Kiistämättömät allekirjoitukset

- Allekirjoitus varmistaa alkuperän ja eheyden. Aitouteen kuuluu myös **kiistämättömyys** (*non-repudiation*) .
- Kiistämättömyyden takaa yleensä **luotettu osapuoli**. (Trent, T)
- Kiistämättömyys on paitsi tekninen myös juridinen kysymys.
- Kiistämätön allekirjoitus salaisen avaimen menetelmällä:

  Trent muistaa vastaanottamiensa dokumenttien sormenjäljet ja todistaa niiden aitouden tarvittaessa.

$$\boxed{\begin{array}{lll} 1. & A \to T: & A, M,\ MAC_{K_{AT}}(A, M) \\ \ldots & & \\ 2. & T \to B: & A, M,\ MAC_{K_{BT}}(A, M) \\ 3. & T \to C: & A, M,\ MAC_{K_{CT}}(A, M) \end{array}}$$

- Kiistämätön allekirjoitus julkisen avaimen menetelmällä:

  Luotettu osapuoli Trent todistaa julkisten avainten aitouden.

$$\boxed{1. \quad A \to B: \quad M,\ S_A(M)}$$

- Käyttäjät eivät saa paljastaa avaimiaan. Mitä jos Alice julkistaa avaimensa ja kiistää allekirjoituksen?

Aikaleimapalvelu

- Liitetään dokumenttiin aikaleima, joka todistaa olemassaolon tiettynä ajanhetkenä. Allekirjoitusta tai dokumentin olemassaoloa ei voi kiistää myöhemmin.
- Menetelmä 1: luotettu osapuoli arkistoi sormenjälkiä, jotka todistavat dokumentin olleen olemassa tiettyyn aikaan.

$$\boxed{1. \quad A \to T: \quad h(M)}$$

  (M sisältää mahdolliset allekirjoitukset ja osapuolien nimet.)

  Hajautusfunktion h pitää olla (heikosti) törmäyksetön.

  Käytännössä sormenjäljen voi esim. julkaista sanomalehdessä.

- Menetelmä 2: luotettu osapuoli lisää sanoman sormenjälkeen aikaleiman ja allekirjoituksen.

$$\boxed{\begin{array}{lll} 1. & A \to T: & h(M) \\ 2. & T \to A: & T_T,\ S_T(T_T, h(M)) \\ \ldots & & \\ 3. & A \to B: & M,\ T_T,\ S_T(T_T, h(M)) \end{array}}$$

- **Ongelma**: luotettu osapuoli voi väärentää aikaleimoja.

  Ratkaisu: liitetään Trentin tekemät allekirjoitukset toisiinsa ketjuksi.

$$S_i \ = \ T_T,\ h(M_i),\ S_T(T_T, h(M_i), S_{i-1}).$$

  Jos seuraava ja edellinen allekirjoitus on arkistoitu, tiedetään sanoman kuuluvan niiden väliin. (Vrt. diario, jossa rivejä ei lisätty eikä poistettu.)

PKCS #1

- RSA Data Security Inc:in määrittelemä standardi RSA:n käyttötavoiksi.

  ISO:lla on vastaava, mutta monimutkaisempi ohjeisto (ISO/IEC 9796 yms.).

- Määrää salattujen ja allekirjoitettujen sanomien **formaatin** sekä yksityiskohtaisen **proseduurin** sanomien luomiseksi ja purkamiseksi.

  Turvallisuus, tehokkuus ja toteutuksen helppous harkittu tarkkaan. Itse ei kannata yrittää suunnitella parempaa.

- Muita PKCS-standardeja:
  - Diffie-Hellman
  - salasanaan perustuva salaus
  - sertifikaatit.

- **PKCS-allekirjoitus**:

$$00\ 01\ ff\ ff\ \ldots\ ff\ ff\ 00\ \mathrm{DATA}.$$

  - 00 alussa varmistaa, että luku on aina pienempi kuin moduli, vaikka niitä esitetään yhtä monella bitillä.
  - Vakioarvoisia täytetavuja ff on vähintään 8 kappaletta, jotta redundanssi riittää erottamaan aidon allekirjoituksen satunnaisesta datasta.
  - 00 ennen dataa mahdollistaa yksikäsitteisen avaamisen.
  - DATA on ASN.1 BER-koodattu hajautusarvo sanomasta.

- **PKCS-salaus**:

$$00\ 02\ rr\ rr\ \ldots\ rr\ rr\ 00\ \mathrm{DATA}.$$

  - 00 alussa varmistaa, että luku on aina pienempi kuin moduli, vaikka niitä esitetään yhtä monella bitillä.
  - $02 \neq 01$, joten salattu sanoma ja allekirjoitus erottuvat.
  - Salattava luku on aina riittävän suuri, jotta jo pienet potenssit redusoituvat modulo n.
  - Satunnaisia, nollasta eroavia täytetavuja (rr) on vähintään 8 kappaletta, jotta *forward search* -hyökkäys ei ole mahdollinen.
  - 00 ennen dataa mahdollistaa yksikäsitteisen avaamisen.
  - DATA on salattava tieto.

Salauksen viranomaisvalvonta

- Viranomaiset ja tiedusteluelimet haluavat seurata tietoliikennettä.

  Menetelmiä:
  - salauksen kieltäminen (Venäjä, Kiina)
  - vahvan salauksen kieltäminen (USA:n vienti, MS, GSM)
  - avainten luovuttaminen viranomaisille (Ranska)
  - algoritmeihin suunnitellut **takaportit** (Lotus Notes)

  Tulevaisuuden salaustuotteisiin tulee takaportit (*key escrow, key recovery, law enforcement access feature*) viranomaisia ja valmistusmaan tiedustelua varten.

  Pyrkimyksenä on estää luotettavan salauksen tulo massatuotteisiin ja siten laaja käyttö liike-elämässä. Sotilasorganisaatiot, pankit tms. käyttävät joka tapauksessa erikoisohjelmistoja.

- Salauksen laaja käyttö on estetty käyttörajoitusten lisäksi
  - vientiä rajoittavalla lainsäädännöllä (USA, COCOM, EU)
  - estämällä vahvojen menetelmien standardointi (USA, UK, Ranska)
  - painostamalla suuria ohjelmisto- ja laitevalmistajia taloudellisesti.

Key escrow, key recovery

- *Key escrow*: (Clipper, Ranska)

  Pääavaimet ovat viranomaisten hallussa tai niitä säilytetään organisaation sisällä viranomaisia varten.

  Ongelmia:
  - pääavainten säilytys on kallista ja vaarallista — uusi viranomaistehtävä tai teollisuuden ala
  - viranomaiset haluavat silti saada avaimet helposti ja huomaamatta
  - korkeat kustannukset rajoittavat salauksen käyttöä
  - viranomaiskontrolli monopolisoi salausohjelmistojen ja raudan valmistuksen harvoihin käsiin (kuka hyötyy?)
  - eri maiden viranomaisilta vaaditaan yhteistoimintaa.

- *Key escrow* **voi olla tarpeen organisaation sisällä** sen omien turvallisuusjärjestelyjen takia!

- USA:n tavoitteena kehittää *key-recovery*-tekniikka:

$$A \longrightarrow B:\ E_K(M),\ E_B(K),\ E_{NSA}(K),\ \text{control field}$$

Avainten varmuuskopiointi

- *Key recovery* on **usein tarpeen organisaation sisällä**.

  Salauksen käyttö ei saa heikentää järjestelmän luotettavuutta. **Talletetun** tiedon on säilyttävä levyvirheistä ja salasanan unohtumisista huolimatta!

  Avainten varmuuskopiot voidaan säilyttää kassakaapissa, mutta se vaatii paljon käsityötä.

  Automatisoitu avainten varmuuskopiointi:

  1. Tieto salataan salaisella avaimella K.
  2. Avain K jaetaan esim. 5 osaan, joista vähintään 4 tarvitaan salauksen avaamiseen (**kynnysrakenne**).
  3. Osat salataan 5 luotetun henkilön julkisilla avaimilla.
  4. Salatut osat joko lähetetään useaan julkiseen tietokantaan tai talletetaan useaan kohtaan samalle levylle kuin salattu tieto.

  Tiedonsiirrossa key recovery on käyttäjälle täysin tarpeetonta. **Siirrettävän** tiedon avaimet halutaankin unohtaa.

Tunnistaminen

(*identification, entity authentication, identity verification*)

- Osoitetaan käyttäjän tai laitteen läsnäolo. Ei luoda salattua yhteyttä.
- Perustuu tunnistettavan
  - tietämään salaisuuteen (salasana, salausavain)
  - hallussa olevaan laitteeseen (ajokortti, avain, älykortti)
  - fyysisiin ominaisuuksiin (sormenjälki, ääni).
- Käytetään
  - käyttäjän tunnistamiseen tietokoneissa
  - älykortin tunnistamiseen lukulaitteessa
  - protokollan osana turvallista yhteyttä tietokoneiden välille muodostettaessa.
- Vahvuudeltaan erilaisia keinoja:
  1. salasana
  2. haaste–vaste
  3. nollatietotodistus.

Salasanat

- **Säilyttäminen salattuna**

  Ongelma: salasanat joudutaan usein tallentamaan tiedostojärjestelmään, joka on käyttäjien luettavissa.

  Ratkaisu: salataan salasanat järjestelmän pääavaimella K ja tallennetaan tiedostoon

  $$A,\ E_K(PW_A).$$

- **"Salaus" yksisuuntaisella funktiolla**

  Ongelma: missä säilytetään ja käytetään pääavainta?

  Ratkaisu: talletetaan salasanasta vain sormenjälki

  $$A,\ h(PW_A).$$

  Pääavainta ei tarvita.

  Huom. Funktiolta h ei vaadita konfliktittomuutta.

- **Brute-force**-haku

  Ongelma: 8-merkkisiä salasanoja on vähän ja niistäkin käytetään vain pientä osaa. Salasanojen entropia on pieni. Ne voidaan löytää kokeilemalla.

  Ratkaisu 1: lisätään salasanojen pituutta tai vaaditaan laajemman merkistön käyttöä.

  Ratkaisu 2: käytetään hitaasti laskettavaa (kompleksinen eikä optimoituja rautatoteutuksia saatavilla) yksisuuntaista funktiota, jotta *brute-force*-hyökkäys hidastuu.

- **Suolaus**

  Ongelma: hyökkääjä kerää paljon salasanojen sormenjälkiä ja laskee sormenjäljen suuresta joukosta arvaamiaan salasanoja. Yhteisiä alkioita löytyy helposti (vrt. syntymäpäivähyökkäys).

  Ratkaisu: lasketaan sormenjäljet salasanasta ja jokaiselle käyttäjälle erilaisesta suolasta R_A (satunnaisluku, uid, käyttäjän nimi tms.) ja tallennetaan tiedostoon

  $$A,\ R_A,\ h(PW_A,\ R_A).$$

  Jokainen salasana kräkättävä erikseen.

- **Esim: Unixin crypt(3)**
  - Salasanoista talletetaan sormenjäljet tiedostoon `/etc/passwd`.
  - Salasana on $8 \cdot 7 = 56$ bittiä.
  - Yksisuuntainen funktio on muunneltu DES. Salasanaa käytetään avaimena.

    $$h(K_A) = E_{PW_A}[vakio].$$

  - *Brute-force*-hyökkäyksen hidastamiseksi DES-salausta iteroidaan 25 kertaa. Tavalliset DES-piirit eivät kelpaa.
  - Suola on 12-bittinen pseudosatunnaisluku, jolla vaikutetaan DES:in laajennuspermutaatioon.

Kertakäyttösalasanat

- Vakiosalasana voi paljastua vahingossa tai salakuuntelijalle. Käytetään joka kerta uutta salasanaa.

  Vaihtuvat salasanat voidaan laskea salaisesta avaimesta ja sarjanumerosta tai kellonajasta yksisuuntaisella funtiolla.

  Salasanalista voidaan myös printata paperille

- Esim: **Lamportin kertakäyttösalasanat**

  Generoidaan satunnaisluku R.

  Talletetaan tiedostoon

$$h^n(R) \;=\; \underbrace{h(h(\ldots(h}_{n \text{ kpl}}(R))\ldots).$$

  (h on yksisuuntainen funktio, $n = 1000$.)

  Ensimmäinen salasana on $h^{n-1}(R)$, seuraava $h^{n-2}(R)$ jne.

  n:s salasana on R, minkä jälkeen generoidaan uusi R.

Haaste–vaste

- Tunnistaja lähettää haasteen, johon vain oikean salaisuuden (salaisen tai yksityisen avaimen) tunteva pystyy vastaamaan.

  Vaste lasketaan funktiolla, joka ei paljasta salaisuutta.

- Yksisuuntainen tunnistaminen:

1.	$B \longrightarrow A:$	R_B
2.	$A \longrightarrow B:$	$MAC_K(R_B, B)$

  Kaksisuuntainen tunnistaminen:

1.	$B \longrightarrow A:$	R_B
2.	$A \longrightarrow B:$	$R_A,\ MAC_K(R_A, R_B, B)$
3.	$B \longrightarrow A:$	$MAC_K(R_B, R_A, A)$

  Turvallista kaksisuuntaista protokollaa ei yleensä saada toistamalla yksisuuntainen protokolla kaksi kertaa!

- Haaste voi olla myös kellonaika:

1.	$A \longrightarrow B:$	$T_A,\ MAC_K(T_A, B)$

  Kelloja käyttämällä säästetään yksi sanoma.

  Kellojen pitää olla synkrosoituja. Synkronointiprotokolla ei itse saa luottaa kelloihin!

Julkisen avaimen haaste–vaste

- Yksityisen avaimen haltija osaa
  1. avata salauksen
  2. allekirjoittaa haasteen.
- Yksisuuntainen tunnistaminen:

1.	$B \longrightarrow A$	$h(R_B), E_A(R_B, B)$
2.	$A \longrightarrow B$	R_B

  h on yksisuuntainen funktio. $h(R)$ on **todistaja** (*witness*) sille, että B tuntee R:n ennestään.

  Protokollalla on **nollatieto**-ominaisuus: B ei saa mitään tietoa salaisuudesta, mutta uskoo, että A tuntee sen.

Nollatietotodistus

- Peggy (*prover*) todistaa Victorille (*verifier*) tuntevansa jonkin salaisuuden ilman, että paljastaa mitään tietoa salaisuudesta.

  Victor vakuuttuu, että Peggy tuntee salaisuuden, mutta ei voi vakuuttaa muita.

- Toteutustapoja:
  - Peggy jakaa vaikean ongelma kahteen osaan, joiden ratkaisut yhdessä ovat koko ongelman ratkaisu, mutta osat yksinään ovat yhtä vaikeita kuin koko ongelma. Victor vallitsee toisen osista ja Peggy ratkaisee sen. Alkuperäisen ongelman ratkaisu ei paljastu.
  - Vaikeassa ongelmassa on kaksi parametriä. Peggy valitsee ensimmäisen ja Victor sen jälkeen toisen. Peggy ratkaisee ongelman noilla parametreillä. Ratkaisusta ei ole mitään apua muilla parametreillä.
- Käytetään tavallisimmin älykortin tunnistamiseen.

  Liikennettä kortin ja lukijan välillä ei salata, mutta kortin aitous varmistetaan.

- Idea: Peggy haluaa todistaa tuntevansa ongelman P ratkaisun. Peggy valitsee uuden ongelman Q P:n ja P:n ratkaisun "puolivälistä". Victorin pyynnöstä Peggy joko näyttää tien P:stä Q:hun tai Q:sta ratkaisuun.

1. Peggy muuntaa ongelman P toiseksi vaikeaksi ongelmaksi Q (käyttäen hyväksi salaista tietoaan).
   - P:n palauttaminen Q:ksi on vaikeaa ilman salaisuutta.
   - Q:n ratkaiseminen on vaikeaa ilman salaisuutta.
2. Peggy kertoo Victorille ongelman Q.
3. Victor pyytää Peggyä joko ratkaisemaan ongelman Q tai osoittamaan, miten P palautuu Q:ksi.
4. Peggy näyttää Victorin valitseman osan ratkaisusta.
5. Askeleita 1–4 toistetaan monta kertaa.

- Victor ei saa saada osien ratkaisuista mitään tietoa ongelman P ratkaisusta. Jäljellä olevan ongelma on aina yhtä vaikea kuin P.
- Victor uskoo Peggyn tuntevan ratkaisun P:hen, muttei voi todistaa sitä muille.

Graafi-isomorfismi

- Kahden graafin isomorfisuus on vaikea ongelma.
- Peggyn salaisuus: hän tuntee graafien G_1 ja G_2 isomorfismin.

  Peggy haluaa todistaa Victorille tuntevansa isomorfismin ilman, että kertoo sitä.

1. Peggy luo uuden graafin G', joka on isomorfinen G_1:n ja G_2:n kanssa. Se käy helposti nimeämällä solmut uudelleen ja muuttamalla järjestystä graafikuvauksessa.
   - G_1:n ja G':n isomorfismi on vaikea löytää.
   - G':n ja G_2:n isomorfismi on vaikea löytää.
2. Peggy lähettää Victorille graafin G'.
3. Victor valitsee toisen isomorfismeista $G_1 \leftrightarrow G'$ ja $G' \leftrightarrow G_2$.
4. Peggy näyttää Victorille tämän valitseman isomorfismin.
5. Askeleita 1–4 toistetaan monta kertaa.

- Victor ei saa mitään tietoa isomorfismista $G_1 \leftrightarrow G_2$. Jäljellä oleva isomorfismi $G_1 \leftrightarrow G'$ tai $G' \leftrightarrow G_2$ on hänelle yhtä vaikea löydettävä.
- Victor uskoo Peggyn tuntevan G_1:n ja G_2:n isomorfismin, muttei voi todistaa sitä muille.

Guillou-Quisquater

- Nollatietotodistukseen perustuva tunnistusprotokolla.
- Suunniteltu älykortteja varten: vähän kommunikaatiota.
- Luotettu taho Trent luo avaimet:

1. Trent generoi luvut v ja $n = p\,q$, joka on kahden salaisen alkuluvun tulo. v ja n ovat julkisia ja yhteisiä kaikille käyttäjille.
2. Peggyn henkilötiedoista lasketaan sormenjälki, Peggyn tunnus J.
3. Trent ratkaisee Peggyn **salaisen avaimen** B yhtälöstä

   $$J\,B^v \equiv 1 \pmod{n}.$$

   Tämä on mahdollista, koska Trent tuntee n:n tekijät.

- Peggy todistaa Victorille henkilöllisyytensä:

1. Peggy valitsee satunnaisluvun r ja lähettää Victorille

   $$T = r^v \bmod n.$$

2. Victor valitsee satunnaisluvun d ja lähettää sen Peggylle.
3. Peggy lähettää Victorille

   $$D = r\,B^d \bmod n.$$

4. Autentikointi onnistui, jos

   $$D^v\,J^d \equiv T \pmod{n}.$$

$$\begin{aligned} D^v\,J^d &= (r\,B^d)^v\,J^d = r^v\,B^{dv}\,J^d \\ &= r^v\,(J\,B^v)^d = r^v \cdot 1^d = r^v = T. \end{aligned}$$