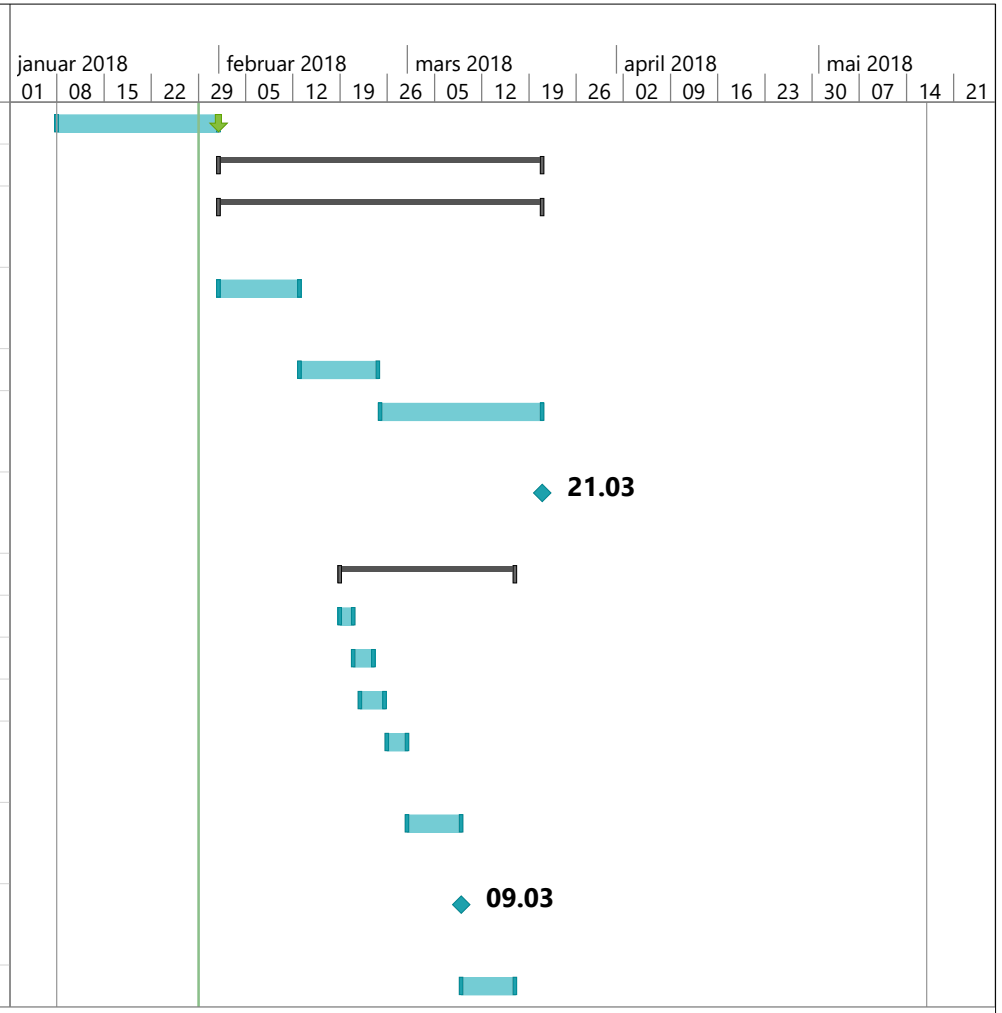
D GANTT og Prosjektplan

I dette vedlegget ligger GANTT diagramet og den originale prosjektplanen som ble levert inn ved begynnelsen av prosjektet. Planen har ikke blitt fulgt til punkt og prikke, og noen av grunnene til dette diskuteres i rapporten.

D.1 GANTT

Vi har laget et Gantt-skjemaet som illustrerer hvor mye tid vi skal bruke på de ulike oppgavene og når vi skal utføre de. Den gir en pekepinn på hvor langt vi burde ha kommet ved ulike datoer.

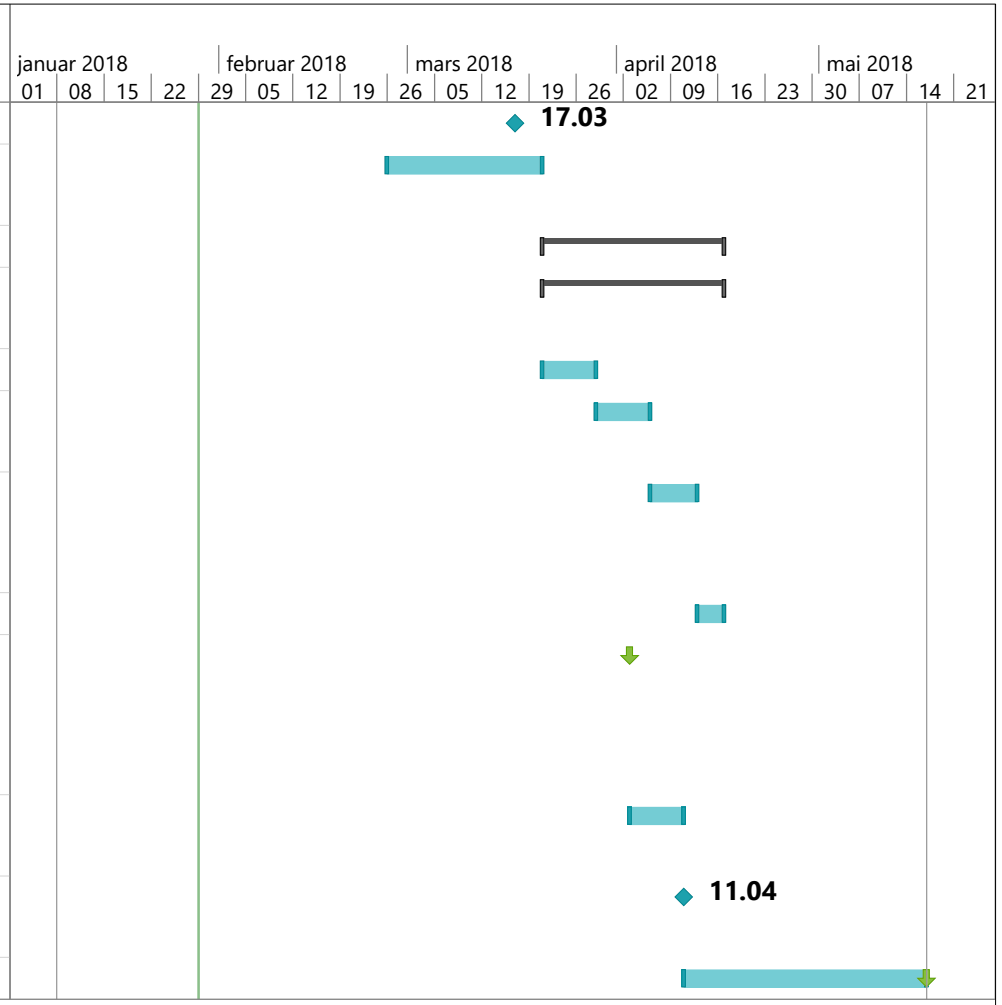
ID	Aktiv	Aktivitetsnavn	Varighet	Start	Slutt
1		Prosjektplanlegging	24 dager	ma 08.01.18	on 31.01.18
2		**Mobil autentisering**	**48 dager**	**to 01.02.18**	**on 21.03.18**
3		**Dypdykk i sikkerhet knyttet til mobil autentisering**	**48 dager**	**to 01.02.18**	**on 21.03.18**
4		Undersøke sikkerhet i Android	12 dager	to 01.02.18	ma 12.02.18
5		Undersøke sikkerhet i iOS	12 dager	ti 13.02.18	lø 24.02.18
6		Undersøke sikkerhet i hardware	24 dager	sø 25.02.18	ti 20.03.18
7		Ferdig med research om mobil autentisering	0 dager	on 21.03.18	on 21.03.18
8		**Applikasjonsutvikling**	**26 dager**	**ma 19.02.18**	**lø 17.03.18**
9		Kravspesifisering	2 dager	ma 19.02.18	ti 20.02.18
10		Design	3 dager	on 21.02.18	fr 23.02.18
11		Trusselmodellering	4 dager	to 22.02.18	sø 25.02.18
12		Implementasjon av back-end	3 dager	ma 26.02.18	on 28.02.18
13		Implementasjon av applikasjonen	8 dager	to 01.03.18	to 08.03.18
14		Ferdig med implementasjon	0 dager	fr 09.03.18	fr 09.03.18
15		Testing	8 dager	fr 09.03.18	fr 16.03.18



Prosjekt: Sikkerhet i mobil infrastruktur og autentisering

Aktivitet	Inaktivt sammendrag	Eksterne aktiviteter	
Deling	Manuell aktivitet	Ekstern milepæl	
Milepæl	Bare varighet	Tidsfrist	
Sammendrag	Manuell sammendragsfremheving	Fremdrift	
Prosjektsammendrag	Manuelt sammendrag	Manuell fremdrift	
Inaktiv aktivitet	Bare start		
Inaktiv milepæl	Bare slutt		

ID	Aktiv	Aktivitetsnavn	Varighet	Start	Slutt
16		Ferdig med applikasjon	0 dager	lø 17.03.18	lø 17.03.18
17		Testing av mobil autentisering	23 dager	ma 26.02.18	ti 20.03.18
18		**Mobil infrastruktur**	**27 dager**	**on 21.03.18**	**ma 16.04.18**
19		**Dypdykk i sikkerhet knyttet til mobil infrastruktur**	**27 dager**	**on 21.03.18**	**ma 16.04.18**
20		Sikkerhet knyttet til SS7	8 dager	on 21.03.18	on 28.03.18
21		Sikkerhet knyttet til IMSI-catchere	8 dager	to 29.03.18	to 05.04.18
22		Undersøke om det foreligger andre sårbarheter/angrep	7 dager	fr 06.04.18	to 12.04.18
23		Undersøk alternativ 2FA	4 dager	fr 13.04.18	ma 16.04.18
24		Frist for å avgjøre om det er lovlig/mulig å teste sikkerheten i infrastrukturen	0 dager	ma 02.04.18	ma 02.04.18
25		Testing av mobil infrastruktur?	8 dager	ti 03.04.18	ti 10.04.18
26		Ferdig med research om mobil infrastruktur	0 dager	on 11.04.18	on 11.04.18
27		Rapportskriving	36 dager	on 11.04.18	on 16.05.18

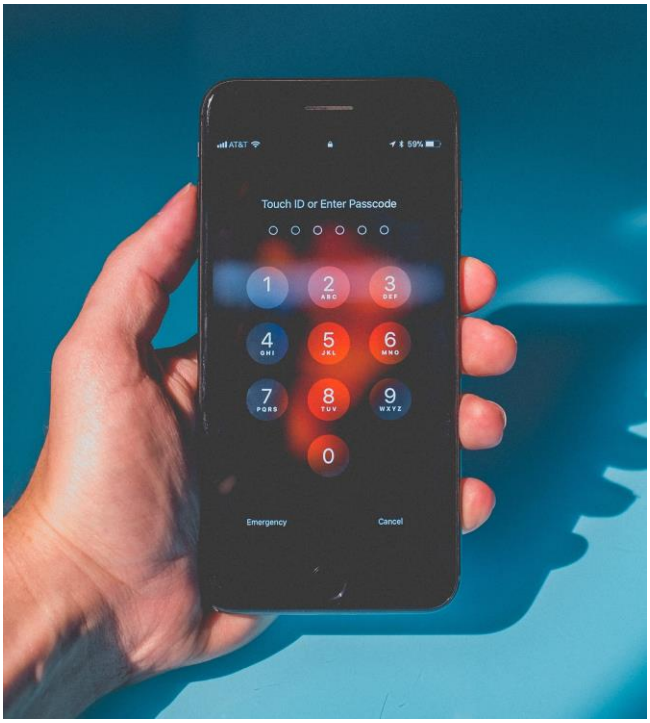


Prosjekt: Sikkerhet i mobil infrastruktur og autentisering

Aktivitet	Inaktivt sammendrag	Eksterne aktiviteter	
Deling	Manuell aktivitet	Ekstern milepæl	
Milepæl	Bare varighet	Tidsfrist	
Sammendrag	Manuell sammendragsfremheving	Fremdrift	
Prosjektsammendrag	Manuelt sammendrag	Manuell fremdrift	
Inaktiv aktivitet	Bare start		
Inaktiv milepæl	Bare slutt		

D.2 Prosjektplan

Prosjektplanen inneholder bakgrunnen, omfanget og planlegging av prosjektet.



PROSJEKTPLAN

Sikkerhet i mobil infrastruktur og mobil autentisering

SAMMENDRAG

I dette dokumentet beskrives bakgrunnen for oppgaven, hva som skal leveres, hvordan vi skal jobbe og en plan for fremdriften.

Henriette Kolby Rohde Garder,
Linn-Mari Kristiansen og
Sturla Høgdahl Bae

BIS3900 – Bacheloroppgave i Informasjonssikkerhet
Januar 2018

Innhold

1. Mål og rammer 2
1.1. Bakgrunn 2
1.2. Prosjektmål 2
1.2.1. Resultatmål 2
1.2.2. Effektmål 2
1.2.3. Læringsmål 3
1.3. Rammer 3
1.3.1. Rammer for del 1 3
1.3.2. Rammer for del 2 3
2. Omfang 4
2.1 Problemområde 4
2.2 Problemavgrensning 4
2.3 Problemstilling 5
3. Prosjektorganisering 5
3.1. Ansvarsforhold og roller 5
3.2. Rutiner og regler i gruppa 5
4. Planlegging, oppfølging og rapportering 6
4.1. Hovedinndeling av prosjektet 6
4.1.1. Prosesstyringsverktøy 6
4.2. Plan for statusmøter og beslutningspunkter i perioden 6
5. Organisering av kvalitetssikring 7
5.1. Dokumentasjon, standardbruk og kildekode 7
5.2. Konfigurasjonsstyring 7
5.3. Risikoanalyse (identifisere, analysere, tiltak, oppfølging) 7
5.3.1. Identifiserte risikoscenarioer 7
5.3.2. Risikomatrise: 8
5.3.3. Prioriterte risikoer og tiltak 8
5.3.3. Estimert resterende risiko 10
6. Plan for gjennomføring 10
6.1. Gantt-skjema 10
6.2. Liste over aktiviteter 11
6.3. Milepæler og beslutningspunkter 12
6.4. Tids- og ressursplan 12
Bibliografi 14

1. Mål og rammer

1.1. Bakgrunn

Mange innloggingssystemer i dag benytter seg av to-faktor autentisering. En populær metode å implementere dette på er ved hjelp av engangskode på SMS [1].

Vi har fått i oppdrag av Eika Gruppen AS å undersøke sikkerheten i mobil infrastruktur knyttet til bruken av disse engangskodene og sikkerheten i mobil autentisering, som innebærer analyse av programvaren og maskinvaren i mobile enheter. Alternativer til engangskode på SMS skal vurderes i forhold til dette, og angrep mot infrastrukturen skal skisseres for å gi best mulig oversikt over hvor sikker denne praksisen er og hvorfor.

Eika Gruppen AS er en del av Eika Alliansen og leverer produkter og tjenester til lokalbanker i Norge [2]. Som en leverandør av produkter og tjenester til banker er det viktig å ha oversikt over hvor sikre ulike autentiseringsmetoder er, og de ønsker derfor at vi skal undersøke sikkerheten til nettopp dette.

Det skal utvikles en applikasjon med sterk autentisering knyttet opp mot et serversystem, som et konseptbevis for sikker mobilautentisering. Sikkerheten i denne applikasjonen skal deretter analyseres ved hjelp av blant annet trusselmodellering for å identifisere eventuelle måter en angriper kan komme seg omkring autentiseringen.

1.2. Prosjektmål

Vi har delt målene til prosjektet inn i tre kategorier: Resultatmål, effektmål og læringsmål. Resultatmålene representerer hva vi skal ha ferdigstilt innen prosjektet er over, og beskriver sluttproduktet. Effektmålene representerer hva vi ønsker at prosjektet skal føre til i etterkant og læringsmålene representerer hva vi som gruppe ønsker å lære av dette prosjektet.

1.2.1. Resultatmål

- Prosjektet skal ende i en rapport som skal gi god oversikt over sikkerheten knyttet til mobil infrastruktur og mobil autentisering.
- Å utvikle en applikasjon som et konseptbevis for å vise hvordan man kan implementere sterk autentisering knyttet opp mot et serversystem.
- Applikasjonen skal implementere sikkert lagrede nøkler, fingeravtrykkslesing og ansiktsgjenkjenning.

1.2.2. Effektmål

- De som leser rapporten skal få en bedre forståelse av sikkerheten knyttet til bruk av SMS som to-faktor autentisering.
- Nye systemer som utvikles av oppdragsgiver og som implementerer to-faktor autentisering skal kun bruke sikre autentiseringsmetoder.

- De som leser rapporten skal få innsikt i hvordan man burde implementere mobil autentisering på sikrest mulig måte og hvordan man kan lagre nøkler sikkert på en mobil enhet.

1.2.3. Læringsmål

- Få økt forståelse for sikkerheten rundt mobil infrastruktur og mobil autentisering.
- Få innsikt i hvordan programvaren og maskinvaren på mobile enheter beskytter sensitiv informasjon.
- Lære å utvikle en mobil applikasjon til Android.
- Lære hvordan man kan lagre data på sikrest mulig måte på mobile enheter.
- Lære å bruke en utviklingsmodell til prosjektarbeid der utvikling ikke står sentralt
- Lære å bruke vitenskapelig metode i et større prosjekt.

1.3. Rammer

Oppdragsgiver er åpen for at gruppen kan bestemme mye selv når det gjelder hva de vil fokusere på, men har allikevel gitt noen rammer som prosjektet må følge.

1.3.1. Rammer for del 1

Første del av prosjektet går ut på å undersøke sikkerheten i mobil infrastruktur, med fokus på bruk av engangskoder på SMS som autentisering.

- Prosjektet skal undersøke sikkerheten basert på at:
    - Personer reiser til andre land (roaming)
    - Personer har ulike modeller fra ulike produsenter (Apple, Samsung, Sony etc.)
    - Noen har nye mobiltelefoner og noen har eldre mobiltelefoner
- Prosjektet skal vurdere sikkerheten til andre to-faktor autentiseringsmetoder og deretter til å sammenligne sikkerheten til disse metodene med SMS, men fokuset skal allikevel være på SMS.

1.3.2. Rammer for del 2

Den andre delen går ut på å se på autentiserings metoder på mobiltelefoner, og vurdere sikkerheten knyttet til disse.

- Prosjektet trenger ikke evaluere om sikkerhetsfunksjonene er korrekt implementert.
- Prosjektet skal undersøke hvordan standarder er implementert ulikt hos ulike produsenter, men skal kun ta for seg de mest vanlige produsentene.
- Applikasjonen skal utvikles til iOS eller Android.
- Det skal utføres praktiske tester for å vise eventuelle sårbarheter i praksis.

2. Omfang

Omfanget av oppgaven beskriver hva oppgaven prøver å løse. Dette innebærer å definere problemområde og å avgrense problemet til en oppgave vi kan løse i løpet av prosjektperioden.

2.1 Problemområde

Mobiltelefoner har blitt en naturlig del av hverdagen til de aller fleste nordmenn, og det har blitt godt integrert i samfunnet. De brukes til å kommunisere, betale regninger, kjøpe og oppbevare billetter, kart og mye mer. Mange nordmenn er flinke til å installere antivirus-programmer og er forsiktige med hva de laster ned på datamaskinene sine, men færre tenker på sikkerheten til smarttelefonen sin [3].

Ettersom det har blitt vanlig å bruke mobiltelefon til alle de overnevnte funksjonene blir man nødt til å stille seg et spørsmål om sikkerheten i programvaren og maskinvaren er tilstrekkelig for å hindre andre i å få tak i sensitiv informasjon. Hvordan vet man at ikke en applikasjon kan få tak i passordet ditt når man logger inn et annet sted på mobiltelefonen?

To-faktor autentisering er en voksende trend. Google har rapportert at de har 1,2 milliarder registrerte Gmail brukere, men mindre enn 10% av disse har tatt i bruk to-faktor autentisering [4]. Stadig flere tjenester tar i bruk to-faktor, men likevel er det få som benytter det, og det er mange som ikke er klar over muligheten.

Ved bruk av to-faktor autentisering skal det mye mer til for at en potensiell angriper skal kunne ta kontroll over noen andres konto, og etter som SMS er noe folk flest har tilgang til [5], er dette en metode som brukes på mange nettsider den dag i dag [1].

Man kan fort tenke at man er fullstendig beskyttet når man trenger en SMS på sin egen telefon for å logge inn, men er det virkelig bare senderen og mottakeren som kan lese innholdet i en SMS?

2.2 Problemavgrensning

Siden oppgaven sier lite om hvor dypt vi skal undersøke sikkerheten, må vi avgrense problemet for at oppgaven ikke skal bli for stor.

I tillegg til rammene som er satt fra oppdragsgiver har vi valgt å avgrense prosjektet på følgende måte:

Ettersom sikkerhet i mobil arkitektur er et veldig vidt tema, skal vi kun undersøke sikkerheten i programvare og maskinvare som kan knyttes til autentisering. Dette vil for eksempel innebære sikkerhet knyttet til lagring av hemmelige nøkler fordi man kan bruke lagrede nøkler til autentisering.

Analysen av programvaresikkerheten skal ikke gå like dypt for iOS som Android, ettersom Android har åpen kildekode og iOS har lukket kildekode. Dersom det ikke er mulig å få tak i

tilstrekkelig informasjon om sikkerheten tilknyttet en funksjon, skal vi heller ikke analysere den gjeldende sikkerheten, slik at vi unngår å måtte gjøre antakelser.

2.3 Problemstilling

Hvor sikkert er det å bruke engangskoder på SMS til autentisering, og hvor sikker er moderne autentiseringsmetoder på mobile enheter?

3. Prosjektorganisering

Oppgaven skal utføres i en gruppe på tre personer. Vi trenger derfor å definere ansvar, roller or regler slik at gruppearbeidet er så effektivt som mulig.

3.1. Ansvarsforhold og roller

Alle gruppemedlemmene er ansvarlige for at prosjektet gjennomføres. Videre må alle gruppemedlemmene overholde punktene som står skrevet i prosjektavtalen og grupperegelene.

- Gruppeleder: Sturla Høgdahl Bae
    - Ansvarlig for:
        - At diskusjoner holdes saklige.
        - At alle har oppgaver de kan gjøre.
        - Å ta valg dersom det er uenighet i gruppen.
- Nestleder: Linn-Mari Kristiansen
    - Påtar seg ansvaret til gruppeleder dersom gruppeleder skulle være utilgjengelig.
- Sekretær: Henriette Kolby Rohde Garder
    - Ansvarlig for:
        - At møteagendaen blir gått gjennom hvert møte.
        - At møtereferat blir skrevet hvert møte.

3.2. Rutiner og regler i gruppa

Hvert møte skal gruppen gjøre følgende:

- Bestill nytt grupperom for 2 uker frem i tid.
- Gå igjennom oppgaver i prosjektstyringsverktøy.
- Last ned ShareLaTeX-prosjekt og last opp i gruppens skylagringsmappe.

Se eget dokument for grupperegler.

Første gruppemøte hver uke skal all data sikkerhetskopieres til en minnepinne eller ekstern harddisk.

4. Planlegging, oppfølging og rapportering

4.1. Hovedinndeling av prosjektet

Helt fra starten av prosjektet har oppdragsgiver vært veldig åpen for hva gruppen skal fokusere på. Det er opp til oss å avgrense problemområdet og vurdere hvor dypt vi skal undersøke sikkerheten på de ulike områdene. Siden vi ikke har klare, definerte oppgaver som vi vet omtrent hvor lang tid det vil ta å fullføre, vil det være vanskelig å bruke en lineær modell, som for eksempel Fossefall. Det kan også være at det underveis i prosjektet dukker opp nye ting tilknyttet mobil infrastruktur eller autentisering som vi vil undersøke.

4.1.1. Prosesstyringsverktøy

Vi har valgt å bruke en smidig tilnærming til oppgaven, og Kanban ble valgt som prosesstyringsverktøy. Trello blir brukt som Kanban brett, hvor vi fordeler oppgaver og får en visuell presentasjon av fremdriften. Siden gruppen er kjent med hverandre fra før og har et felles ambisjonsnivå så vil det ikke være noe problem å ikke sette harde frister for når ulike ting skal være ferdig, så Kanban sin "pull" metodikk passer bra for oss. Dersom vi oppdager at en oppgave tar lengre tid enn antatt, fortsetter vi å jobbe med den til den er ferdig slik at vi kan bli helt ferdig med oppgaven. Dermed unngår vi å bruke ekstra tid på å gå frem og tilbake mellom oppgaver. Under hvert møte blir kvaliteten på de utførte oppgavene kontrollert, og satt som ferdig hvis de passerer. Fremdriften blir vurdert opp mot et Gantt-skjema for å vurdere hvilke oppgaver som må settes opp videre.

En egenskap fra Kanban som vi ikke tar i bruk er begrenset work in progress. Vi deler opp oppgavekortene så mye vi kan for å ha best mulig visualisering av arbeidet, og dette kan bli mange små oppgavekort som varierer i størrelse. Vi tilpasser heller ved hjelp av tidligere erfaringer, og skulle ikke dette stemme tilpasser vi oppgavene på nytt.

4.2. Plan for statusmøter og beslutningspunkter i perioden

Gruppen skal ha statusmøte med oppdragsgiver hver andre uke, men oppdragsgiver var klar på at møtene kan være fleksible. Dersom det trengs møter oftere, eller ikke trengs møte en uke, er det enkelt å løse.

To uker mellom hvert møte gir gruppen nok tid til å utarbeide noen resultater som kan presenteres mellom hver gang, samtidig som det åpner for at oppdragsgiver jevnlig kan komme med innspill til hvor de ønsker at gruppen skal fokusere videre.

Det skal være et møte med veileder hver uke, hvor arbeidet som har blitt gjort blir diskutert sammen med veien videre.

5. Organisering av kvalitetssikring

5.1. Dokumentasjon, standardbruk og kildekode

Alle funn og idéer skal dokumenteres skriftlig slik at de ikke blir glemt.

Applikasjonen skal dokumenteres med Javadoc for å sikre lesbarhet og at andre som leser kildekoden enkelt kan implementere tilsvarende sikker autentisering. Koden som blir skrevet skal også alltid være kommentert.

Kildekoden til applikasjonen skal være åpen.

5.2. Konfigurasjonsstyring

Kildekoden skal lagres i et git repository. Utviklerne skal pushe ferdige endringer til felles utviklingsgren omtrent hver time de jobber, slik at man ikke ender opp med å måtte bruke lang tid på å flette sammen endringer.

5.3. Risikoanalyse (identifisere, analysere, tiltak, oppfølging)

For å redusere risikoer tilknyttet oppgaven har vi gjennomført en kortfattet risikoanalyse og behandling av de største risikoene.

5.3.1. Identifiserte risikoscenarioer

Teknologi

A. Nødvendig testutstyr bestilt fra Asia blir ikke levert i tide.
B. Samtlige gruppemedlemmer får løsepengevirus som krypterer data på disk og i nettskyen nær innleveringsfristen.
C. ShareLaTeX prosjekt med alt vi har skrevet blir slettet eller utilgjengelig frem til etter innleveringsfristen.

Forretningsmessig

D. Sluttresultatet tilfredsstiller ikke kravene til oppdragsgiver.
E. Rapporten inneholder feilinformasjon og villeder leserne av rapporten som igjen fører til at det gjøres dårlige beslutninger hos oppdragsgiver.

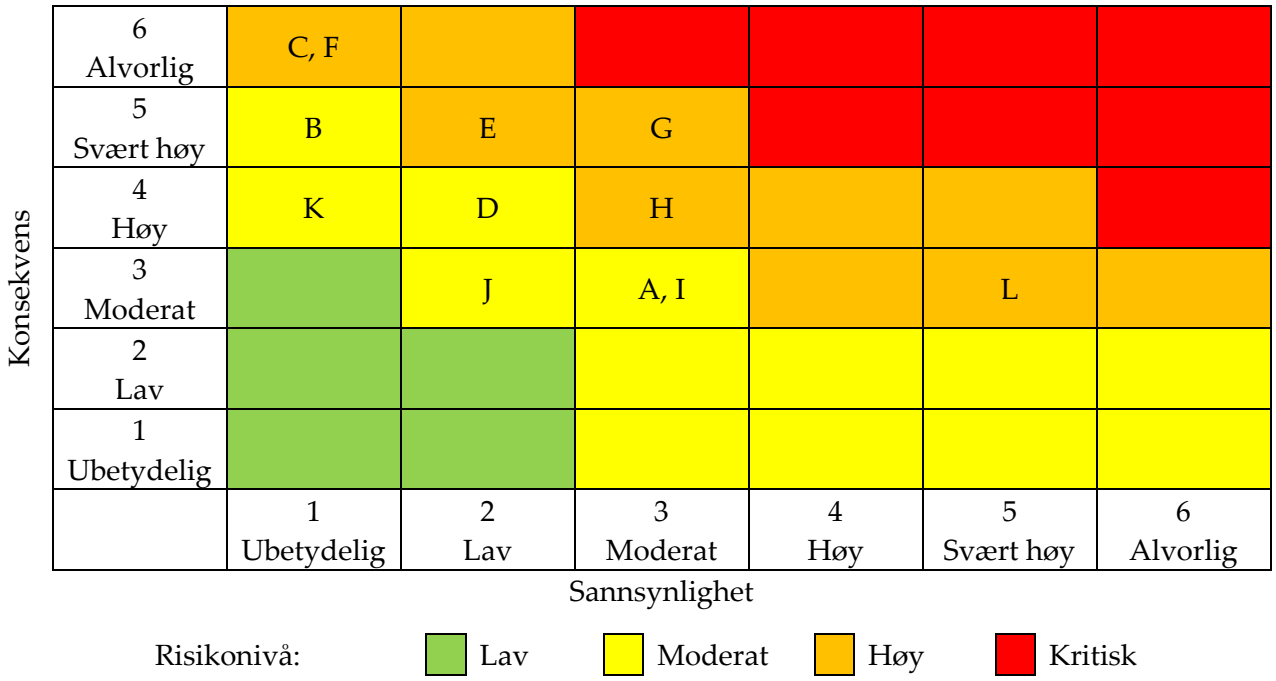
Prosjektrisiko

F. Noen i gruppen blir alvorlig syk eller dør, slik at de ikke kan delta i arbeidet.
G. Testingen i prosjektet bryter med en norsk lov.

H. Gruppen ender opp med å bruke for lang tid på en del av oppgaven, og rekker ikke dekke hele oppgaven fra oppdragsgiver.
I. En del av oppgaven har lavere kvalitet enn resten av oppgaven og trekker ned resultatet.
J. Epost til eller fra oppdragsgiver når ikke frem og fører til at spørsmål ikke blir besvart eller at viktig informasjon går tapt.
K. Uenigheter i gruppen fører til at dårlig arbeidsmoral og dårlig resultat.
L. Misforståelser mellom gruppen og veileder eller oppdragsgiver fører til dårligere kvalitet på oppgaven.

5.3.2. Risikomatrise:

Risikomatrisen brukes til å identifisere hvor høy risikoen er for de ulike scenarioene [6].



5.3.3. Prioriterte risikoer og tiltak

Fordi bacheloroppgaven er et såpass viktig prosjekt, har vi valgt å behandle alle risikoscenarioer som har moderat eller høyere risikonivå.

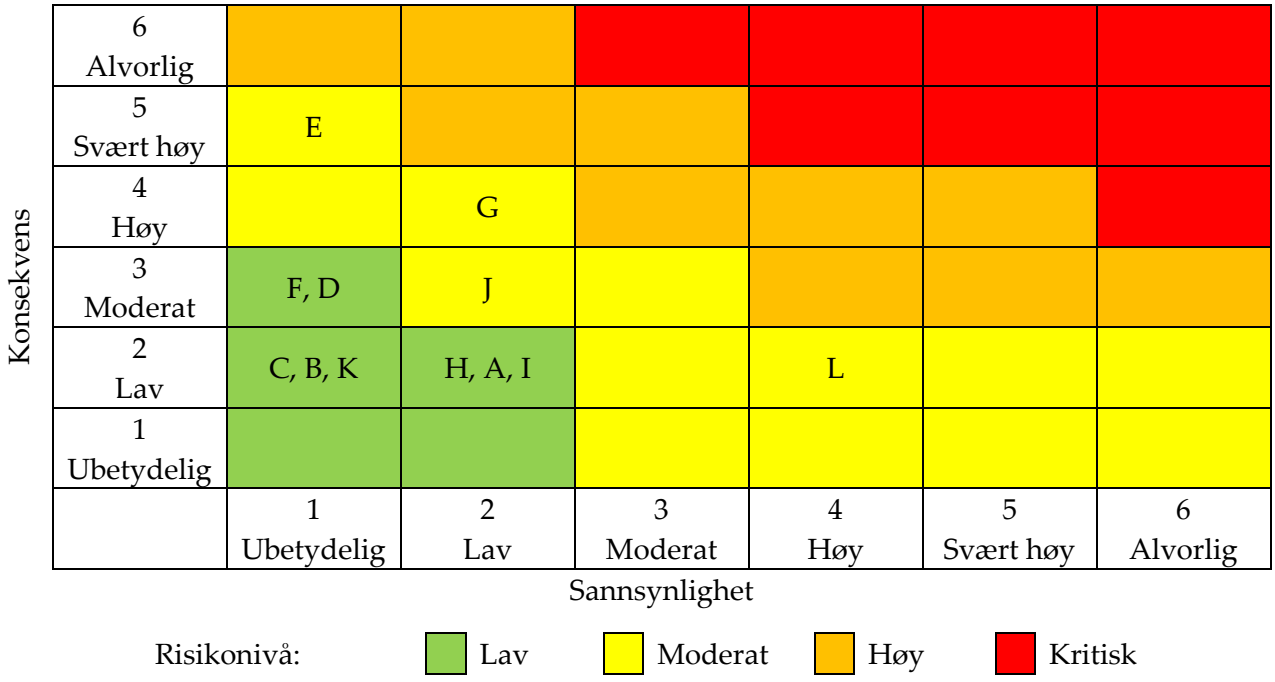
Risikoscenario	Tiltak
G. Testingen i prosjektet bryter med en norsk lov.	Minst en av gruppemedlemmene leser seg opp på norske lover eller kontakter relevante instanser rundt sniffing av mobile signaler (uten dekryptering). Få veileder til å bekrefte at testingen er tillatt.
L. Misforståelser mellom gruppen og veileder eller oppdragsgiver	Dersom det er noen usikkerheter, prøv å få det oppklart.

fører til dårligere kvalitet på oppgaven.	Ha mye skriftlig kommunikasjon slik at man kan dokumentere hva som har blitt sagt. Skriv møtereferater som forklarer hva som ble bestemt.
H. Gruppen ender opp med å bruke for lang tid på en del av oppgaven, og rekker ikke dekke hele oppgaven fra oppdragsgiver	Lag plan for gjennomføring og følg planen i størst mulig grad. Dersom en oppgave tar lengre tid enn antatt, må man enten redusere omfanget av den gjeldende oppgaven eller sette av mer tid til å arbeide med prosjektet. Står man fast, gå videre. Diskuter det man står fast på med oppdragsgiver eller veileder.
E. Rapporten inneholder feilinformasjon og villeder leserne av rapporten som igjen fører til at det gjøres dårlige beslutninger hos oppdragsgiver.	Vær kritisk til alle kilder. Sjekk om andre kilder kan bekrefte funn. Diskuter resultatene med veileder og andre professorer.
C. ShareLaTeX prosjekt med alt vi har skrevet blir slettet eller utilgjengelig frem til etter innleveringsfristen.	Synkroniser ShareLaTeX prosjekt med Dropbox slik at all data også ligger i Dropbox. Last ned ShareLaTeX prosjekt minst en gang i uken og sikkerhetskopier sammen med annen data.
F. Noen i gruppen blir alvorlig syk eller dør, slik at de ikke kan delta i arbeidet.	Få utsatt innleveringsfristen dersom noe slikt skulle skje: Send søknad om utsettelse og dokumentasjon til eksamen@gjovik.ntnu.no (jfr. "Bacheloroppgaver, utfyllende informasjon" punkt 2.3). Dersom noe slikt skjer tidlig i prosjektperioden, kontakt oppdragsgiver og spør om å få redusert omfanget av oppgaven.
B. Samtlige gruppemedlemmer får løsepengevirus som krypterer data på disk og i nettskyen.	Last opp dokumentet på flere nettskyer som sikkerhetskopi og ta sikkerhetskopi av all data til minnepinne eller ekstern harddisk minst en gang i uken.
A. Nødvendig testutstyr bestilt fra Asia blir ikke levert i tide	Bestill utstyr tidlig i prosjektet. Oppdragsgiver har ikke bedt om at sikkerheten skal testes praktisk; Å emulere et miljø vil også underbygge teorien. Ha alternativer klare.
I. En del av oppgaven har lavere kvalitet enn resten av oppgaven og trekker ned resultatet.	Presenter og diskuter resultatene med veileder, og gjøre en kvalitetssjekk på arbeidet som har blitt gjort før hvert møte.
D. Sluttresultatet tilfredsstiller ikke kravene til oppdragsgiver.	Sterk kommunikasjon med oppdragsgiver. Vi må alltid være enige om hva vi skal frem til og oppdragsgiver skal bli oppdatert på hva vi gjør.
K. Uenigheter i gruppen fører til at dårlig arbeidsmoral og dårlig resultat.	Lag grupperegler som kan brukes til å forebygge konflikter. Diskuter problemet med veileder dersom gruppen ikke klarer å løse konflikten selv.
J. Epost til eller fra oppdragsgiver når ikke frem og fører til at	Både oppdragsgiver og veileder har vært veldig raske til å svare på alle e-poster. Dersom de ikke

spørsmål ikke blir besvart eller at viktig informasjon går tapt.	svarer innen 4 arbeidsdager, kontakt vedkommende via telefon.

5.3.3. Estimert resterende risiko

Dersom de foreslåtte tiltakene implementeres, antar vi at den resterende risikoen vil være som følgende:



6. Plan for gjennomføring

6.1. Gantt-skjema

Vi har laget et Gantt-skjemaet som illustrerer hvor mye tid vi skal bruke på de ulike oppgavene og når vi skal utføre de. Den gir en pekepinn på hvor langt vi burde ha kommet ved ulike datoer, og så lenge vi følger planen vil vi dermed rekke å ferdigstille oppgaven innen fristen.

6.2. Liste over aktiviteter

- Prosjektplanlegging
    - Lese tidligere bacheloroppgaver
    - Lage regler og rutiner for gruppen.
    - Identifisere mål, rammer og omfang for prosjektet
    - Generell undersøkelse av juridiske begrensninger
    - Gjennomføre risikoanalyse
    - Skrive prosjektplan
- Litteraturstudie
    - Skaffe oversikt over hvordan mobile enheter er sikret.
    - Lese om sikkerhet implementert i Android og iOS.
    - Lese om hvordan sikkerhet er implementert i hardware på mobile enheter.
    - Dypdykk i sikkerhet knyttet til mobil autentisering.
    - Lese om svakheter i og angrep mot mobilautentisering.
    - Dypdykk i sikkerhet knyttet til mobil infrastruktur.
    - Lese om svakheter i og angrep mot infrastruktur.
    - Standarder
- Applikasjonsutvikling
    - Lære Android applikasjonsutvikling
    - Kravspesifisering
    - Design
    - Trusselmodellering
    - Implementasjon av back-end
    - Implementasjon av applikasjon
    - Testing av applikasjon
- Testing av mobil infrastruktur
    - Konseptbevis
    - Penetrasjonstesting
- Testing av mobil sikkerhet
    - Konseptbevis
    - Penetrasjonstesting
- Rapportskriving
    - Kravspesifisering
    - Førskriving (forberedelse)
    - Skrive utkast
        - Problemstilling
            - Bakgrunn for valgt problemstilling
            - Problemstillingen med begrunnelse
            - Formålet med prosjektet
            - Generell innføring i temaet med oversikt over tidligere forskning
        - Avgrensning

            - Rammer for arbeidet (fysiske, tidsmessige, økonomiske, juridiske)
        - Teori
        - Argumentasjon
            - Analyse
            - Diskusjon
                - Funnene sammenlignes med egne hypoteser og andre studier.
                - Stille kritiske spørsmål til egne resultater for å avsløre svakheter i materialet, metoden eller analysen.
        - Konklusjon
            - Trådene samles
            - Anbefalinger
            - Forslag til videre forskning på emnet?
    - Omskrive (forbedring av utkast)

6.3. Milepæler og beslutningspunkter

Vi har satt opp følgende milepæler og beslutningspunkter i Gantt-skjemaet:

- Vi skal være ferdig med å undersøke sikkerheten tilknyttet mobil autentisering innen den 21. mars. Dette gir oss omtrent 48 dager til å jobbe med mobil autentisering.
- Applikasjonen skal være ferdig implementert den 9. mars, og hele applikasjonsutviklingen (inkludert testing) skal være ferdig den 17. mars.
- Fristen for at vi skal ha bestemt om vi skal gjennomføre en test på mobil infrastruktur er den 2. april. Om vi skal gjennomføre en test av sikkerheten i mobil infrastruktur kommer an på om det er noen relevante tester som også er lovlig og mulig for oss å gjennomføre. Fristen er satt til den 2. april for at vi skal ha tid til å teste uten at det går utover tiden til å skrive på rapporten.
- Videre skal vi være ferdig med å undersøke sikkerheten i mobil infrastruktur den 11. mars.
- Den endelige fristen for å være ferdig med rapporten og prosjektet er den 16. mai.

6.4. Tids- og ressursplan

På grunn av metodikken vi jobber med (Kanban), så vil ikke tidsplanen bli fastbestemt før oppgavene er i gang. Det vil fort oppstå endringer i hvor mye tid de ulike personene bruker på ulike oppgaver.

Vi har allikevel gjort et estimat for hvor mange timer hver av oss skal bruke på de ulike oppgavene. Estimatene baserer seg på hva vi har satt opp som parallelle oppgaver i Gantt-skjemaet og hvilke kunnskaper de ulike gruppemedlemmene besitter fra før av.

Tidsplanen er fleksibel, og bare en veiledning for omtrent hvor mye vi burde jobbe med hver oppgave for å få dekket alt. Etter hvert som vi jobber kan det være at vi oppdager at noen oppgaver tar lengre tid enn estimert, og noen kan ta kortere tid.

	Dypdykk mobil autentisering	Applikasjons-utvikling	Testing av mobil autentisering	Dypdykk mobil infra-struktur	Testing av mobil infrastruktur	Rapport-skriving
Henriette	176 t	30 t		66 t	29 t	176 t
Linn-Mari	110 t	56 t	40 t	66 t	29 t	176 t
Sturla	110 t	37 t	60 t	106 t	10 t	154 t

Bibliografi

[1] J. Davis, «Two Factor Auth,» [Internett]. URL: https://twofactorauth.org/. [Funnet 20 Januar 2018].

[2] Eika Gruppen AS, «Om oss - Eika,» [Internett]. URL: https://www2.eika.no/eikagruppen/om-oss. [Funnet 22 Januar 2018].

[3] Telia, «Hvor trygg er mobilen din – egentlig?,» [Internett]. URL: https://e24.no/betalt-innhold/bak-tallene/hvor-trygg-er-mobilen-din-egentlig/23700089. [Funnet 22 Januar 2018].

[4] I. Thomson, «The Register,» 17 Januar 2018. [Internett]. URL: http://www.theregister.co.uk/2018/01/17/no_one_uses_two_factor_authentication/. [Funnet 19 Januar 2018].

[5] T. Worstall, «Forbes,» 23 Mars 2013. [Internett]. URL: https://www.forbes.com/sites/timworstall/2013/03/23/more-people-have-mobile-phones-than-toilets/. [Funnet 20 Januar 2018].

[6] M. E. Whitman og H. J. Mattord, «Principles of Information Security,» Boston, Cengage Learning, 2014, p. 267.

E Prosjektavtale

Den vedlagte prosjektavtalen beskriver pliktene og rettighetene til oppdragsgiveren og studentene.

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Eika Gruppen AS

(oppdragsgiver), og

Henriette Kolby Rohde Gardar

Linn-Mari Kristiansen

Sturla Høgdahl Bae (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra januar 2018 til juni 2018 .

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Basel Katt

Oppdragsgivers kontaktperson (navn): Jon Hagen

Student(er) (signatur): Linn-Mari Kristiansen dato 09-01-18

Henriette Kolbjørnsrud Gaarder dato 09-01-18

Sturla H. Bae dato 9-01-18

dato

Oppdragsgiver (signatur): Thomas Eriksen dato 09-01-18

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): dato

F Møtereferater

Vi har ført opp møtereferat for alle møter der beslutninger ble tatt eller sentrale deler av oppgaven ble diskutert.

08.01.2018

- Startet med planlegging av hva som burde være på plass innen første uke.
    - Enighet om å lese igjennom oppgaven fra oppdragsgiver, informasjon om bacheloroppgaven, prosjektavtalen og metoderapport om mobilovervåkingen.
- Bestemt at rapporten skal skrives i ShareLaTeX.
- Laget en Word-mal for alle dokumenter som ikke er selve rapporten.
    - Inkluderer hvilke skrifttyper som skal brukes.
        - Palatino Linotype for brødtekst
        - Segoe UI for overskrifter
- Planlagt utførelse av rombestilling
    - Skal bestilles rom for 2 uker frem i tid i løpet av møtet.
- Skrive ned spørsmål til møte med oppdragsgiver (09/01-2018).
- Planlagt (fleksible) møter for arbeidsgruppa
    - Mandag 10:15 – 14:00
    - Tirsdag 11:05 – 14:00
    - Onsdag 10:15 – 13:00
    - Torsdag 10:15 – 14:00 (bortsett fra helger der noen skal reise bort)
- Valgt Trello som prosjektstyringsverktøy.
- Bestemt at alle dokumenter lagres i felles mappe i OneDrive
- Blitt enige om at alt skal sikkerhetskopieres jevnlig og at man skal teste at det er mulig å gjenopprette fra kopiene.

09.01.2018

- Møte med oppdragsgiver.
    - Gjennomgang av Jon sine tanker
    - Gjennomgang av spørsmål til oppgaven

10.01.2018

- Funnet relevante artikler som burde leses
- Valgt gruppeleder: Sturla
- Kontaktet veileder og avtalte møtetider
- Deltok på lynkurs om prosjektarbeid
- Bestemt språk for oppgaven: Norsk

11.01.2018

- Bestemt hva som skal tas med i loggbok: Antall timer møte, antall timer lesing (og hva man har lest på) og antall timer skriving (og hva man har skrevet på). Hva som blir gjort under møter føres opp under møtereferater.
- Skrevet grupperegler
- Signert grupperegler
- Valgt nestleder: Linn-Mari
- Valgt sekretær: Henriette
- Laget mal for timeføring
- Laget struktur for møtereferater
- Funnet relevante artikler som burde leses
- Bearbeidet og ryddet opp i notater fra møte med oppdragsgiver
- Funnet frem bachelor-oppgaver fra tidligere år som skal leses
- Diskutert utviklingsmodell

15.01.2018

- Gått gjennom oppgavene til i dag
- Laget dokument for prosjektplan
- Skrevet bakgrunn, mål og rammer for prosjektplan.

16.01.2018

- Forberedt spørsmål til veileder
- Skrevet om omfang, prosjektorganisering, planlegging, oppfølging og rapportering, organisering av kvalitetssikring og plan for gjennomføring på prosjektplanen.

17.01.2018 - møte med veileder

- Gjennomgang av møte med Eika
- Diskuterte lover og regler i forhold til parktiske tester

17.01.2018

- Bestilt antenne for å sniffe 900 MHz GSM trafikk
- Bearbeidet notater fra møte med veileder
- Lastet opp signert prosjektavtale til Blackboard

22.01.2018

- Kontaktet oppdragsgiver og fått besvart spørsmål
- Fullført mål og rammer på prosjektplanen
- Fullført omfang på prosjektplanen

23.01.2018

- Skrevet på prosjektplan
- Gjennomført risikoanalyse

24.01.2018 - møte med veileder

- Diskuterte forslag til praktiske tester
- Diskutert hva som ikke er lovlig

24.01.2018

- Diskutert Android sikkerhet
- Estimert og ført opp restrisiko på prosjektplanen

25.01.2018

- Lage liste over aktiviteter
- Brutt ned aktiviteter i mindre oppgaver
- Laget utkast av Gantt-skjema

29.01.2018

- Fulført utkast av Gantt-skjema
- Funnet relevante artikler om mobilsikkerhet

30.01.2018

- Laget tids- og ressursplan
- Forberedt møte med veileder

31.01.2018

- Gjennomgå spørsmål til veileder
- Sette opp OneNote notatblokk for samling av notater
- Kontaktet oppdragsgiver(Eika) ang å avtale nytt møte

31.01.2018 - møte med veileder

- Rask gjennomgang av spørsmålene til Eika

05.02.2018

- Planlegge nye oppgaver
- Forberede møte med oppdragsgiver (06/02)
- Diskutert veien videre

06.02.2018 - møte med oppdragsgiver

- Diskutere applikasjonen
- Diskutere biometriske autentiseringsmetoder

07.02.2018

- Forberedt møte med veileder
- Funnet relevant lesestoff
- Distribuert oppgaver

07.02.2018 - møte med veileder

- Diskutere hvor man kan finne vitenskapelige artikler
- Diskutere litt om mulige tester

12.02.2018

- Diskutert funn fra individuelt arbeid
- Funnet relevant lesestoff

14.02.2018

- Rask gjennomgang av funn.

19.02.2018

- Gjennomgang av funn
- Gjort klart et git repository til utviklingen av applikasjonen og sørget for at alle har tilgang
- Diskutert veien videre.
- Begynt på kravspesifikasjonene

21.02.2018 - møte med veileder

- Diskutere applikasjonen
- Diskutere bruk av nøkkler

26.02.2018

- Gått gjennom funn rundt iOS
- Diskutert funn rundt implementasjon av applikasjon
- Gått gjennom alternativer for hvordan vi tenker applikasjonen burde implementere fingeravtrykk-autentisering opp mot en server.
- Sendt e-post til oppdragsgiver og spurt om hvilket alternativ de ønsker at vi skal implementere.

28.02.2018 - møte med veileder

- Diskutere hvordan autentisere en økt med fingeravtrykk

05.03.2018

- Gjennomgang av fremgangen i applikasjonsutviklingen
- Diskutert videre arbeid
- Arbeide med trusselmodellering og applikasjonen

06.03.2018

- Jobbe med applikasjonen
- Jobbe litt med trusel modellering, med fokus på bruks/misbruks diagram

07.03.2018 - møte med veileder

- Gjennomgang av sekvensdiagrammer
- Diskutere applikasjonen

07.03.2018

- Gjennomgang av notater fra møtet med veileder

12.03.2018

- Gjennomgang av kommunikasjon mellom applikasjon og server
- Diskutert fordeler og ulemper med ulike måter å håndtere økter på
- Refkatorering av trusselmodellering

13.03.2018

- Jobbing med applikasjon
- Jobbe med trusselmodellering

15.03.2018

- Jobbing med applikasjon

- Jobbe med trusselmodellering

19.03.2018

- Jobbing med applikasjon
- Jobbe med risikoanalyse

20.03.2018

- Jobbing med applikasjon
- Diskutert videre arbeid med applikasjonen
- Jobbe med risikoanalyse

21.03.2018

- Diskutere applikasjonen

22.03.2018

- Diskutert hva som er det mest sentrale produktet ved konseptbevis-applikasjonen.
- Avgjort at vi vil sette fokuset på at kildekoden skal være lettlest og godt dokumentert for å enkelt kunne brukes til implementere tilsvarende autentisering i en annen applikasjon.

03.04.2018

- Gjennomgang av Trusselmodellering
- Gjennomgang av Design dokument
- Gjennomgang av Applikasjons utvikling
- Angitt prioritering på de kommende oppgavene

11.04.2018 - møte med veileder

- Diskutert beskrivelse av angrep mot infrastruktur

25.04.2018 - møte med veileder

- Gjennomgang av trusselmodellering

09.05.2018 - møte med veileder

- Gjennomgang av rapport utkastet
- Blitt enige om å rekonstruere rapporten, ved å legge til et kapittel for bakgrunn

09.05.2018

- Bearbeidet tilbakemelding fra veileder
- Lagd arbeidsoppgaver basert på tilbakemelding fra veileder
- Fordelt arbeidsoppgaver

G Arbeidslogg

Gjennom semesteret har vi jevnlig ført opp hvor mange timer vi har arbeidet med prosjektet og hva vi har jobbet med de ulike dagene. Denne arbeidsloggen er presentert i tabellen under.

Dato	Oppgave	Deltakere	Tidsforbruk
2018-01-08	Gruppemøte	Henriette, Linn-Mari og	3 timer
2018-01-09	Møte med oppdragsgiver	Henriette, Linn-Mari og Sturla	1 time
2018-01-09	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time og 15 minutter
2018-01-09	Lese metoderapport om mobilovervåkningen	Henriette, Linn-Mari og Sturla	1 time og 30 minutter
2018-01-10	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time og 50 minutter
2018-01-10	Lynkurs i prosjektarbeid	Henriette, Linn-Mari og Sturla	45 minutter
2018-01-10	Kort møte med veileder	Henriette, Linn-Mari og Sturla	10 minutter
2018-01-10	Lese artikler om mobil sikkerhet og mobil arkitektur	Henriette, Linn-Mari og Sturla	1 time og 30 minutter
2018-01-11	Gruppemøte	Henriette, Linn-Mari og Sturla	5 timer
2018-01-13	Lese artikler om mobil sikkerhet og mobil arkitektur	Henriette, Linn-Mari og Sturla	4 timer
2018-01-15	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-01-15	Lese på tidligere bachler oppgave(Pyroeis)	Linn-Mari	1 time
2018-01-15	Lese på tidligere bachler oppgave(Stopmotion)	Henriette	1 time
2018-01-16	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-01-16	Lese på tidligere bachler oppgave(Stopmotion)	Henriette	2 timer
2018-01-16	Lese på tidligere bachler oppgave(Pyroeis)	Linn-Mari	1 time
2018-01-16	Lese på tidligere bachler oppgave(Viten i senter)	Sturla	5 timer
2018-01-17	Møte med veileder	Henriette, Linn-Mari og Sturla	45 minutter
2018-01-17	Gruppemøte	Henriette, Linn-Mari og Sturla	3 timer
2018-01-17	Lese om Android Security & Architecture	Henriette	30 minutter
2018-01-17	Lese på tidligere bachelor oppgave(Pyroeis)	Linn-Mari	3 timer
2018-01-17	Lære Android-apputvikling	Sturla	2 timer
2018-01-17	Lære Android-apputvikling	Henriette	3 timer
2018-01-18	Lese på tidligere bachelor oppgave(Pyroeis)	Linn-Mari	4 timer
2018-01-18	Lese om mobil sikkerhet	Henriette	2 timer
2018-01-18	Lese skriveregler og råd om bacheloroppgaven	Sturla	3 timer
2018-01-19	Skrive på prosjektplan	Linn-Mari	2 timer

2018-01-19	Lese om Android Security & Architecture	Linn-Mari	2 timer
2018-01-20	Skrive på prosjektplan	Henriette og Sturla	2 timer
2018-01-20	Lære Android-apputvikling	Linn-Mari	3 timer
2018-01-21	Skrive utkast av e-post med spørsmål til oppdragsgiver	Sturla	30 minutter
2018-01-21	Skrive på prosjektplan	Henriette og Sturla	3 timer
2018-01-21	Lese om mobil sikkerhet	Henriette og Sturla	2 timer
2018-01-21	Lese om mobil sikkerhet	Linn-Mari	2 timer
2018-01-22	Gruppemøte	Henriette, Linn-Mari og Sturla	4 time
2018-01-22	Lese om Android Security	Henriette, Linn-Mari og Sturla	2 timer
2018-01-23	Gruppemøte	Henriette, Linn-Mari og Sturla	3 timer
2018-01-23	Skrive på prosjektplan	Sturla	3 timer
2018-01-23	Lese om Android Security	Linn-Mari	2 timer
2018-01-23	Skrive på prosjektplan	Linn-Mari	2 timer
2018-01-23	Skrive på prosjektplan	Henriette	30 minutter
2018-01-23	Lese om Android Security	Henriette	2 timer og 30 minutte
2018-01-24	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time
2018-01-24	Møte med veileder	Henriette, Linn-Mari og Sturla	45 minutter
2018-01-24	Lese om Android Security	Henriette, Linn-Mari	2 timer
2018-01-25	Gruppemøte	Henriette, Linn-Mari og Sturla	6 timer
2018-01-26	Lese om Android Security	Henriette	4 timer og 30 minutter
2018-01-26	Lese om Android Security	Sturla, Linn-Mari	5 timer
2018-01-27	Finne og lese vitenskaplige artikler	Linn-Mari	3 timer og 30 minutter
2018-01-27	Se på video om SS7	Henriette, Linn-Mari og Sturla	30 minutter
2018-01-28	Lese om iOS sikkerhet	Henriette	3 timer og 30 minutter
2018-01-28	Lese om iOS sikkerhet	Sturla	4 timer
2018-01-28	Lese om iOS sikkerhet + artikkel om 2FA sync vulnerabilities	Linn-Mari	4 timer
2018-01-29	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-01-30	Gruppemøte	Henriette, Linn-Mari og Sturla	2 timer og 30 minutte
2018-01-31	Gruppemøte	Henriette, Linn-Mari og Sturla	2 timer
2018-01-31	Lese om iOS sikkerhet	Linn-Mari	1 time og 30 minutter
2018-01-31	Møte med veileder	Henriette, Linn-Mari og Sturla	45 minutter
2018-02-01	Les om iOS sikkerhet	Henriette og Sturla	3 timer
2018-02-01	Lese om mobil infrastruktur	Henriette	2 timer

2018-02-01	Skrive på prosjektplan	Sturla	2 timer
2018-02-01	Lese om mobil infrastruktur	Linn-Mari	4 timer
2018-02-02	Lese om mobil infrastruktur og autentisering	Henriette, Linn-Mari og Sturla	5 timer
2018-02-03	Lese om mobil infrastruktur og autentisering	Henriette, Linn-Mari og Sturla	5 timer
2018-02-04	Undersøke sikkerhet i mobil autentisering	Sturla	5 timer
2018-02-04	Lese om mobil infrastruktur og autentisering	Henriette	5 timer
2018-02-04	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-02-05	Lese i Android security cookbook	Henriette	3 timer
2018-02-05	Lese om Permission based android security	Linn-Mari	3 timer
2018-02-05	Sammenligne Android og iOS	Sturla	1 time
2018-02-06	Møte med oppdragsiver	Henriette, Linn-Mari og Sturla	1 time
2018-02-06	Lese i Android Security Internals	Sturla	2 timer
2018-02-06	Lese om Permission based android security	Linn-Mari	2 timer
2018-02-06	Lese i Android security cookbook	Henriette	2 timer
2018-02-07	Gruppemøte	Henriette, Linn-Mari og Sturla	2 timer
2018-02-07	Møte med veileder	Henriette, Linn-Mari og Sturla	1 time
2018-02-08	Lese i Android security cookbook	Henriette	3 timer
2018-02-09	Lese i Android security cookbook	Henriette	7 timer
2018-02-09	Lese om Android Permissions	Linn-Mari	6 timer
2018-02-09	Lese i Android Security Internals	Sturla	8 timer
2018-02-10	Lese i Android security cookbook	Henriette	5 timer
2018-02-10	Lese om vendor customizations	Linn-Mari	7 timer
2018-02-10	Lese i Android Security Internals	Sturla	7 timer
2018-02-11	Lese om Android Security(Malware and defenses)	Linn-Mari	4 timer
2018-02-11	Lese i Android Security Internals	Sturla	4 timer
2018-02-11	Lese i Android security cookbook	Henriette	6 timer
2018-02-12	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time og 30 minutter
2018-02-12	Lese i Android Security Internals	Sturla	3 timer og 30 minutter
2018-02-12	Lese om Android Security(Malware and defenses)	Linn-Mari	3 timer
2018-02-12	Lese i Android Security cookbook	Henriette	2 timer
2018-02-13	Lese om Android Security(Malware and defenses)	Linn-Mari	4 timer
2018-02-14	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time
2018-02-14	Lese om Android Security	Linn-Mari	3 timer
2018-02-14	Lese i Android Security cookbook	Henriette	3 timer

2018-02-15	Lese i Android Security cookbook	Henriette	4 timer
2018-02-15	Lese om Android Security	Linn-Mari	4 timer
2018-02-15	Lese i Android Security Internals	Sturla	5 timer
2018-02-16	Lese om Android Bootloader sårbarheter	Henriette	6 timer
2018-02-17	Lese i Android Security Internals	Sturla	7 timer
2018-02-17	Lese om Android Security	Linn-Mari	5 timer
2018-02-17	Lese om Android Bootloader sårbarheter	Henriette	6 timer
2018-02-18	Lese i Android Security Internals	Sturla	9 timer
2018-02-18	Lese om Android Security	Linn-Mari	7 timer
2018-02-18	Lese om Android Bootloader sårbarheter	Henriette	7 timer
2018-02-19	Gruppemøte	Henriette, Linn-Mari og Sturla	3 timer
2018-02-19	Lese om hvordan autentisere opp mot en server i en Android applikasjon	Sturla	4 timer
2018-02-20	Sette opp utvikler-telefon og skrive kravspesifikasjon til applikasjon	Sturla	3 timer
2018-02-21	Lese om iOS Penetration Testing	Linn-Mari	2 timer
2018-02-21	Lese om hvordan å implementere HTTPS sikrest mulig	Sturla	3 timer
2018-02-22	Lese om iOS Penetration Testing	Linn-Mari	2 timer
2018-02-22	Lese om iOS Security Internals	Henriette	2 timer
2018-02-23	Lese om iOS Security Internals	Henriette	9 timer
2018-02-23	Utforme design og modellering av applikasjonen	Sturla	8 timer
2018-02-23	Diskutere autentiseringsmetode for applikasjon	Henriette, Linn-Mari og Sturla	30 minutter
2018-02-23	Lese om iOS Penetration Testing	Linn-Mari	6 timer
2018-02-24	Lese om iOS Penetration Testing	Linn-Mari	2 timer og 30 minutter
2018-02-24	Lese om Analysis and research on iOS security system	Henriette	7 timer
2018-02-24	Refaktorere kravspesifikasjon til valgt autentiseringsmetode	Sturla	5 timer og 30 minutter
2018-02-25	Designet arkitektur og begynt på sekvensdiagrammer	Sturla	6 timer
2018-02-25	Lese om iOS Penetration Testing	Linn-Mari	5 timer
2018-02-25	Lese om Analysis and research on iOS security system og relaterte artikler	Henriette	9 timer
2018-02-26	Gruppemøte	Henriette, Linn-Mari og Sturla	2 timer
2018-03-02	Using the smartphone pentest framework	Henriette	6 timer
2018-03-02	Integrering med git og research om android studio	Linn-Mari	6 timer

2018-03-02	Identifikasjon av sikrest mulig autentisering med nøkkelpar	Sturla	7 timer
2018-03-03	Oppsett av registrering, logginn og fingeravtrykk i app	Linn-Mari	9 timer
2018-03-03	Begynne å skrive på rapporten	Henriette	9 timer
2018-03-03	Planlegging av hvordan design og krav skal implementeres	Sturla	9 timer
2018-03-04	Implementasjon av registrering og innlogging på serveren	Sturla	9 timer
2018-03-04	Implementasjon av fingeravtrykk + research om best practices	Linn-Mari	9 timer
2018-03-04	Jobbe med rapporten og notatene	Henriette	9 timer
2018-03-05	Gruppemøte	Henriette, Linn-Mari og Sturla	2 timer
2018-03-05	Refaktorering av autentiseringsnøklene	Linn-Mari	3 timer
2018-03-05	Begynne med trusselmodellering	Henriette	2 time
2018-03-06	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-03-06	Undersøking av server-app kommunikasjon	Linn-Mari	1 time
2018-03-06	Jobbe med trusselmodellering	Henriette	1 time
2018-03-06	Jobbe med trusselmodellering og forberede møte med veileder	Sturla	4 timer
2018-03-07	Møte med veileder	Henriette, Linn-Mari og Sturla	1 time
2018-03-07	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time
2018-03-07	Skrivekurs	Henriette, Linn-Mari og Sturla	1 time
2018-03-07	Jobbe med trusselmodellering	Henriette	2 timer
2018-03-07	Lage sekvensdiagram for å starte en økt og utforming av kort-navn	Sturla	2 timer
2018-03-08	Applikasjonsutvikling	Linn-Mari	7 timer
2018-03-08	Jobbe med trusselmodellering	Henriette	7 timer
2018-03-08	Refaktorere økt til å bruke nøkkel fra nøkkel-lager	Sturla	7 timer
2018-03-09	Applikasjonsutvikling	Linn-Mari	3 timer
2018-03-09	Implementere økthåndtering med sikkert lagret nøkkel	Sturla	4 timer
2018-03-09	Jobbe med trusselmodellering	Henriette	6 timer
2018-03-09	Applikasjonsutvikling	Linn-Mari	5 timer
2018-03-10	Applikasjonsutvikling	Linn-Mari	4 timer
2018-03-11	Jobbe med trusselmodellering	Henriette	7 timer
2018-03-11	Jobbe med design av applikasjon	Sturla	5 timer
2018-03-12	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-03-12	Jobbe med trusselmodellering	Henriette	3 timer
2018-03-13	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer

2018-03-14	Jobbe med trusselmodellering	Henriette	4 timer
2018-03-15	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-03-15	Jobbe med trusselmodellering	Henriette	4 timer
2018-03-15	Applikasjonsutvikling	Linn-Mari	3 timer
2018-03-15	Implementere lagring og verifisering av nøkler og signaturer på server	Sturla	6 timer
2018-03-16	Laget sekvensdiagramm og beskrivelse for alternativer autentiseringsmetoder	Sturla	3 timer
2018-03-16	Ryddet opp i app og lagt inn feilmeldinger	Linn-Mari	4 timer
2018-03-16	jobbe med trusselmodellering	Henriette	4 timer
2018-03-17	Jobbe med risikoanalyse	Henriette	3 timer
2018-03-17	Applikasjonsutvikling	Linn-Mari	2 timer
2018-03-17	Jobbe med risikoanalyse	Sturla	7 timer
2018-03-19	Gruppemøte	Henriette, Linn-Mari og Sturla	3 timer
2018-03-19	Jobbe med risikoanalyse	Henriette	1 time
2018-03-20	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-03-20	Refaktorere design av applikasjon	Sturla	2 timer
2018-03-20	Jobbe med risikoanalyse	Henriette	1 time
2018-03-21	Møte med veileder	Henriette, Linn-Mari og Sturla	30 minutter
2018-03-21	Jobbe med risikoanalyse	Henriette	4 timer
2018-03-22	Gruppemøte	Henriette, Linn-Mari og Sturla	3 timer
2018-03-22	Sammenslåing av førstegangsautentisering og opprettelse av økt	Sturla	3 timer
2018-03-22	Jobbe med risikoanalyse	Henriette	1 time
2018-03-22	Applikasjonsutvikling	Linn-Mari	3 timer
2018-03-22	Oppdatere server etter nytt design	Sturla	5 timer
2018-03-23	Jobbe med risikoanalyse	Henriette	4 timer
2018-03-23	Refaktorering	Linn-Mari	4 timer
2018-03-24	Jobbe med risikoanalyse	Henriette	6 timer
2018-03-24	Refaktorering	Linn-Mari	3 timer
2018-03-24	Jobbe med trusselmodellering og kartlegging av gjenstående oppgaver	Sturla	6 timer og 30 minutter
2018-03-25	Jobbe med trusselmodellering	Henriette og Sturla	3 timer
2018-03-25	Applikasjonsutvikling	Linn-Mari	6 timer
2018-03-26	Jobbe med trusselmodellering	Henriette	6 timer
2018-03-26	Applikasjonsutvikling	Linn-Mari	3 timer
2018-03-26	Utarbeide beskrivelse av design	Sturla	8 timer
2018-03-27	Applikasjonsutvikling	Linn-Mari	3 timer
2018-03-27	Utarbeide beskrivelse av design	Sturla	8 timer
2018-03-28	Refaktorering og feilretting av app	Linn-Mari	5 timer

2018-03-29	Refaktorering av app	Linn-Mari	3 timer
2018-03-31	Debugging av app	Linn-Mari	2 timer
2018-04-02	Debugging av app	Linn-Mari	7 timer
2018-04-03	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-04-03	Applikasjonsutvikling	Linn-Mari	2 timer
2018-04-03	Jobbe med trusselmodellering	Henriette og Sturla	1 time
2018-04-04	Gruppemøte	Henriette, Linn-Mari og Sturla	3 timer
2018-04-04	Lese på mobil-infrastruktur	Henriette	2 timer
2018-04-04	Applikasjonsutvikling	Linn-Mari	2 timer
2018-04-05	Lese på mobil-infrastruktur	Linn-Mari	1 time
2018-04-05	Gruppemøte	Henriette, Linn-Mari og Sturla	4 timer
2018-04-05	Lese på mobil-infrastruktur	Henriette og Sturla	1 time
2018-04-06	Lese om angrep på og beskyttelse for SS7	Henriette og Sturla	6 timer
2018-04-06	Lese om SS7	Linn-Mari	5 timer
2018-04-07	Lese om angrep på og beskyttelse for SS7	Henriette og Sturla	4 timer
2018-04-07	Lese om SS7	Linn-Mari	2 timer
2018-04-08	Lese om angrep på og beskyttelse for SS7	Henriette	6 timer
2018-04-08	Lese om sikkerhet i norsk mobilinfrastruktur	Sturla	8 timer
2018-04-09	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time
2018-04-09	Lese om angrep på og beskyttelse for SS7	Henriette	4 timer og 30 minutter
2018-04-09	Lese og skrive om SS7 angrep	Linn-Mari	4 timer
2018-04-09	Lese om sikkerhet tilknyttet Huawei	Sturla	2 timer
2018-04-10	Beskrive angrep mot infrastruktur	Henriette	3 timer
2018-04-10	Beskrive hvordan nøkkellagere er implementert	Sturla	2 timer
2018-04-10	Beskrive angrep mot infrastruktur	Linn-Mari	3 timer
2018-04-11	Gruppemøte	Henriette, Linn-Mari og Sturla	1 time
2018-04-11	Beskrive hvordan nøkkellagere er implementert	Sturla	2 timer
2018-04-11	Beskrive angrep mot infrastruktur og lese på overvåkning	Henriette	3 timer
2018-04-11	Beskrive angrep mot infrastruktur og lesing om MAP-meldinger	Linn-Mari	3 timer
2018-04-12	Beskrive angrep mot infrastruktur	Linn-Mari	2 timer
2018-04-12	Beskrive angrep mot infrastruktur og lese på overvåkning	Henriette	1 time
2018-04-13	Beskrive og forklare overvåkning via SS7 sårbarheter	Henriette	6 timer
2018-04-13	Finne mer informasjon om SS7 angrep	Linn-Mari	6 timer

2018-04-13	Vurdere praksisen ved å stole på SMS som kodebærer	Sturla	9 timer
2018-04-14	Lese og skrive på kjøp av tilgang til SS7	Henriette	6 timer
2018-04-14	Beskrive angrep mot infrastruktur og lesing om detaljer rundt angrep	Linn-Mari	8 timer
2018-04-14	Vurdere SMS opp mot andre kodebærere	Sturla	9 timer
2018-04-15	Beskrive hvordan ulike aktører kan skaffe seg tilgang til SS7	Sturla	4 timer
2018-04-15	Skrive om mobilovervåkning og hvordan skaffe tilgang til SS7	Henriette	6 timer
2018-04-16	Gruppemøte	Henriette	1 time
2018-04-18	Rapportskriving	Linn-Mari	2 timer
2018-04-19	Rapportskriving og lesing på andre rapporter	Linn-Mari	5 timer
2018-04-19	Rapportskriving	Henriette og Sturla	3 timer
2018-04-20	Rapportskriving	Henriette og Sturla	9 timer
2018-04-20	Rapportskriving	Linn-Mari	8 timer
2018-04-21	Rapportskriving	Henriette, Sturla og Linn-Mari	8 timer
2018-04-21	Rapportskriving	Henriette og Sturla	8 timer
2018-04-22	Videre dokumentasjon av kode	Linn-Mari	6 timer
2018-04-23	Rapportskriving	Henriette	1 time
2018-04-24	Rapportskriving	Linn-Mari	2 timer og 30 minutter
2018-04-24	Rapportskriving	Sturla	4 timer
2018-04-24	Rapportskriving	Henriette	3 timer
2018-04-25	Rapportskriving	Henriette	4 timer
2018-04-27	Rapportskriving	Sturla	9 timer
2018-04-27	Rapportskriving	Henriette	7 timer
2018-04-27	Rapportskriving	Linn-Mari	8 timer
2018-04-28	Rapportskriving	Sturla	9 timer
2018-04-28	Rapportskriving	Linn-Mari	8 timer
2018-04-28	Rapportskriving	Henriette	8 timer
2018-04-29	Rapportskriving	Henriette	9 timer
2018-04-29	Rapportskriving	Linn-Mari	7 timer
2018-04-29	Rapportskriving	Sturla	9 timer
2018-04-30	Rapportskriving	Henriette	3 timer
2018-04-30	Rapportskriving	Linn-Mari	2 timer
2018-04-30	Rapportskriving	Sturla	2 timer
2018-05-01	Rapportskriving	Sturla	2 timer
2018-05-01	Rapportskriving	Henriette	3 timer
2018-05-02	Rapportskriving	Henriette	4 timer
2018-05-02	Rapportskriving	Linn-Mari	3 timer
2018-05-02	Rapportskriving	Sturla	5 timer
2018-05-03	Rapportskriving	Sturla	4 timer
2018-05-03	Rapportskriving	Linn-Mari	3 timer
2018-05-05	Rapportskriving	Sturla	8 timer
2018-05-05	Rapportskriving	Linn-Mari	8 timer

2018-05-05	Rapportskriving	Henriette	10 timer
2018-05-06	Rapportskriving	Sturla	8 timer
2018-05-06	Rapportskriving	Linn-Mari	7 timer
2018-05-06	Rapportskriving	Henriette	8 timer
2018-05-07	Gruppemøte	Henriette, Sturla og Linn-Mari	3,5 timer
2018-05-07	Rapportskriving	Linn-Mari	5 timer
2018-05-07	Rapportskriving	Henriette	7 timer
2018-05-07	Rapportskriving	Sturla	4 timer
2018-05-08	Gruppemøte	Henriette, Sturla og Linn-Mari	4 timer
2018-05-08	Rapportskriving	Linn-Mari	3 timer
2018-05-08	Rapportskriving	Henriette	3 timer
2018-05-08	Rapportskriving	Sturla	3 timer
2018-05-09	Gruppemøte	Henriette, Sturla og Linn-Mari	5 timer
2018-05-09	Rapportskriving	Linn-Mari	2 timer
2018-05-09	Rapportskriving	Henriette	3 timer
2018-05-10	Rapportskriving	Henriette	8 timer
2018-05-10	Rapportskriving	Linn-Mari	7 timer
2018-05-10	Rapportskriving	Sturla	7 timer
2018-05-11	Rapportskriving	Sturla	7 timer
2018-05-11	Rapportskriving	Henriette	7 timer
2018-05-11	Rapportskriving	Linn-Mari	6 timer
2018-05-12	Rapportskriving	Henriette	8 timer
2018-05-12	Rapportskriving	Linn-Mari	7 timer
2018-05-12	Rapportskriving	Sturla	6 timer
2018-05-13	Rapportskriving	Henriette	6 timer
2018-05-13	Rapportskriving	Linn-Mari	5 timer
2018-05-13	Rapportskriving	Sturla	7 timer
2018-05-14	Gruppemøte	Henriette, Sturla og Linn-Mari	4 timer
2018-05-14	Korrekturlesing av rapport	Sturla	2 timer
2018-05-14	Korrekturlesing av rapport	Henriette	3 timer
2018-05-14	Korrekturlesing av rapport	Linn-Mari	2 timer
2018-05-15	Gruppemøte	Henriette, Sturla og Linn-Mari	6 timer
2018-05-15	Korrekturlesing av rapport	Henriette	3 timer
2018-05-15	Korrekturlesing av rapport	Linn-Mari	3 timer
2018-05-15	Korrekturlesing av rapport	Sturla	3 timer
2018-05-16	Korrekturlesing av rapport	Henriette, Sturla og Linn-Mari	3 timer