KAPITEL 9

# IT-SIKKERHED

Sikkerhed er helt afgørende for mange IT-systemer.

**EKSEMPEL**

### SYSTEMER, HVOR SIKKERHED ER KRITISK

Et usikkert kontrolsystem på et atomkraftværk, i et fly eller i en rumfærge kan skade mennesker, dyr eller miljø. En usikker hæveautomat kan tillade ondsindede personer at hæve andre folks penge. Et usikkert hospitalssystem kan udstille personfølsomme oplysninger som fx patientjournaler. En usikker tyverialarm på et museum kan tillade tyveri af uerstattelige kunstværker. En usikker hjemmecomputer risikerer at blive inficeret med ondsindede programmer, aflyttet eller overtaget af hackere.

Som eksemplet viser, er IT-sikkerhed en bredt favnende disciplin. Dette kapitel beskæftiger sig kun med de IT-tekniske aspekter af IT-sikkerhed, selvom man med rette kan sige at IT-sikkerhed også handler om fx psykologi, jura og økonomi.

**ORDFORKLARING**

### IT-SIKKERHED

*IT-sikkerhed* handler om at bygge IT-systemer, der giver mindst mulig risiko for

- interne fejl,
- uheld,
- ondsindede angreb.

Skulle systemet alligevel udsættes for fejl, uheld eller angreb, skal IT-sikkerhed sørge for, systemet fungerer pålideligt.

## 9.1. IT-SYSTEMER OG AKTØRER

Et *IT-system* kan i sikkerhedsmæssig sammenhæng være mange ting – vi vil her skelne mellem tre niveauer:

1. Et hardware- eller softwareprodukt.
2. En enkelt computer.
3. En samling computere i et lokalnetværk.

Hvert IT-system har *en eller flere brugere*.

En bruger af et IT-system har en række rettigheder. Groft sagt sikrer *udvikling af IT-systemer*, at brugere kan udøve deres rettigheder (fx Anna kan læse og skrive i denne fil), mens *sikring af IT-systemer* sørger for, at hverken brugere eller andre kan gøre ting, de ikke har rettigheder til (fx Terese kan ikke slette Annas fortrolige data).

Lad for nemheds skyld en *aktør* betyde enten en person (fx en privatperson eller en ansat i et firma) eller en software-enhed (fx en computervirus eller et antivirusprogram). Begrebet bliver praktisk, fordi vi ønsker at omtale ondsindede personer og ondsindede programmer under et.

## 9.2. MÅL MED IT-SIKKERHED

Sikkerhed er groft sagt en velfungerende *beskyttelse af værdier*, fx menneskeliv, natur, penge eller kunst. Indenfor IT-sikkerhed er de værdier, der skal beskyttes, *information*, fx kreditkortnumre, kodeord, personfølsom data eller oplysninger om et atomkraftværks IT-sikkerhedssystem. Uretmæssig omgang med eller ødelæggelse af information, der skal beskyttes, er et sikkerhedsbrud. Får en spion fat i statshemmeligheder, kan han videresælge dem til terrorister. Får en tyv fat i et kreditkortnummer, kan han hæve penge, der ikke er hans. Ødelagte konto-data i en bank kan resultere i en personlig fallit – eller at banken skal af med millioner af kroner.

Visse ideer fra traditionel fysisk beskyttelse af genstande kan oversættes til IT-sikkerhed (fx ideen om adgangskontrol). Der er imidlertid en lang række forskelle på traditionel sikkerhed og IT-sikkerhed – vi oplister et par stykker:

- Hvis et maleri bliver stjålet, er det væk. Hvis en information aflyttes, fx et kreditkortnummer, har tyven blot en kopi. Man kan altså ikke umiddelbart vide, om ens kreditkortnummer er blevet stjålet, bare ved at tjekke, om man stadig har det.
- Modsat traditionel sikkerhed kan det være svært at forstå, *hvad* IT-sikkerhed skal beskytte – informationer er mere uhåndgribelige end malerier, guld eller fladskærme.
- De fleste låser deres bil af, inden de forlader den. Ingen efterlader en 1000-krone-seddel på gaden og regner med at finde den dagen efter. Folk er opmærksomme på traditionelle sikkerhedstrusler (fx tyve). Men mange ved slet ikke, at de bør tænke på IT-sikkerhed.
- IT-kriminalitet udøves typisk ved at sende "onde bits i en ond rækkefølge" gennem et lille internet-stik i væggen. Så selv om man er opmærksom på at være IT-sikker, kræver det en større teknisk indsigt at blokere for "onde bits" og samtidig lade "gode bits" slippe igennem.

### 9.2.1. FORTROLIGHED, INTEGRITET OG TILGÆNGELIGHED

Normalt skelner man mellem 3 overordnede typer af informationsbeskyttelse:

- *Fortrolighed*: Forhindring af, at information uretmæssigt afsløres.
- *Integritet*: Forhindring af, at information uretmæssigt ændres eller slettes.
- *Tilgængelighed*: Sikring af, at information altid er tilgængelig og brugbar, når dens retmæssige brugere ønsker.

**EKSEMPEL**

#### FORTROLIGHED, INTEGRITET OG TILGÆNGELIGHED: BANKKONTO

Et banksystem beskytter Ingrids konto, herunder hendes kreditkortnummer og saldo. *Fortrolighed* er, at kun Ingrid og hendes bankrådgiver kender til kontooplysningerne – og ikke afslører det til tilfældige personer på gaden. *Integritet* er, at Ingrids saldo er "som den bør være": saldoen bliver kun ændret, hvis Ingrid sætter penge ind eller hæver penge på kontoen, og saldoen bliver i så fald opdateret korrekt. *Tilgængelighed* er, at Ingrid kan hæve og indsætte penge – og på andre måder tilgå sin konto – når hun har ret til og brug for det.

**EKSEMPEL**

#### FORTROLIGHED, INTEGRITET OG TILGÆNGELIGHED: FORRETNINGSHEMMELIGHED

Det landsdækkende firma DK-Hotdog sælger hotdogs. Chefen udtænker nu *cold dog'en*. Den kolde hotdog skal lanceres i firmaets pølsevogne inden for et år. Indtil da er opfindelsen en forretningshemmelighed. Firmates chef lægger en tekst-fil med en beskrivelse af cold dog'en på DK-Hotdogs webside, så de ansatte kan lære, hvordan cold dog'en tilberedes.

*Fortrolighed* er, at kun firmaets ansatte kan læse beskrivelsen. Det skal altså sikres at konkurrenten Dansk Skodmad ikke opsnapper beskrivelsen. DK-Hotdogs webside skal altså have *adgangkontrol*.

*Integritet* er, at kun firmaets chef kan ændre i beskrivelsen. Da det er chefen, der har udtænkt cold dog'en, skal ansatte forhindres i (med vilje, eller ved en fejl) at ændre beskrivelsen til noget forkert.

*Tilgængelighed* er, at at alle firmaets ansatte faktisk kan komme til at læse beskrivelsen når som helst. DK-Hotdogs webside skal altså ikke gå ned, og den skal have et system, der sikrer alle medarbejdere adgang.

**TIP**

#### CIA-MODELLEN

På engelsk forkortes confidentiality (fortrolighed), integrity (integritet) og availability (tilgængelighed) ofte til "CIA", og man taler om *CIA-modellen* for IT-sikkerhed.

### 9.2.2. TRUSLER, SÅRBARHEDER OG MODMIDLER

Indenfor IT-sikkerhed er det nyttigt at skelne mellem *trusler* mod et system og systemets *sårbarheder*.

ORDFORKLARING

**SÅRBARHED**

Systemsvaghed.

En sårbarhed kan ofte give en ondsindet aktør uautoriseret adgang til en information.

ORDFORKLARING

**TRUSSEL OG TRUSSELSAKTØR**

En *trussel* er en potentiel fare for et system. En *trusselsaktør* kan gøre truslen til virkelighed. En trusselsaktør kan være en ondsindet aktør, der kan finde en sårbarhed og udnytte den mod systemet, eller en velmenende aktør, der ved en fejl kan skade et sårbart system.

Overordnet kan man sige, at

*trussel = sårbarhed + trusselsaktør.*

| TRUSSELSAKTØR | SÅRBARHED(ER) | TRUSLER |
|---|---|---|
| Computervirus | Manglende antivirussoftware, sikkerhedshuller i software | Virusinfektion |
| Hacker | Manglende adgangskontrol | Uautoriseret adgang til information<br>Systemet bliver overtaget |
| Systembruger | Bruger systemet forkert | Systemnedbrud<br>Tab af data |
| Ildebrand | Brændbart materiale, ingen ildslukker | Tab af computere og information, måske endda menneskeliv |

*Fig. 9.147. Eksempler på trusselsaktører, sårbarheder og trusler – der vil blive uddybet i resten af dette kapitel.*

Vil man nedsætte risikoen for, at en trussel bliver til virkelighed, må man bygge værn, trussel for trussel. Man kalder sådanne værn for *modmidler*. Man som regel ikke kan fjerne trusselsaktører – de kan være alt fra hackere i fjernøsten eller medarbejdere med rent mel i posen til ondsindende programmer i omløb på internettet udenfor menneskelig kontrol. Modmidler virker derfor som regel ved at reducere et systems sårbarhed.

ORDFORKLARING

**MODMIDDEL**

Et *modmiddel* mindsker en trussel, som regel ved at mindske en sårbarhed.

Et modmiddel kan konkret være mange ting, fx en *firewall*, en softwarekonfiguration eller en bestemt procedure, som brugere af et IT-system skal følge.

| SÅRBARHED | MINDSKES AF MODMIDDEL | TYPE AF MODMIDDEL |
|---|---|---|
| Manglende antivirussoftware | Antivirussoftware | Software |
| Manglende adgangskontrol | Firewall | Hardware |
| Systembruger bruger systemet forkert | Indstil systemet, så brugere har mindst mulig risiko for at begå fejl | Software-konfiguration |
| Systembruger bruger systemet forkert | Undervis brugere i, hvordan systemet bruges korrekt | Procedurer, brugere skal følge |

*Fig. 9.148. Eksempler på typer af modmidler.*

Selvom sårbarheden "manglende adgangskontrol" mindskes med modmidlet, er den ikke udryddet. En sårbarhed kræver ofte flere modmidler for at blive udryddet. Men selv med uhyrlige bunker af modmidler:

*Et IT-system er aldrig 100 % sikkert!*

### 9.2.3. PRIVACY

Man kan opdele IT-sikkerhedsmålet *fortrolighed* i

- fortrolighed på et *personligt plan*: Evnen til og retten til at beskytte personlige hemmeligheder.
- fortrolighed på *gruppeplan*: Evnen til og retten til at beskytte hemmeligheder internt i en organisation, branche, virksomhed, institution, kommission, forskergruppe, …

Fortrolighed på det personlige plan kaldes *privacy* (engelsk for "privat-hed"). Der er mange ting fra privatsfæren, man som enkeltperson (eller familie) ikke ønsker udbasuneret til alle og enhver – fra politisk ståsted, religion, hårfarve, økonomisk status, sygdomme eller medicinforbrug til information om, hvor man befinder sig på hvilke tidspunkter, hvornår man står op og ligger ned, hvad man spiser til morgenmad eller hvad man skriver sms'er til sine kammerater om.

IT-systemernes stadige udbredelse i hverdagslivet har gjort debatten om privacy aktuel: Hvilken information må websider lagre om deres besøgende? Må en virksomhed overvåge de ansattes brug af internettet? Må politiet aflytte borgeres internetbrug?

Selvom enkeltlande lovgiver omkring fx lagring af persondata, er der et stykke vej fra at håndhæve disse love til fx at kunne sige, at vi færdes anonymt på internettet:

- Al internettrafik går gennem en internetudbyder (ISP), der ofte lagrer information om,
  - hvilke computere, der har kontaktet hvilke andre computere via internettet
  - hvilke data, de kommunikerende computere har udvekslet
- Søgemaskiner husker søgninger fra de enkelte computere
- Internetforretninger husker købsinformation om kunder
- Sociale online-fora har personadresser mv. lagret
- …

Men selvom vi stoler på vores internetudbyder og kun afslører information, vi ønsker at afsløre, til internetforretninger, søgemaskiner, mv. – er der stadigt langt til anonymitet: Blot det *at sende en enkelt pakke* på et netværk afslører information: En pakke fra computer A til computer B afslører fx, at A og B snakker sammen. Hverken A eller B er altså som udgangspunkt *anonyme* – og selve det faktum, at der er trafik imellem A og B, kan observeres. Det kunne sagtens tænkes, at A og B gerne ville snakke sammen, uden at nogen vidste, at der overhovedet fandt en samtale sted. Dette ønske kaldes *ikke-observabilitet*.

Vi har altså identificeret to delaspekter af privacy:

- anonymitet,
- ikke-observabilitet.

Som privatperson kunne man ønske sig maksimal privacy. I realiteten er det dog meget svært at undgå at afsætte elektroniske spor, når man fx har med internettet at gøre.

### 9.2.4. UAFVISELIGEHED

Ideen med at underskrive et dokument er, at underskriveren ikke senere kan "løbe fra sin underskrift". Har man skrevet under på en aftale om at sælge villaen, kan man ikke bagefter afvise at sælge. Det kaldes *uafviselighed*. Uden uafviselighed ville underskriften være værdiløs.

Et tilsvarende problem gør sig gældende i IT-verdenen. Man ønsker at kunne underskrive visse elektroniske informationer på en uafviselig måde. På den måde kan modtager være sikker på, afsender ikke "løber fra" løfter, der af givet i den elektroniske information. Det kaldes *uafviselighed* og er et aspekt af informationens integritet.

**ORDFORKLARING**

**UAFVISELIGHED**

En handling er uafviselig, hvis den, der udfører handlingen, ikke bagefter kan påstå, at vedkommende ikke udførte handlingen.

**EKSEMPEL**

**DIGITAL SIGNATUR**

Et underskrift foretaget med en digital signatur er uafviselig.

Vi kommer ind på digital signatur i afsnit 9.5.7.

## 9.3. SÅRBARHEDER OG TRUSLER

Der er 3 typer trusler mod et IT-systems sikkerhed:

- *naturskabte*: fx jordskælv, brand, oversvømmelse eller strømafbrydelse
- *tekniske*: computerne i IT-systemet kan gå i stykker
- *menneskelige*: mennesker kan
  - begå fejl,
  - spionere, stjæle og sabotere.

Vi vil kun beskæftige os med menneskeskabte trusler.

### 9.3.1. VELMENENDE PERSONER (BRUGERE)

Et computersystem er ikke mere sikkert end de mennesker, der betjener sig af det – brugerne. De færreste brugere har kriminelle hensigter, men kan alligevel

- begå fejl,
- være skødesløse med kodeord eller andre følsomme oplysninger,
- lade sig narre.

Menneskelige fejl tegner sig for størsteparten af alle brud på sikkerheden i ethvert IT-system. En stor del af sådanne brud på sikkerheden sker, hvor naive eller fortravlede brugere afgiver følsomme oplysninger. Følsomme oplysninger afgives typisk på to måder:

- *Intetanende*: Det er svært at undgå at afgive oplysninger, hvis man færdes på internettet: Fx kan websteder sagtens lokke en masse information om en internetsurfers computer ud af hans browser, og offentliggør man et dokument på internettet, kan det indeholde information om forfatterens computer og det netværk, den er tilkoblet.
- *Frivilligt*: Brugere narres til at udføre falske instruktioner – som at åbne en bestemt webside, åbne en vedhæftet fil i en email, angive et kodeord, el.l. Websiden eller den vedhæftede fil indeholder ondsindet kode, der udnytter de følsomme oplysninger, som brugeren har afgivet frivilligt. Brugerne lader sig narre, fordi instruktionerne tilsyneladende kommer fra fx IT-afdelingen i deres firma, en ven fra adresselisten, el.l. En ondsindet aktør udgiver sig altså for at være en bekendt – det kaldes *social engineering*.

### 9.3.2. ONDSINDEDE PERSONER

Der er ikke én bestemt type person, der skriver ondsindede programmer: Der er unge og gamle, professionelle og amatører, europæere og asiater, rige og fattige. Men *hvorfor* overhovedet skrive ondsindet kode? Her er nogle af de væsentligste svar på spørgsmålet:

- *Penge*: Computerbrugere verden over har i stigende grad brug for at logge ind på forskellige portaler og betale med kreditkort i netbutikker. Ved at stjæle kodeord eller kreditkortnumre kan ondsindede personer stjæle penge. De kan også videresælge information eller lave mere organiseret kriminalitet.
- *Politiske årsager*: Er man utilfreds med det politiske indhold på en webside, kan man jo bare gøre websiden utilgængelig eller ændre indholdet på den – så er den klaret! Det er ulovligt, men ikke desto mindre en almindelig type angreb.
- *Berømmelse*: Både unge og erfarne programmører kan ønske berømmelse eller anseelse – og forsøger at opnå det ved at at skrive en effektiv computervirus (uheldigt!).
- *Forskning og udvikling af antivirus-software*: Professionelle programmører udfordrer konstant sikkerheden i eksisterende operativsystemer, browsere og andre programmer. Finder de et sikkerhedshul, sørger de for at lave sikkerheds-software, der kan lukke hullet. På den måde kan man være på forkant med de IT-kriminelle.

En *hacker* er en person, der tiltvinger sig uretmæssig adgang til et IT-system.

### 9.3.3. KODEORD

Kodeord giver adgang til information, som kun kenderen af kodeordet bør have adgang til. Alligevel regnes kodeord ikke for en særlig sikker adgangskontrol – mange brugere

- *bruger gætbare kodeord* som fx konens fødselsdag eller hundens navn.
- *bruger svage kodeord*, der kan knækkes på kort tid med en kodeordsknækker. Et kodeord regnes for svagt, hvis det er kort, eller gætbart, eller danner eksisterende ord. Fx er "?}_1", "Bastian" og "Jegerenfalk" svage kodeord.
- *skriver kodeord ned* for at huske dem. En ondsindet aktør kan finde det nedskrevne kodeord og misbruge det.
- *afslører kodeordet* til ondsindede personer, der udgiver sig for at være en anden end de er (social engineering).
- *bruger samme kodeord flere steder*: De fleste skal huske en stor bunke kodeord til mange forskellige systemer – PIN-kode, kode til email, kode til arbejdscomputeren, mv. For at kunne huske kodeordene, bruges så vidt muligt det samme kodeord alle steder. Det gør det lettere for en angriber at knække kodeordet.

De bedste kodeord er genereret tilfældigt – på den måde kan de ikke sammenkædes med brugeren. Fx er kodeordene

"`_t[71WLNiNx`", "`x?[3qD€ct(GnM`" og "`+(LaNqZ-/67-8`"

genereret tilfældigt og meget lidt gætbare. Til gengæld er de svære at huske. En endnu bedre løsning er *smart cards*, der genererer et nyt, tilfældigt kodeord hver gang, kodeordet skal bruges. På den måde undgås genbrug af kodeord fuldstændigt.

### 9.3.4. SOFTWARE

Mange sårbarheder opstår, fordi eksisterende programmer ikke er sikkert kodet. Der kan være *bugs* (deciderede fejl i koden), manglende sikkerhedstjek eller bagdøre, som (den ellers velmenende) programmør har glemt at lukke. Ondsindede aktører kan undersøge software-usikkerhederne og udnytte dem til målrettede angreb.

EKSEMPEL

**INTERNETBROWSERE**

I 2004 rådede CERT (*US Computer Emergency Readiness Team*, en IT-sikkerhedsorganisation) internetbrugere til at bruge en hvilken som helst anden browser end Internet Explorer, fordi der var så mange sikkerhedshuller i programmet. Selvom andre browsere i princippet var lige så usikre, havde Internet Explorers daværende allestedsnærværelse gjort programmet til et oplagt hackermål. Hvert år finder man hundredevis af sikkerhedshuller i aktuelle internetbrowsere, som skyldes forhastet eller uigennemtænkt programmering. Sådanne huller lukkes ved at installere sikkerhedsopdateringer.

### 9.3.5. MALWARE

*Malware* er en samlebetegnelse for ondsindet kode. Ordet er en sammentrækning af *malicious* (engelsk for ondsindet) og *software*. Der findes adskillige typer malware, hvoraf nogle af de mere kendte omfatter

- virus,
- orme,
- spyware,
- trojanske heste.

At kode er *ondsindet*, betyder at koden griber ind i et systems stabilitet og sikkerhed på en uønsket måde, fuldstændig som en sygdom griber ind i en krops sundhed og stabilitet. Påvirkningerne af malware kan variere:

- *mindre påvirkninger*: små forstyrrelser og systemlangsomhed,
- *alvorlige påvirkninger*: sletning af programmer eller data, ændring af indstillinger i operativsystem,
- *katastrofale påvirkninger*: ændring af data, omfattende forsinkelser og forstyrrelser, udspionering eller videresalg af følsomme data.

En *virus* er et lille stykke programkode, der blot ønsker at reproducere sig selv. For at gøre det inficerer virusen et andet program ved at vedhæfte en kopi af sin programkode i programmet. Udover at sprede sig selv og dermed tvinge folk til at spilde tid på fjerne den, kan virusen have direkte skadelige virkninger, som fx at slette data.

EKSEMPEL

**CREEPER**

Den første (veldokumenterede) virus dukkede op lang tid før internettet blev folkeeje: I starten af 1970'erne opdagede amerikanske militærfolk virusen Creeper på deres interne datanetværk, ARPA-nettet. Virusen gjorde ingen egentlig skade, men udskrev følgende besked på skærmen: "I'm the creeper! Catch me if you can!" – hvorefter den kopierede sig selv videre til yderligere computere på netværket.

*Fig. 9.149. Virus, orm og trojansk hest.*

I modsætning til en virus er *en orm* et *selvstændigt program*. Uafhængigt af andre programmer spreder ormen sig selv til andre computere – via fx adresselister i emailprogrammer.

**EKSEMPEL**

### DEN KÆRLIGHED, DEN KÆRLIGHED

En berømt computer-orm hed "I love you" og oversvømmede internettet i år 2000. Ormen ankom til intetanende ofre som en email med overskriften "I love you", tilsyneladende fra en bekendt i adresselisten. Nysgerrige emailbrugere, der åbnede den vedhæftede fil i emailen, blev inficeret. Ormen overskrev musik, film, tekstfiler mv. med teksten "I love you!" og sendte sig selv videre til alle i emailprogrammets adresseliste. Mere end 45 millioner computerbrugere verden over blev ramt. I Danmark blev computersystemer hos Tele Danmark, TV2 og Folketinget sat ud af drift.

*Spyware* er spion-software (deraf navnet). Spyware installerer sig på en computer uden brugerens viden. Hvor en virus- eller ormeinfektion typisk vil afsløre sig selv hurtigt, skjuler spywaren sig, følger med i brugerens færden på nettet og kan i princippet registrere al information, brugeren afgiver – fx kodeord og personfølsom data. Den indsamlede information bruges til at lave en personlig profil af brugeren. Spywaren sender som regel den oprettede profil til et tvivlsomt firma (som har lavet spywareprogrammet).

- I bedste fald bruger firmaet profilen til at målrette fx reklamer eller email og sender det til brugeren. Spywaren er altså skyld i spildtid og spild af computerkræfter.
- I værste fald sender offentliggør eller videresælger firmaet de personlige oplysninger, stjæler penge eller afpresser brugeren.

*Fig. 9.150. Spyware.*

*Trojanske heste* er opkaldt efter den berømte træhest, grækerne indtog byen Troja med: Grækerne byggede en stor, hul træhest. En håndfuld græske soldater gemte sig i den, mens resten af den græske hær gemte sig udenfor bymuren, ude af syne for trojanske udkigsposter. Trojanerne troede nu, at grækerne havde opgivet deres langvarige belejring af byen, var rejst hjem og havde efterladt hesten som afskedsgave. Byporten blev åbnet, hesten trukket ind og så blev sejren ellers fejret med betragtelige mængder vin. Samme nat kravlede grækerne lydløst ud fra hestens bug og åbnede byporten. Den udhvilede græske hær kunne da frit strømme ind i byen og besejre de døddrukne modstandere.

En trojansk hest er et ondsindet program (≈ grækere), som programmøren har gemt i et uskyldigt udseende program (≈ træhesten). Hvis man intetanende kører det "uskyldige" program, installeres det skjulte program, som derefter kan udføre uønskede handlinger. Det kunne fx

- opføre sig som spyware,
- oprette en "bagdør", som giver afsenderen adgang til computeren,
- sætte computeren ud af drift,
- slå antivirusprogrammer og firewalls fra, inden et større angreb sættes ind.

En trojansk hest spredes ved at godtroende brugere installerer "hesten" uden at opdage "grækerne" – på den måde er trojanske heste anderledes end virus og orme, der er installerer sig selv (virus og orme er *selv-replikerende*).

EKSEMPEL

**FALSK LOTTOSPIL**

En trojansk hest kunne fx ankomme i form af en email, angiveligt fra en kammerat, med en opfordring til at hente og installere et sjovt lille lottospil fra internettet. Spillet hentes og installeres, og alle er glade – et stykke tid. Imidlertid var emailens afsender ikke den kammerat, han udgav sig for at være – brugeren, der installerede spillet, var udsat for social engineering. Spillet er en trojansk hest og installerede en bagdør, da det blev installeret. Bagdøren giver den trojanske hests afsender fuld kontrol over den inficerede computer: Alle tastetryk (fx kodeord) kan overvåges, angriberen kan ændre systemfiler, lytte med på mikrofonen eller kigge med på den inficerede computers webcam.

### 9.3.6. SPAM

*Spam* er uønsket information, fx emails, reklamer eller pop-up-vinduer. Spam er ofte ganske ufarligt, men det irriterer og tager tid. Åbnes en spam-email, risikerer man ikke blot at spilde tid på fx løgnagtige tilbud i emailen, men også at der bliver sendt information tilbage til afsenderen om, at adressen er aktiv. Resultatet: mere spam.

### 9.3.7. KOMMUNIKATION OVER NETVÆRK

Netværkssikkerhed handler om at *kommunikere sikkert* over et netværk. Kommunikationen kan fx være udveksling af kærestebreve mellem hemmelige elskende eller HTTP-forespørgsler til en webserver. Alt efter sammenhængen kræver sikker kommunikation mellem aktørerne A og B, at forskellige (eller alle) af følgende sikkerhedsmål er opfyldt:

- *fortrolighed*: Uvedkommende kan ikke opsnappe kommunikationen.
- *integritet*:
  - *data-integritet*: Indholdet af en besked fra A til B kan ikke ændres under transporten over netværket.
  - *autenticitet*: En uvedkommende aktør C kan ikke udgive sig for at være A eller B.
  - *uafviselighed*: Hvis A sender B en besked, kan A ikke bagefter nægte det.
- *tilgængelighed*: Uvedkommende kan ikke afbryde A's og B's mulighed for at kommunikere.

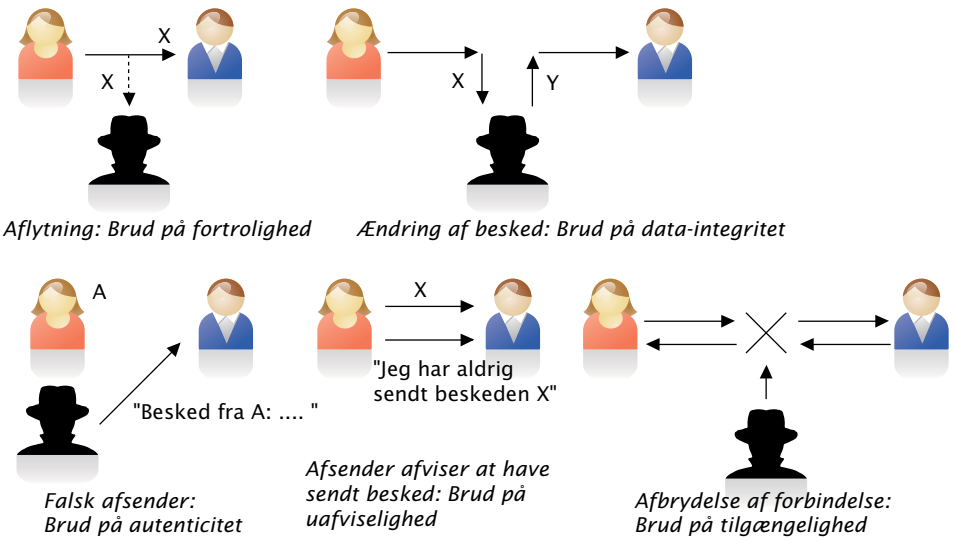


*Fig. 9.151. Trusler: Brud på fortrolighed, integritet og tilgængelighed.*

Inden du læser resten af dette afsnit, kan det være en god ide at repetere kapitel 5 (især begreberne protokol, port og netværkslag).

Mange netværk har sårbarheder, der muliggør trusler mod sikker kommunikation, fx:

- netværket kan bruge usikre protokoller,
- netværket kan tillade aflytning ude- eller indefra,
- netværket kan være "for åbent" – fx acceptere potentielt skadelig kommunikation på porte, der ikke bruges til standardformål.

Hvilke konkrete trusler findes der så?

*Packet sniffing*: Pakker i netværkslaget kan blive aflyttet eller kopieret, uden at det bliver opdaget. Ondsindede aktører kan dermed

- opsnappe fortrolig kommunikation,
- indsamle information om et IT-system og bruge det til at udføre et detaljeret angreb.

*Spoofing og phishing*: En ondsindet aktør udgiver sig for at være en anden, end han er. Man bruger som regel ordet spoofing om "aktivt" bedrag af denne art, fx email-henvendelser med falsk afsender, mens phishing bruges om mere "passivt" bedrag, fx at opretholde en falsk netbanks-webside, der lokker folk til at indtaste kreditkortinformationer.

*Replays:* Et replay-angreb forklares bedst ved et eksempel: Lad os sige, at A er en netbank-kunde og B en netbank. A henvender sig nu til netbanken og oplyser kodeord mv. A betaler sit kontingent til tennisklubben. Derefter logger A af netbanken og tager til tennis. Hvad A og B ikke har opdaget, er at hackeren C har "optaget" A's logon-information til netbanken B – ved packet sniffing. Lidt senere, mens A træner sin serv, "afspiller" C den del af "samtalen" mellem A og B, der bestod i, at A loggede på netbanken. Heraf navnet "replay" – C genafspiller dele af en kommunikation. Netbanken B opfatter C's henvendelse som om A henvender sig igen, og den giver adgang til A's konto. Resultatet er, at C har adgang til A's netbank!

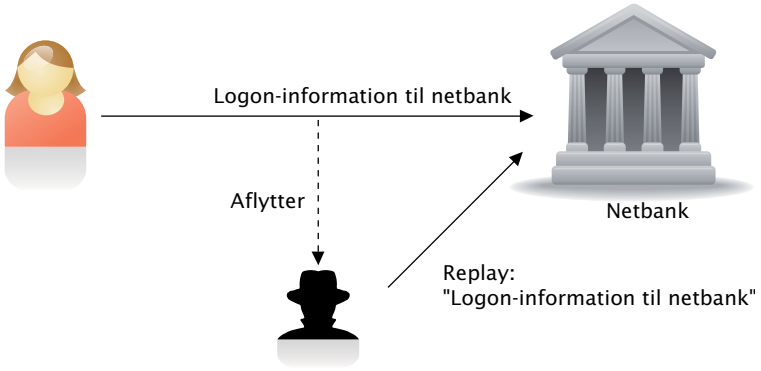


*Fig. 9.152. Replay-angreb.*

*Man in the middle-angreb (forkortet: MITM-angreb):* Hvis en ondsindet aktør C kan opsnappe al information, der udveksles mellem A og B, kan han lave et MITM-angreb – C er "manden i midten". Fx kan C opsnappe en besked fra A, ændre den, og sende den videre til B, uden at B opdager, at beskeden ikke længere er fra A.

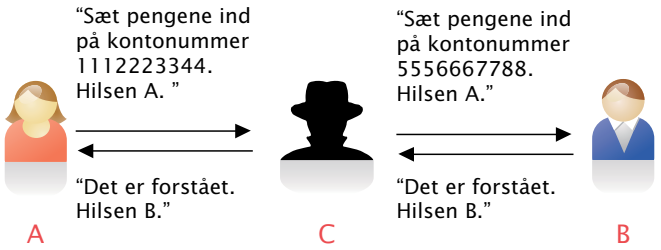


*Fig. 9.153. Man in the middle-angreb.*

Hvis en man-in-the-middle kaprer en forbindelse fuldstændigt, og fx smider B af, kaldes det en *session hijacking* (at "hijacke" betyder at kapre).

*DoS*: Ondsindede aktører kan umuliggøre kommunikation ved fx at "lægge en webserver ned". Det gøres ved at genere webserveren med så mange henvendelser, at den ikke kan passe sit job overfor andre klienter. Sådanne afbrydelser kaldes *Denial of Service* (DoS), fordi parter, der ønsker at kommunikere, nægtes adgang til det.

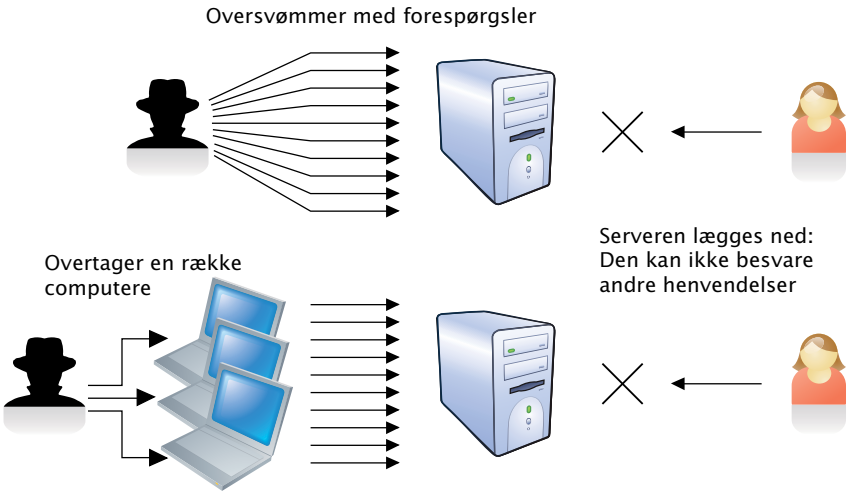


*Fig. 9.154. DOS og DDoS.*

En særlig ondsindet type DoS-angreb er såkaldte *distribuerede DoS-angreb* (DDoS), hvor en hacker kaprer en række computere til at hjælpe sig med et målrettet angreb.

### 9.3.8. TRÅDLØSE NETVÆRK

Trådløse netværk har ikke bare alle de samme sårbarheder som kabel-netværk, men også en lang række ekstra. Vi gennemgår nogle af de væsentligste sårbarheder og trusler:

Traditionelt har trådløse netværk *tilladt enhver at koble sig på* – uden nogen form for tjek. Denne sårbarhed giver angribere adgang til netværket.

Trådløse netværk er *lette at opdage, aflytte og koble sig på* – modsat et kabelnetværk, som man kun koble sig på med et kabel, der er i fysisk i kontakt med en netværksenhed. En bygning med et trådløst netværk har – med en metafor – "netværkskabler hængende ud af vinduerne i store klaser", og forbipasserende kan bare slutte ondsindede apparater til disse "netkabler".

Trådløse netværk er *lette ofre for DoS-angreb*: Et trådløst netværk er følsomt over for støj på de frekvensområder, der bruges til data-transmission.

*Ondsindede trådløse netværk kan opstilles*, fx nær eksisterende trådløse netværk. Hvis brugere ved en fejl benytter sig af det ondsindede netværk, fx i stedet for det netværk, de troede de brugte, kan de afsløre fortrolig information. Indenfor trådløse netværk omtaler man problemet – altså det, at man som aktør på et trådløst netværk ikke kan vide, om anden aktør er en "ven" eller en "fjende" – med det farverige navn "De Byzantinske Generalers Problem".

WWW

**TRÅDLØSE NETVÆRK**

På WWW finder du links til mere viden om trådløse netværk – og også en forklaring af, hvor de byzantinske generaler kommer ind i billedet.

## 9.4. MODMIDLER

For at mindske et IT-systems sårbarheder kan en lang række modmidler tages i brug. Modmidler kan groft sagt opdeles i

- *fysiske*, fx overvågningskameraer, pigtrådsindhegning og sprinkleranlæg,
- *tekniske*, fx antivirussoftware, firewalls og kryptering,
- *menneskelige*, fx agtpågivenhed og sikkerhedsprocedurer.

Yderligere kan modmidler inddeles i

- *protektive*: beskytter mod brud på sikkerheden,
- *detektive*: kan opdage brud på sikkerheden,
- *reaktive*: kan genoprette sikkerheden, hvis der er sket et brud på sikkerheden.

EKSEMPEL

**TYPER AF MODMIDLER**

En pigtrådsindhegning er fysisk-protektiv. Typisk antivirus-software er både teknisk-protektiv, teknisk-detektiv og teknisk-reaktiv. Sikkerhedsprocedurer, der følges til punkt og prikke af alle brugere, er et menneskeligt-protektivt modmiddel. Et sprinkleranlæg er fysisk-detektivt og fysisk-reaktivt.

### 9.4.1. AUTORISATION

Fortrolighed, integritet og tilgængelighed kan alle formuleres som egenskaber ved *autoriseret adgang* til information:

- *fortrolighed*: Kun personer og systemer, der er autoriseret til at læse informationen, har adgang til informationen.
- *integritet*: Kun personer og systemer, der er autoriseret til at ændre eller slette informationen, har adgang til informationen.
- *tilgængelighed*: Alle personer og systemer, der er autoriseret til at læse og/eller ændre informationen, kan komme til det.

En såkaldt *sikkerhedspolitik* for IT-systemet definerer, hvem der er autoriseret til at gøre hvad. Som udgangspunkt siger man, at en IT-politik bør følge princippet om *least privileges* (engelsk for "færrest privilegier"): Enhver aktør er kun autoriseret til netop det, han/hun/den/det har brug for at være autoriseret til.

**EKSEMPEL**

**LEAST PRIVILEGES**

Lad os kigge på IT-systemet "et operativsystem". Her fastlægger operativsystemet, hvilke autorisationer de enkelte brugerprogrammer har. En webbrowser (der er et brugerprogram) bør ifølge princippet om least privileges *ikke* være autoriseret til at ændre i systemfiler – det er ikke en nødvendig autorisation for at webbrowseren kan fremvise webindhold. Et operativsystem, der giver en webbrowser for kraftfulde autorisationer – fx adgang til at ændre systemfiler – udsætter sig selv for fare: Narres en bruger til at åbne en skadelig webside, kan websiden narre browseren til at ændre fatalt i systemfilerne. Det kan ikke ske med least privileges.

### 9.4.2. ADGANGSKONTROL

Vi har lige konstateret, at uautoriseret adgang er en meget generel trussel mod IT-sikkerhed. Det er derfor ingen overraskelse, at *adgangskontrol* er et centralt modmiddel. En adgangskontrol er protektiv og kan være af fysisk, teknisk eller menneskelig art.

Når en aktør vil tilgå et system med adgangskontrol, afgør adgangskontrollen, om aktøren er autoriseret:

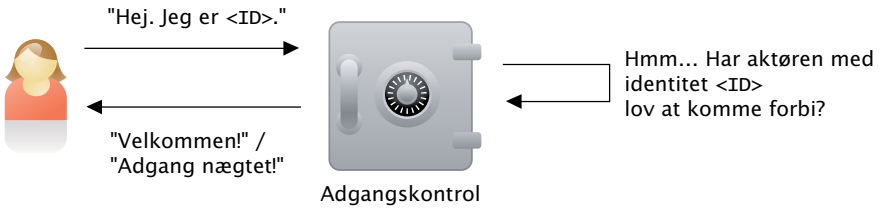


*Fig. 9.155. Autentificering 1.0 (protokol). 1) Aktøren oplyser sin identitet <ID>. 2) Adgangskontrollen tjekker, at aktøren med identitet <ID> har adgang til systemet.*

Adgangskontrollens afgørelse kaldes en *autentificering* – en autentificering kan enten resultere i et "velkommen" eller et "adgang nægtet".

De to skridt i en autentificering, som Figur 9.155 viser, er imidlertid ikke tilstrækkelige i sig selv: En ondsindet aktør A kan blot oplyse <B-ID> for en anden aktør B, der har de adgangsrettigheder, A ønsker at gøre brug af til onde formål. Derfor suppleres en identifikation altid med ekstrainformation, *autencitetsinformation*. Adgangskontrollen kan nu sammenholde identifikation med autencitetsinformation og derved afgøre, om aktøren virkelig er den, vedkommende giver sig ud for at være – altså, at det oplyste <ID> er autentisk.

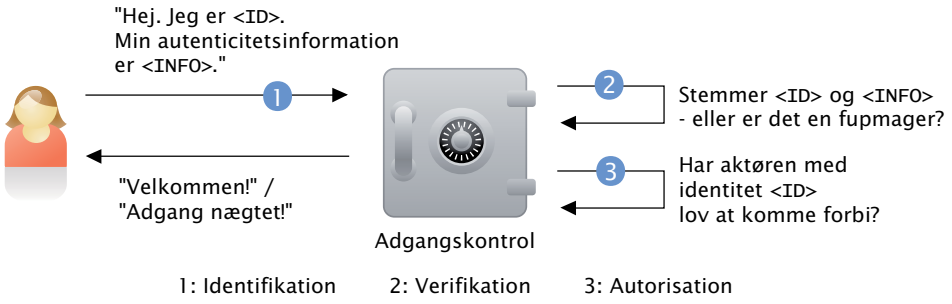


*Fig. 9.156. Autentificering 2.0 (protokol).*

Den reviderede autentificeringsprotokol har altså følgende tre skridt:

1. *Identifikation*: Aktøren oplyser identitet <ID> og autenticitetsinformation <INFO>.
2. *Verifikation*: Adgangskontrollen tjekker, at den oplyste autenticitetsinformation <INFO> stemmer med den oplyste identitet <ID>, sådan at aktørens identitet virkelig er <ID>.
3. *Autorisation*: Adgangskontrollen tjekker, at aktøren med identitet <ID> har adgang til systemet.

Vi opsummerer:

$$\textit{autentificering} = \textit{identifikation} + \textit{verifikation} + \textit{autorisation}.$$

En særlig type autenticitetsinformation er den digitale signatur (se afsnit 9.5.7).

**EKSEMPEL**

### BRUGERNAVN OG KODEORD

Mange brugersystemer tilgås gennem en adgangskontrol, hvor brugernavn og kodeord skal indtastes. Her består autentificeringen i

- identifikation = opgivelse af brugernavn <ID>,
- autenticitetsoplysning = opgivelse af kodeord <INFO>,
- verifikation = tjek af, at <ID> er en gyldig bruger, og at <ID> har kodeord <INFO>.

Hvis et ugyldigt brugernavn oplyses, vil adgangskontrollen nægte adgang med begrundelsen "bruger ukendt". Oplyses et korrekt brugernavn, men en forkert adgangskode, vil adgangskontrollen opfatte den oplyste identifikation som ikke-autentisk – fordi der er uoverenstemmelse mellem brugernavn og kodeord. I denne situation nægter adgangskontrollen adgang med begrundelsen "kodeord forkert".

**EKSEMPEL**

### BIOMETRISK AUTENTIFICERING

Et IT-system kan bruge biometrisk autentificering af mennesker. Det betyder, at unikke biologiske egenskaber måles som autenticitetsoplysniger, når mennesket præsenterer sin identitet. De biologiske egenskaber, der måles, kan fx være øjets irismønster eller fingeraftryk.

Den opmærksomme læser har allerede nu opdaget, at autentificeringsprotokollen Autentificering 2.0 i Figur 9.156 har et sikkerhedshul! Den er nemlig sårbar overfor *replay*-angreb, som vi hørte om i afsnit 9.3.7 (hvorfor?). For at undgå replay-angreb må man tage skrappe midler i brug – bl.a. kryptografi – vi kommer nærmere ind på hvordan i afsnit 9.5.6.

### 9.4.3. KRYPTOGRAFI

Du har sikkert hørt om kryptografi – læren om at kryptere. Kryptering er helt afgørende for IT-sikkerhed. Med rigtigt anvendt kryptering kan man opnå:

- fortrolighed,
- data-integritet,
- autenticitet af afsender- og modtager-identitet,
- uafviselighed.

EKSEMPEL

**LØNFORHØJELSE**

Du er ansat i Krypto Labs DK. En dag får du en email fra chefen, der fortæller dig, at din løn stiger til 100.000 kroner om måneden. Fordi der er brugt kryptering (på den rigtige måde), kan du være sikker på, at emailen faktisk kom fra din chef (autenticitet), at emailens indhold ikke er blevet ændret, siden din chef afsendte den (data-integritet), at ingen andre har læst emailen, mens den rejste fra din chefs computer over netværket til din computer (fortrolighed), og at din chef ikke kan nægte at have sendt emailen, når han senere ombestemmer sig (uafviselighed).

Med "rigtigt anvendt" menes, at kryptering skal bruges i en sikkerhedsprotokol uden sikkerhedshuller – ellers er krypteringen værdiløs. Vi kommer ind på kryptering i afsnit 9.5.

### 9.4.4. ANTIVIRUS-SOFTWARE

Et udbredt modmiddel mod truslen "malware" er antivirus-programmer, som kan opdage malware, inden det når at inficere en computer i et IT-system – eller kurere systemet, hvis det er blevet inficeret. Malwaren uskadeliggøres eller, bedre, fjernes.

Sådanne programmer har to primære metoder til at opdage malware:

- *Scan efter malware*: Systemet scannes jævnligt efter mønstre, der matcher virusdefinitioner i en opdateret oversigt over kendt malware.
- *Overvåg systemet*: Hvis mistænkelige begivenheder indtræffer – hvis fx filstørrelser ændres uden grund, eller programmer holder op med at virke.

Hvis en mistænkelig fil eller mistænkelig program-opførsel opdages, sørger antivirus-programmet for at uskadeliggøre filen eller programmet – eller alarmere brugeren.

### 9.4.5. FIREWALLS

Som vi ved, er der "gode" og "onde" aktører på internettet. Med et IT-system, der er koblet på internettet, er det afgørende at forhindre ondsindende aktører udefra i at tiltvinge sig adgang. Ligesom man i middelalderen beskyttede borge med en voldgrav og en vindebro, beskytter man nu om dage IT-systemer med *firewalls*. En firewall er protektiv.

ORDFORKLARING

**FIREWALL**

En kombination af hardware og software, der afgrænser et internet-opkoblet IT-system fra resten af internettet. Firewallen tillader noget trafik at passere og blokerer for anden trafik.

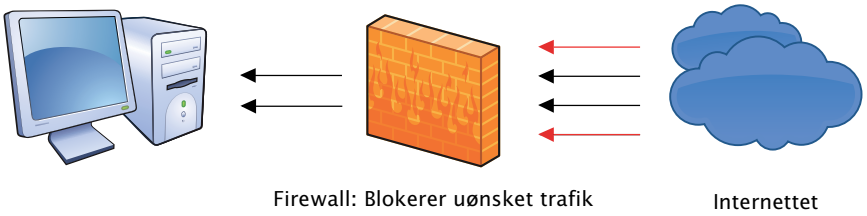


*Fig. 9.157. Firewall.*

Ideen med en firewall er, at den kun lukker "velkommen" trafik ind i IT-systemet – det kaldes *filtrering*, fordi trafikken sluses gennem et filter. En firewall frasorterer en stor del af den uønskede trafik, men kan aldrig give fuldstændig beskyttelse.

**TIL SÆRLIGT INTERESSEREDE**

**FILTRERING I FLERE LAG AF PROTOKOLSTAKKEN**

Firewalls filtrerer ofte på flere niveauer i protokolstakken:

- *pakkefiltrering i netværkslaget*: Firewall'en inspicerer de enkelte ankommende pakker. Hvis en pakke kommer fra en tvivlsom adresse, har et ukendt format, genkendes som skadelig, eller vil kontakte en port, der ikke ellers bruges, kasseres pakken.
- *beskedfiltrering i applikationslaget*: Det kan være nødvendigt at filtrere trafik på basis af fx enkeltbrugeres identiteter. Sådan filtrering kræver information fra applikationslaget.

Udover at filtrere fører de fleste firewalls en sikkerhedslog (se afsnit 9.4.10).

### 9.4.6. INTRUSION DETECTION-SYSTEMER (IDS)

Kan man ikke forhindre indtrængen, kan man forsøge at opdage forsøg på indtrængen. Et *intrusion detection system* (IDS) kan opdage brud på sikkerheden gennem overvågning – og er altså et detektivt modmiddel. Der findes mange typer IDS, bl.a.:

- *Fysisk IDS*: Opdager fysisk indtrængen. Fx er tyverialarmer og overvågningskameraer fysiske IDS'er.
- *Netværks-IDS*: Opdager uautoriseret indtrængen på et netværk ved at overvåge netværkstrafik.
- *Computerbaseret IDS*: Opdager uautoriseret indtrængen på en computer. Et antivirus-program er typisk et computerbaseret IDS.

### 9.4.7. SPAMFILTRE

*Spamfiltre* scanner indkommende emails med en opslagsteknik, der minder om virkemåden for antivirussoftware og IDS-systemer. Hvis en scanning resulterer i et match i en opdateret oversigt over kendt spam, kasseres emailen.

### 9.4.8. AWARENESS

De fleste brud på IT-sikkerhed skyldes menneskelige fejl. Menneskelige fejl kan i et vist omfang undgås ved at uddanne og oplyse brugere af et IT-system om, hvilke trusler der lurer. En bruger, der forstår farer og risici, får skærpet sin agtpågivenhed – og opfører sig dermed mere sikkert og bliver sværere at narre til fx at udføre falske ordrer. Denne agtpågivenhed kaldes med et engelsk ord for *awareness*.

Folk, der fremstiller IT-systemer, højner IT-systemets sikkerhed ved fra start at udvise awareness. Fx laver programmører, der er opmærksomme på at programmere sikkert, mere sikre programmer: De anvender sunde principper for softwarearkitektur, undgår kendte sikkerhedsproblemer og tester programmer for at undgå bugs, der sidenhen kan udnyttes af ondsindede aktører.

### 9.4.9. FYSISK SIKKERHED

Fysiske sikkerhedsforanstaltninger beskytter et IT-system mod ulykker som indbrud, gasudslip eller ildebrand på det sted, IT-systemet fysisk er. Fysiske sårbarheder mindskes med åbenlyse modmidler som fx fysiske adgangskontroller – det kunne være svært bevæbnede dørvagter. Også detektive og reaktive modmidler kan tages i brug, fx overvågningskameraer, nødudgange, backup-servere, ildslukkere og nødstrømanlæg.

### 9.4.10. SPORBARHED

Som vi har set, kan en adgangskontrol medvirke til at forhindre uønskede begivenheder. Virkeligheden viser desværre, at vi ikke med 100 % sikkerhed kan forhindre alle uønskede begivenheder:

- selvom kun autoriserede handlinger udføres i et system, kan det alligevel føre til brud på sikkerheden
- enhver adgangskontrol, uanset hvor smart og gennemtænkt den er, kan snydes af en udspekuleret misdæder

Et fornuftigt ekstra krav at stille indenfor IT-sikkerhed er derfor, at enhver aktør stilles til regnskab for sine egne handlinger. Det kræver viden om, *hvem der har gjort hvad hvornår*.

**ORDFORKLARING**

### SPORBARHED

Hver gang en begivenhed indtræffer, der er relevant i forhold til et systems IT-sikkerhed, noteres begivenhed, tidspunkt og aktør i en særlig systemfil, *sikkerhedsloggen*.

Sikkerhedsloggen er en slags "sikkerheds-dagbog" for systemet.

**EKSEMPEL**

### SPORING

Følgende er eksempler på ting, det kunne være relevant at logge i en sikkerhedslog for et lokalnetværk af computere på fx en skole:

- Tidspunkt, dato og logon-ID for alle forsøg på at logge på en computer.
- Brug af administrator-konti.
- Brug af hardware – hvem har printet hvornår, hvem har haft en CD i en computer hvornår, mv.
- Hvilke systemfiler der åbnes og lukkes hvornår.

Listen kunne fortsættes. Det er op til det enkelte systems sikkerhedsansvarlige at fastlægge, præcis hvad der skal logges. En meget omfattende sikkerhedslog kunne fx endda medtage "hvilke taster der trykkes på hvornår".

Med sporbarhed kan man efter et eventuelt brud på sikkerheden *spore*, hvem der gjorde hvad hvornår. Det kræver blot et opslag i sikkerhedsloggen. Er der foregået ulovligheder, fx, kan sikkerhedsloggen bruges til bevisførelse.

For at implementere sporbarhed skal systemet kunne *identificere* og *autentificere* aktører.

## 9.5. KRYPTOGRAFI

Kryptografi er læren om at *kryptere data*. Krypteret data er tilsløret: Kun afsender og de modtagere, der ved, hvordan data kan *dekrypteres*, kan forstå indholdet. Bliver krypteret data opsnappet af en, der aflytter forbindelsen, får vedkommende ikke noget ud af det.

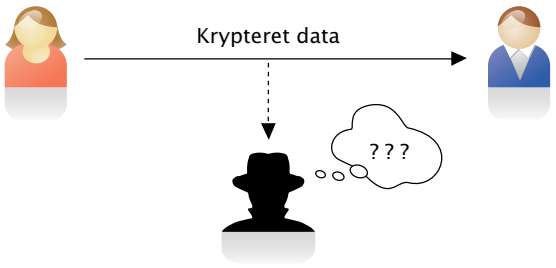


*Fig. 9.158. Kryptering.*

### 9.5.1. EKSEMPEL: KRYPTERING À LA CÆSAR

Allerede den romerske kejser Julius Cæsar brugte kryptering, når han sendte taktiske ordrer med kurer mellem sine udsendte generaler. På den måde kunne hverken en spion, der udgav sig for at være kurer, eller overfaldsmænd, der bemægtigede sig den krypterede besked, aflure beskeden. Cæsars teknik var simpel – han erstattede simpelt hen hvert bogstav i sin besked med bogstavet tre gange længere henne i alfabetet:

```
Før kryptering:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
Efter kryptering: DEFGHIJKLMNOPQRSTUVWXYZABC
```

Hvis vi koder teksten "Top secret!" med Cæsars metode, får vi den krypterede tekst "Wrs vhfuhw!". Uforståeligt – medmindre man kender krypteringsmåden.

Det er klart, at man også kunne kryptere ved at forskubbe et andet antal pladser end lige netop 3.

### 9.5.2. KLARTEKST, CIFFERTEKST OG NØGLE

Ikke-krypterede beskeder er som postkort: Uvedkommende kan læse indholdet under transporten. Krypterede beskeder er som breve i lukkede kuverter: Uvedkommende kan ikke læse indholdet (men kan som regel se afsender og modtager).

Til kryptering bruges en *krypteringsalgoritme*. Man bruger primært kryptering der ikke afhænger af, at krypteringsalgoritmen er hemmelig. Princippet om, at krypteringsalgoritmen bør være offentligt kendt, kaldes for *Kerckhoffs princip*.

Lad os sige, at Alice vil sende en besked til Bob, fx "Jeg elsker dig, Bob." (Alice og Bob er tilbagevendende figurer i litteratur om kryptografi). Beskeden, inden den er blevet krypteret, kaldes for *klarteksten*. Alice krypterer nu sin besked med en eller anden krypteringsalgoritme, sådan at den resulterende *ciffertekst* er uforståelig for udenforstående, fx aflyttersken Eva. Idet krypteringsalgoritmen er offentligt kendt, bliver Alice nødt til at bruge en eller anden hemmelig information, hvis hun vil undgå, at Eva dekrypterer beske-

den. Denne hemmelige information kaldes *nøglen*. Nøglen bruges til at "låse" cifferteksten med – Eva kender ikke nøglen, der fx kan være et tal, et stykke tekst eller et kodeord. For at dekryptere cifferteksten til klartekst, skal Bob bruge den rigtige nøgle til at "låse cifferteksten op" med.

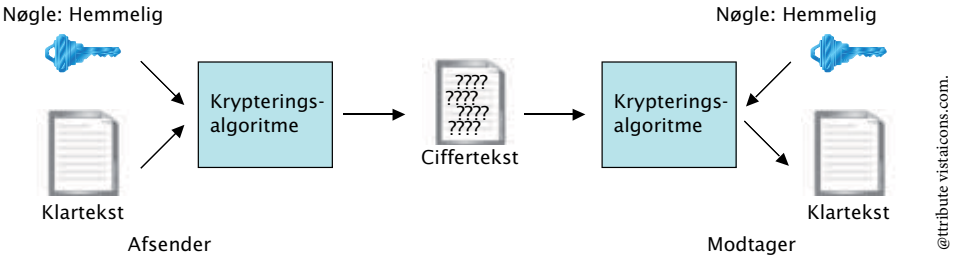


*Fig. 9.159. Kryptering: Inden afsendelse krypteres klartekst til ciffertekst med nøglen. Ved modtagelse dekrypteres cifferteksten med nøglen.*

En krypteringsalgoritme tager altså to input: Klartekst og nøgle, og producerer et output: cifferteksten. Vil man dekryptere, bruges også to input: Ciffertekst og nøgle. Outputtet er da klarteksten.

**EKSEMPEL**

### CÆSARKRYPTERING

I forrige afsnit krypterede vi beskeden "Top secret!" med Cæsars metode og fik den uforståelige tekst "Wrs vhfuhw!" ud af det. Klarteksten er i dette tilfælde "Top secret!", nøglen er tallet 3, mens cifferteksten er "Wrs vhfuhw!". For at dekryptere cifferteksten, skal vi blot bruge nøglen (tallet 3) – så kan vi "gå den anden vej".

## 9.5.3. SYMMETRISK KRYPTERING

Cæsars kryptering er *symmetrisk*.

**ORDFORKLARING**

### SYMMETRISK KRYPTERING

Symmetrisk kryptering er kryptering, hvor afsender og modtager skal bruge den samme nøgle: Afsender krypterer med nøglen, modtager dekrypterer med nøglen.

Sikkerheden ved en symmetrisk krypteringsalgoritme afhænger af, at nøglen svær at gætte. Jo mere kompleks nøgle, desto sværere er det at knække koden. Til gengæld tager kryptering og dekryptering længere tid. Cæsars algoritme har en simpel nøgle og er dermed let at gætte. Der er kun 29 forskellige nøgler (hvorfor?), og de er hurtigt gennemprøvet.

EKSEMPEL

### DES OG AES: FORTROLIGHED

DES (Data Encryption Standard) er en symmetrisk krypteringsalgoritme, der blev lanceret i 1977. DES bruger et 56-bit lang *binært tal* som nøgle. Der er altså $2^{55}$, eller lidt over 70 millioner millarder, forskellige DES-nøgler. Man skulle tro, at DES dermed er ubrydelig, men et effektivt kryptoanalysehold dekrypterede en DES-besked på mindre end 4 måneder i 1997. Siden voksede computerkraften, og i 1999 behøvedes kun 22 timer for at dekryptere en DES-besked. For at imødegå ondsindede personer med masser af computerkraft til at knække DES-krypterede beskeder, lancerede NIST (en amerikansk organisation for tekniske standarder) i 2001 efterfølgeren til DES: AES (Advanced Encryption Standard). AES bruger nøgler med bitlængde 128, 156 eller 256. En maskine, der kan knække en DES-krypteret besked på 1 sekund, skal bruge omtrent 150 billioner år på at knække en besked, der er AES-krypteret med en 128-bits nøgle.

WWW

### KERBEROS: AUTENTIFICERING

På **WWW** kigger vi på en udbredt protokol til autentificering, der bygger på symmetrisk kryptering, Kerberos (protokollen er opkaldt efter den trehovedede hunde-skildvagt, der i den græske mytologi bevogter indgangen til underverdenen).

## 9.5.4. ASYMMETRISK KRYPTERING

Symmetrisk kryptering forudsætter, at afsender og modtager i al hemmelighed har kunnet blive enige om, hvilken nøgle der skal bruges. Men det kræver jo (sikker) kommunikation! Hvor Cæsar og hans generaler kunne mødes over en gladiatorkamp og aftale nøgler, mens tilskuernes opmærksomhed var rettet mod arenaen, er det meget sjældent tilfældet, at folk der skal sende krypteret data til hinanden over fx internettet, kan mødes i det virkelige liv og aftale hemmelige nøgler først. Hvem ved, måske bor Alice i Grønland og Bob i Tunesien – og de skal udveksle krypteret data med det samme – hvad gør de?

Svaret er *asymmetrisk kryptering* (en genial teknik opfundet i 1970'erne), hvor afsender og modtager ikke behøver at have udvekslet en hemmelig nøgle for at kunne kommunikere sikkert. Asymmetrisk kryptering kan ikke bare bruges til fortrolig kommunikation, men også til autentificering og digitale signaturer.

ORDFORKLARING

### ASYMMETRISK KRYPTERING

Kryptering, hvor afsender og modtager hver har sit eget sæt nøgler, en *privat nøgle* og en *offentlig nøgle*. Den private nøgle er hemmelig og kun kendt af ejermanden, mens alle har adgang til at se den offentlige nøgle. Data krypteret med den offentlige nøgle kan kun dekrypteres med den private nøgle. Data krypteret med den private nøgle kan kun dekrypteres med den offentlige nøgle.

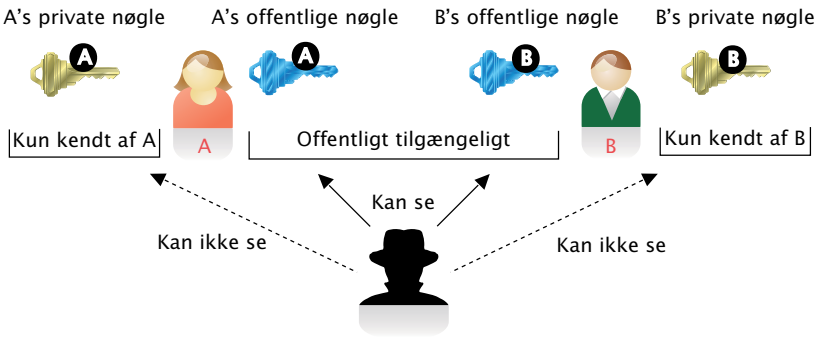


*Fig. 9.160. Offentlig og privat nøgle.*

**TIP**

**PUBLIC KEY-KRYPTERING**

I kraft af, at asymmetrisk kryptering hviler på brugen af offentlige nøgler, modsat symmetrisk kryptering, hvor der kun er hemmelige nøgler, kaldes asymmetrisk kryptering også for *public key-kryptering* (public betyder offentlig på engelsk).

Hvis Alice og Bob kommunikerer med asymmetrisk kryptering, er der altså i alt 4 nøgler i spil (modsat asymmetrisk kryptering, hvor der var 1 nøgle): Alices private nøgle, Alices offentlige nøgle, Bobs private nøgle og Bobs offentlige nøgle.

Vi tager et eksempel på asymmetrisk kryptering:

1. Alice vil sende en besked, L, til Bob. Hun henter derfor Bobs offentlige nøgle, $Bob_{offentlig}$. Den kan alle hente, idet den er offentlig.
2. Alice krypterer sin besked, L, ved hjælp af Bobs nøgle, og får cifferteksten $Bob_{offentlig}(L)$. Hun bruger (i overensstemmelse med Kerckhoffs princip) en velkendt krypteringsalgoritme.
3. Alice sender den krypterede besked til Bob.
4. Bob modtager $Bob_{offentlig}(L)$. Han kender, som den eneste, sin private nøgle, $Bob_{privat}$. Bob kan derfor dekryptere $Bob_{offentlig}(L)$. Det gør han ved at beregne $Bob_{privat}(Bob_{offentlig}(L)) = L$.
5. Hvis Bob beslutter sig for at svare, bruger han den helt tilsvarende procedure: han henter først Alices offentlige nøgle, krypterer sit svar med den, og sender den krypterede besked til Alice. Fordi Alice kender sin private nøgle, kan hun dekryptere Bobs besked.

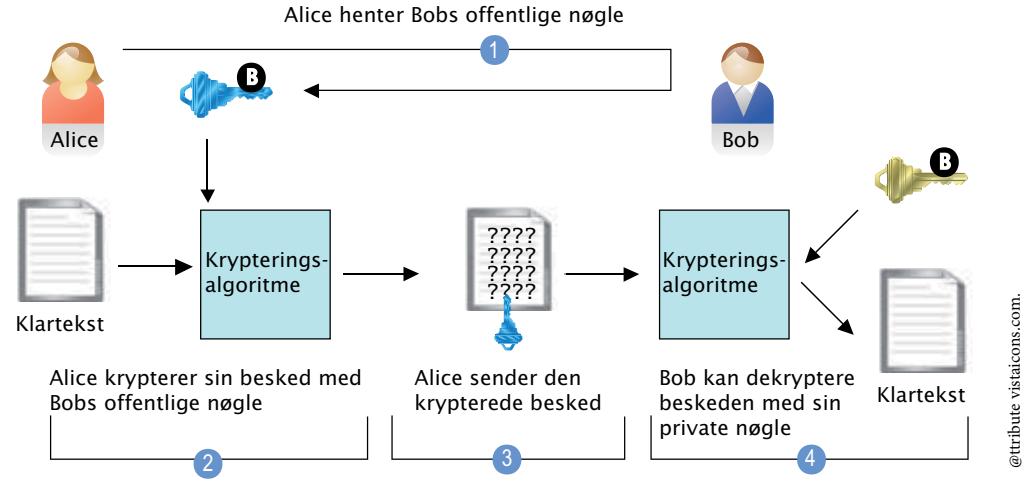


*Fig. 9.161. Asymmetrisk kryptering med privat og offentlig nøgle.*

**ANALOGI**

### AFLÅSTE KISTER

Dronning Alice af Alicien vil gerne sende et hemmeligt brev til Kong Bob af Bobbistan. Desværre har dronning Alice kun en mistænkelig kurer, Igor, til sin rådighed. Igor kunne sagtens finde på selv at læse brevet undervejs eller udlevere det til fremmede. Hun sender derfor Igor til Bobbistan for at hente Kong Bobs hængelås. Igor adlyder. I Bobbistan udleverer Kong Bob sin hængelås (åben). Hængelåsen er ikke i sig selv værdifuld, så Igor stjæler den ikke på hjemvejen. Hjemme i Alicien putter Dronning Alice brevet ned i en kiste, spænder jernkæder om kisten og låser kæderne fast med Kong Bobs hængelås. Dronning Alice sender Igor afsted med den aflåste kiste. Igor kan ikke åbne kisten, og undlader derfor at stjæle den – det ville han alligevel ikke få noget ud af. Møder han mistænkelige fremmede på rejsen, vil de heller ikke kunne åbne kisten og se brevet derinde. Vel fremme i Bobbistan modtager Kong Bob kisten. Efter at have sendt Igor hjemad, fremdrager Kong Bob sin nøgle, sætter den i hængelåsen – den passer! – og låser op. Brevet er nået sikkert frem.

Fortællingen kan bruges til at forstå asymmetrisk kryptering med:

| | | |
|---|---|---|
| Hemmeligt brev | ≈ | besked, Alice vil sende til Bob |
| Kong Bobs hængelås | ≈ | Bobs offentlige nøgle |
| Kiste låst med Kong Bobs hængelås | ≈ | besked krypteret med Bobs offentlige nøgle |
| Kong Bobs nøgle | ≈ | Bobs private nøgle |
| Igor | ≈ | transport over et netværk, der måske bliver aflyttet |

**EKSEMPEL**

**RSA**

RSA er en asymmetrisk krypteringsalgoritme (opkaldt efter opfinderne Ron Rivest, Adi Shamir og Leonard Adleman). RSA er en globalt accepteret standard for kryptering.

### 9.5.5. AFTALE AF SYMMETRISK NØGLE: DIFFIE-HELLMAN

Asymmetrisk kryptering kan bruges til at aftale en hemmelig symmetrisk nøgle med. Til formålet bruges *Diffie-Hellman-protokollen*, opfundet af – ja … – Diffie og Hellman.

Lad os sige, at Alice og Bob ønsker at aftale en symmetrisk nøgle til at udveksle data fortroligt med. De gennemfører derfor skridtene i Diffie-Hellman-protokollen:

1. Alice og Bob udveksler offentlige nøgler – selvom netværket er usikkert, er der ingen problemer, fordi nøglerne er offentlige.
2. Alices Diffie-Hellman-program tager hendes private nøgle og Bobs offentlige nøgle og beregner et tal K1 ved hjælp af Diffie-Hellman-algoritmen.
3. Bobs Diffie-Hellman-program tager hans private nøgle og Alices offentlige nøgle og beregner et tal K2 ved hjælp af Diffie-Hellman-algoritmen.
4. Fordi Diffie-Hellman-algoritmen er baseret på smuk og smart matematik, gælder
   a. K1 = K2.
   b. En ondsindet aktør, der har aflyttet udvekslingen af de offentlige nøgler, kan ikke beregne tallet K1 – for at gøre det, skal han bruge enten Bobs eller Alices private nøgle – og den har han ikke.

Fordi der gælder lige præcis både a og b, kan Alice og Bob bruge den beregnede værdi som symmetrisk nøgle.

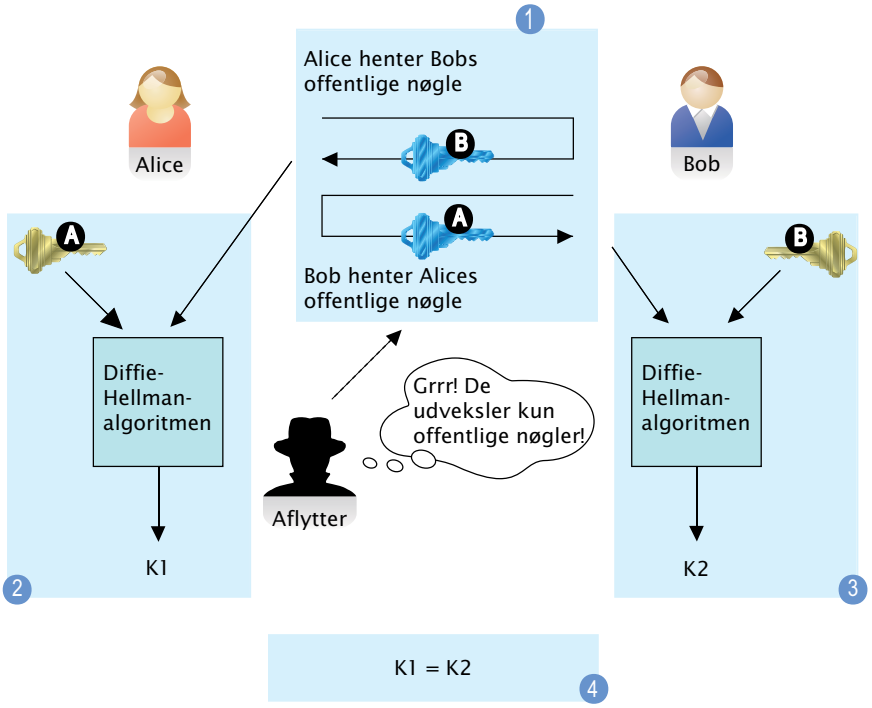


*Fig. 9.162. Diffie-Hellman.*

### 9.5.6. AUTENTIFICERING MED RANDOM CHALLENGE

Vi husker autentificeringsprotokollen i Figur 9.156. For at sikre protokollen mod replay-angreb må den ændres. Vi forsyner derfor vores autentificeringsprotokol med en mekanisme til *random challenge* og *response*.


*Fig. 9.163. Alice autentificerer sig overfor Bob med challenge-response. Ondsindede aktører, der giver sig ud for at være Alice, har ikke en chance.*

Til hver *challenge* svarer et rigtigt *response*. Lad os sige, at Alice vil autentificere sig overfor Bob. Alice henvender sig hos Bob og siger "Jeg er Alice. Jeg vil gerne autentificeres." Bob svarer med en random challenge – en tilfældig udfordring – som kun Alice kender det rigtige svar (response) til. Alice sender sit svar til Bob. Fordi svaret er sandt, ved Bob, at det må være Alice han taler med.

Ideen med random challenge og response er, at hver gang Alice vil autentificere sig, får hun en ny udfordring – og derfor er det nu et nyt svar, der er det eneste acceptable svar. Da udfordringen udvælges tilfældigt af Bob, kan hverken Alice eller andre udregne på forhånd, hvilken udfordring der skal svares på. En aflytter kan altså ikke bevidstløst lave et replay-angreb med Alices response, fordi et response ikke kan genbruges.

**EKSEMPEL**

### CHALLENGE-RESPONSE

Alice vil autentificere sig over for Bob. Bob udfordrer derfor med den tilfældigt valgte udfordring "Hvad er navnet på den bamse, du havde som 8-årig?" (challenge). Alice svarer "Bamse-Lillian!" (response). Bob konstaterer, at svaret er sandt. Samtidig ved han, at kun Alice kender til Bamse-Lillian, så det må altså være Alice, han taler med. Næste gang Alice vil autentificere sig, giver Bob hende spørgsmålet "Hvad var det første ord, du lærte på fransk?". Alice svarer denne gang med "crayon". Bob konstaterer at svaret er sandt. Samtidig ved han, at kun Alice kender svaret, så det må være Alice han snakker med. Svaret "Bamse-Lillian" er helt klart et forkert svar på challenge nummer to. En replay-angriber ville derfor ikke med succes kunne udgive sig for at være Alice.

I praksis laves random challenge-response med en kompliceret kombination af asymmetrisk og symmetrisk kryptering. Det vigtige her er blot at forstå princippet.

### 9.5.7. DATA-INTEGRITET MED HASHING

På internettet bruges en checksum til at opdage, om en pakke er blevet ændret under sin rejse fra afsender til modtager. En tilsvarende teknik, *hashing* (af det engelske ord *hash*, der betyder noget i retning af hakkelse), bruges til at sikre data-integritet af beskeder.

**ORDFORKLARING**

### HASH-FUNKTION

Til hashing bruges en *hash-funktion*. Hash-funktionen er en algoritme, der tager en besked (kort eller lang) som input og leverer en hash-værdi af en bestemt størrelse som output. En hash-funktion beregner altid samme output-hash-værdi for den samme input-besked. En hash-funktion er *en-vejs*: Udfra en hash-værdi kan den oprindelige besked ikke genskabes.

**EKSEMPEL**

### HASHING

Alexander vil sende beskeden "Du er sød! Alexander" til Malene over et usikkert netværk. For at sikre sig, at Malene ikke modtager en forkert besked, hasher Alexander beskeden med en hash-funktion og beregner hash-værdien "507113e9". Alexander sender nu både besked og hash-værdi til Malene. Beskeden når frem til Malene, og hun beregner hash-værdien af den modtagne besked. Hvis hun får hash-værdien "507113e9", kan hun sammenligne med Alexanders medsendte hash-værdi og konstatere, at beskeden er nået uændret frem. Hvis hun derimod får en anden hash-værdi, må beskeden være blevet ændret under sin rejse på netværket. I dette tilfælde kasserer Malene beskeden.

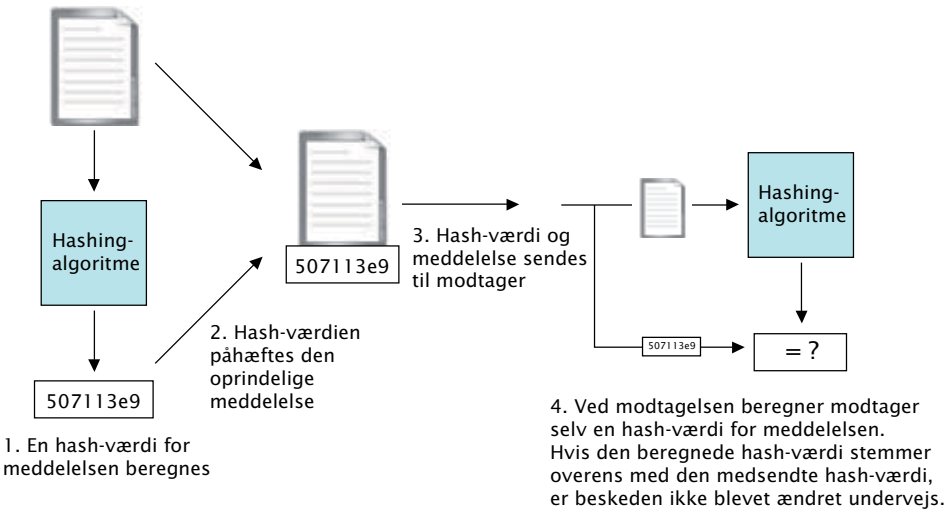




*Fig. 9.164. Hashing.*

Hashing bruger ingen nøgler. Hashing-algoritmen er heller ikke hemmelig – i overensstemmelse med Kerckhoffs princip. Fidusen med hashing er, at det er en en-vejs-proces. Opsnapper en ondsindet aktør en hash-værdi, fx " aac16ea6", *kan han ikke genskabe den besked*, der gav hash-værdien.

Der er altså en afgørende forskel på, hvordan kryptering med nøgler fungerer, og så hashing. Med den rette nøgle kan man dekryptere en krypteret besked. Ingen kan nogensinde "dekryptere" en hash-værdi og få den originale besked tilbage.

**TIL SÆRLIGT INTERESSEREDE**

### KOLLISIONSFRI HASH-FUNKTIONER

Der findes mange anvendte hash-funktioner. En god hash-funktion er *kollisionsfri*, dvs, forelagt en besked og dens hash-værdi er det *umuligt* at finde en anden besked med samme hashværdi.

## 9.5.8. DIGITAL SIGNATUR

En *digital signatur* er en elektronisk underskrift. Den identificerer og autentificerer underskriveren, og den er uafviselig. Sagt på dansk: Man skal kunne afgøre, om en digital signatur tilhører en bestemt person, og signaturen må ikke kunne kopieres.

**EKSEMPEL**

### BRUG AF DIGITAL SIGNATUR

Digitale signaturer bruges overalt, hvor der indgås kontrakter eller økonomiske aftaler over netværk, fx mellem personer og banker, mellem banker og banker, eller mellem virksomheder.

Inden vi ser på, *hvordan* man laver en digital signatur, kigger vi på et eksempel, der illustrerer, *hvorfor* digitale signaturer er nødvendige. Hashing-teknikker alene er nemlig ikke nok til at sikre besked-autenticitet:

**EKSEMPEL**

### HASHING UDEN AUTENTIFICERING

Alexander sender (besked,hash-værdi)-parret

```
("Du er sød! Alexander", " 507113e9")
```

til Malene over et usikkert netværk. Erika, der er jaloux, opsnapper både besked og hash-værdi. Hun ændrer beskeden til `"Jeg hader dig, Malene. Alexander"` og beregner en ny hash-værdi, `"029a159e"`, svarende til den nye besked. Erika sender nu (besked,hash-værdi)-parret

```
("Jeg hader dig, Malene. Alexander", " 029a159e")
```

til Malene. Malene modtager beskeden, beregner hash-værdien `"029a159e"` og tror nu – fordi hash-værdierne stemmer overens – at beskeden ikke er blevet ændret undervejs.

Vil man sikre sig, at beskedens afsender er den rigtige, må *digitale signaturer* tages i brug.

En digital signatur implementeres med en kombination af asymmetrisk kryptografi og hashing:

**ORDFORKLARING**

**DIGITAL SIGNATUR**

En digital signatur på en besked er en hash-værdi af beskeden, krypteret med afsenderens private nøgle.

Ideen er, at *hvis* kan man dekryptere hash-værdien med en eller andens offentlige nøgle, så *må* den "en eller anden" lige præcis være afsenderen. Signeringen sikrer på den måde autentificering og uafviselighed. Hashingen sikrer data-integritet.

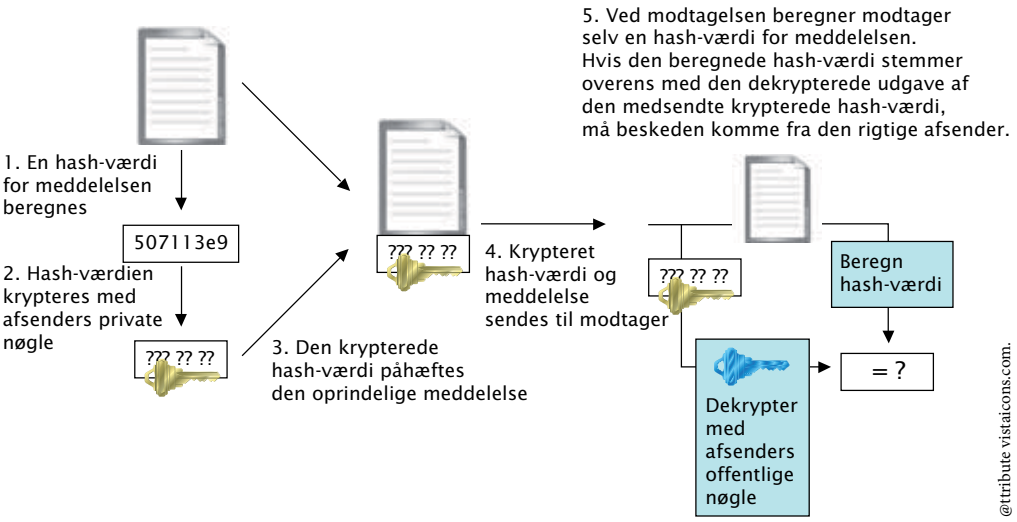


*Fig. 9.165. Digital signatur.*

Vi illustrerer ideen med et eksempel.

**EKSEMPEL**

### DIGITAL SIGNATUR

Alexander vil sende beskeden "Du er sød! Alexander" til Malene. For at sikre dataintegritet og for at Malene ved, at beskeden er fra ham, gør Alexander følgende:

1. Alexander beregner hash-værdien "507113e9" af beskeden "Du er sød! Alexander".
2. Han krypterer hash-værdien "507113e9" med sin private nøgle og får den krypterede hash-værdi "bdb74befc887188". Kun Alexanders offentlige nøgle kan dekryptere den krypterede hash-værdi.
3. Alexander sender (besked, krypteret hash-værdi)-parret

   ("Du er sød! Alexander", "bdb74befc887188")

   til Malene.
4. Malene beregner først hash-værdien af den modtagne besked – hun får hash-værdien "507113e9", fordi en hash-funktion altid giver den samme hash-værdi for den samme input-besked.
5. Malene dekrypterer herefter den modtagne krypterede hash-værdi "bdb74befc887188" med Alexanders offentlige nøgle. Det giver hende værdien "507113e9".
6. Fordi den modtagne og den beregnede hash-værdi er ens, er beskeden ikke blevet ændret undervejs. Fordi Alexanders offentlige nøgle kunne bruges til at dekryptere med, *må* beskeden være fra ham.

### 9.5.9. PKI

Man skulle tro, at alle vore trængsler vedrørende sikker kommunikation var overstået, nu hvor vi både har kryptering, hashing og digitale signaturer. Der er imidlertid én ting, vi har haft som stiltiende antagelse gennem hele krypteringsafsnittet: Nemlig at

*afsender og modtager stoler på hinanden.*

Hvad nu hvis Josefine vil kommunikere med Stefan, men ikke er sikker på, om Stefan er ondsindet – er han til rødvin og madvarer eller røveri og malware?

Løsningen på problemet er at lave en *public key-infrastruktur* (*PKI*). En PKI er et system, der tager højde for "hvordan aktører stoler på hinanden". De vigtigste elementer i PKI er

- certifikater,
- certifikat-autoriteter (CA'er).

Hver aktør (fx Josefine) i et PKI registreres hos en certifikat-autoritet, der mod konkrete oplysninger om fx navn, adresse, mv., udsteder et personligt *certifikat*. Vil Josefine snakke

med Stefan, spørger hun sin certifikat-autoritet, om Stefan er registreret der. Hvis han er, får Josefine Stefans certifikat tilsendt – det indeholder Stefans offentlige nøgle – og Josefine kan derefter snakke med Stefan. Josefine vælger altså at stole på de aktører, der er registreret hos hendes certifikat-autoritet.

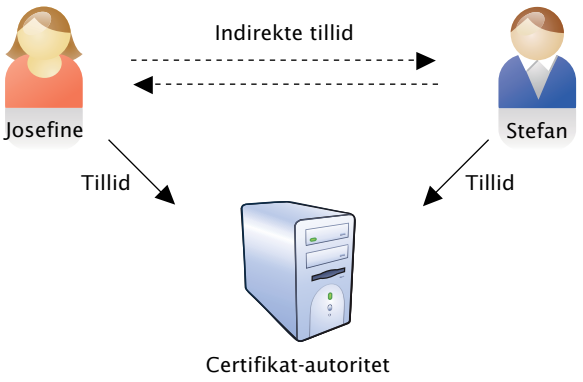


*Fig. 9.166. Josefine stoler på sin CA. Stefan stoler på sin CA. Derfor stoler Josefine og Stefan indirekte på hinanden.*

### 9.5.10. KRYPTOGRAFISK OPSUMMERING

Vi opsummerer alle de mange kryptografiske begreber.

| ... sikrer ... | fortrolighed | data-integritet | autenticitet | uafviselighed |
|---|---|---|---|---|
| Kryptering ... | X | - | - | - |
| Hashing ... | - | X | - | - |
| Digital signatur ... | - | X | X | X |
| Kryptering og digital signatur ... | X | X | X | X |

*Fig. 9.167. Ikke alle kryptografiske teknikker sikrer alle IT-sikkerhedsmål. Med de rette kombinationer af kryptografiske teknikker kan man opnå de sikkerhedsmål, en given beskedudveksling kræver.*

## 9.6. REALISERING AF IT-SIKKERHED

Som allerede nævnt kan et IT-system aldrig blive 100 % sikkert. Det er derfor op til sikkerhedspolitikken for et givent IT-system at fastlægge hvor meget anstrengelse der skal bruges på IT-sikkerhed. Jo mere sikkerhed, jo større omkostninger.

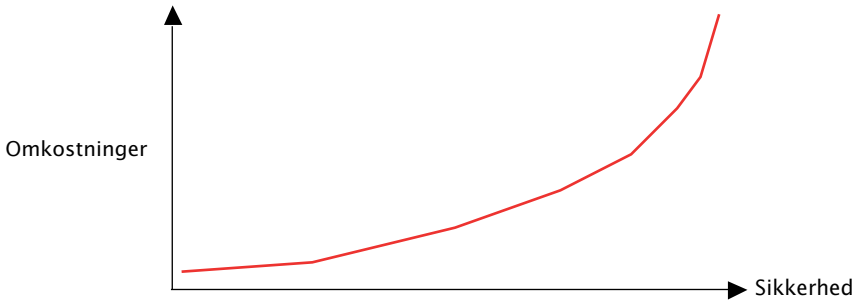


*Fig. 9.168. Omkostninger og sikkerhed.*

### 9.6.1. OMKOSTNINGER

Ethvert IT-system har brugere. Brugerne, der ikke nødvendigvis ved noget om IT-sikkerhed, har alligevel bestemte IT-sikkerhedsmæssige behov. Samtidig er det klart, at jo flere sikkerhedsprocedurer, brugere af system skal lære at bruge, desto sværere er det at bruge systemet. Et godt system har derfor en fornuftig balance mellem at være *sikkert* og at være *let at bruge*.

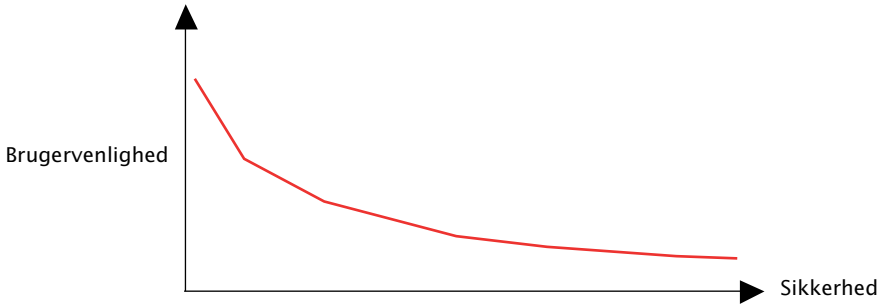


*Fig. 9.169. Et system uden nogen sikkerhedsforanstaltninger er let at bruge, men meget sårbart. Et system med mange sikkerhedsforanstaltninger er svært at bruge, men ikke så sårbart.*

Hvis et system har sikkerhedsprocedurer eller -foranstaltninger, der på klodset vis afbryder eller irriterer brugere, ender det som oftest med, at systemets brugere blæser på sikkerheden – enten undlader de at følge sikkerhedsprocedurer, eller også foretager de sig uhensigtsmæssige ting for at spare tid.

EKSEMPEL

### SKIFTENDE KODEORD

Mange systemer vil sikre sig mod svage bruger-kodeord og kræver derfor, at alle brugere skifter kodeord en gang imellem – fx en gang om måneden. Fordi brugere ikke gider huske på nye kodeord hele tiden, vælger brugere tit at skifte mellem 2 kodeord, eller bruge et gennemskueligt system til at vælge næste kodeord med. Det gør det ikke væsentligt sværere for en angriber, og sikkerhedsforanstaltningen med de roterende kodeord ender med ikke at højne sikkerheden, men blot gøre brugerne frustrerede.

Sikkerhedsforanstaltninger kræver også tid og ressourcer af andre end brugerne:

- Sikkerhed kræver computerressourcer
- Sikkerhedsprodukter (fx firewalls eller antivirusprogrammer) skal udvælges, installeres og vedligeholdes
- Sikkerhed skal betales

IT-sikkerhed er derfor en omkostning, der skal retfærddiggøres. Det kan være svært at argumentere for denne omkostning, da resultaterne af velfungerende sikkerhed er ikke så synlige (der skete *ikke* noget). Det er selvfølgelig klart, at omkostningerne ved manglende sikkerhed kan være fatale.

## 9.7. NØGLEORD

- IT-sikkerhed
- Fortrolighed
- Integritet
- Tilgængelighed
- Data-integritet
- Autenticitet
- Uafviselighed
- Privacy
- Autorisation
- Adgangskontrol
- Trussel
- Sårbarhed
- Modmiddel
- Malware
- Firewall
- Sporbarhed
- Kryptografi
- Krypteringsalgoritme
- Klartekst
- Ciffertekst
- Nøgle
- Symmetrisk kryptering
- Hashing
- Digital signatur