Krüptograafia? ... see on imelihtne!
β *versioon*

Sven Laur

30. september 2004. a.

1 Olematud ühesuunalised funktsioonid

<u>Def.</u> (OWF) Ühesuunaliseks funktsiooniks $f : \mathcal{D} \to \mathcal{R}$ nimetatakse efektiivselt arvutatavat funktsiooni, mille korral on raske leida elementi $x \in f^{-1}(y)$, kui y on võetud jaotusesest $f(z) : z \xleftarrow{u} \mathcal{D}$.

- Konkreetse funktsiooni korral pole polünomiaalsel keerukusel mõtet.
- Pööramise raskuse saab sõnastada täpselt, andes ette maksimaalse tööaja t ja maksimaalse õnnestumistõenäosuse ϵ. Asümptootiline turvagarantiil iseenesest ei taga veel turvalisust.

On kerge taibata, et kui $|\mathcal{D}|$ või $|\mathcal{R}|$ on piisavalt väiksed, siis ei saa <u>põhimõtteliselt</u> leiduda ühtegi ühesuunalist funktsiooni $f : \mathcal{D} \to \mathcal{R}$. Järgnevad ülesanded illustreerivad seda fakti.

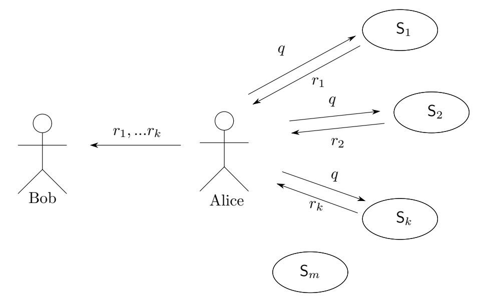
Ülesanne 1 [10] *Näidata, et järgmine ühisosa leidmise protokoll pole turvaline. Leida efektiivne rünne. Funktsioon $f : \{0,1\}^{320} \to \{0,1\}^{160}$ ühesuunaline funktsioon.*

PRIVAATSED SISENDID: Hulgad A ja B universumist $\mathcal{X}$.
ÜHINE VÄLJUND: Ühisosa $A \cap B$.

- Alice saadab Bobile väärtused $f(a_1), \ldots, f(a_k)$.
- Bob saadab need elemendid b_j Alice'ile, mille korral $f(a_i) = f(b_j)$.

Iga e-teenuse turvalisust (verifitseeritavust ja töökindlust) saab tõsta käivitades sellest m sõltumatut koopiat. Eriti oluline on formaalne verifitseeritavus teenustes, kus kasutakse (tingimusteta) usaldatavat osapoolt nagu autentimine, ajatembeldus, delegeeritud digitaalallkirjad vms. Alice esitab sellisel juhul k päringut serveritele $\mathsf{S}_1, \ldots, \mathsf{S}_m$. Kombineeritud vastus loetakse korrektseks vaid siis, kui kõik vastused on korrektsed.

Vastasel korral võib Alice esitada kaebuse konkreetse serveri vastu. Et parandada teenuse käideldavust on mõtekas nõuda $k < m$ serveri kaasamist, muidu pole süsteem kaitstud DoS rünnete või tehniliste tõrgete eest.



Ülesanne 2 [10] *Ainus reaalne oht laiendatud protokollis on kokkumäng Alice'i ja korrumpeerunud serverite vahel (teenuse tõkestamine kahjustab vaid serveri usaldusväärsust). Mida suurem on k, seda rohkem servereid peab Alice'ga kokku mängima. Kuid samas kulub rohkem ressursse (näiteks digitaalallkirjade suurus kasvab lineaarselt). Olgu t ausate serverite arv.*

(a) *Kui suur peab olema k, kui Alice valib ise serverid?*

(b) *Kui suur peab olema k, kui serverid valitakse juhuslikult?*

(c) *Kui suur peab olema k, kui serverid valitakse kasutades ühesuunalist funktsiooni?*

(d) *Kas ühesuunalise funktsiooni kasutamine saab praktilistes tingimustes ($m \leq 10$) tõsta turvalisust?*

Üks esimesi ning ilmselt levinumaid krüptograafia rakendusi on ühesuunalistel funktsioonidel põhinevad paroolisüsteemid. Kuid arvutusvõimsuse oluline kasv on vähendanud tunduvalt selle meetodi turvalisust. Vaatleme järgnevas on paroole, mis on genereeritud vastavalt heale tavale, see tähendab:

- koosnevad 8 ülimalt sümbolist;
- iga sümbol on kas täht (A-Z, a-z), number (0-9) või kirjavahemärk (.,!?;:).

Pikemate ja keerulisemate paroolide kasutamisel on mitmeid takistusi nagu inimese piiratud mälumaht ja erinevad klaviatuuri konfiguratsioonid.

Ülesanne 3 [20] *Linux'i paroolide räsi hoitakse failis* `/etc/shadow` *koos 12 bitise* soolaga*, mis lisatakse paroolile enne ühesuunalise funktsiooni rakendamist.*

(a) *Hinnata soolamiseta paroolihaldussüsteemi on turvalisust, kui kasutatakse räsifunktsioone MD5 või SHA-1.*

(b) *Milline räsifunktsiooni omadus parandaks oluliselt paroolisüsteemi turvalisust?*

(c) *Kas paroolide soolamine parandab olukorda?*

(d) *Kas on vahet kui* sool *genereeritakse juhuslikult või fikseeritakse vastavalt kasutajatunnusele.*

Käsitleme veelkord ühisosa $A \cap B$ turvalist leidmist. On kerge taibata, et mida suurem on A, seda rohkem infot B kohta lekib. Piirjuhul $A = \mathcal{X}$ saab Alice teada $A \cap B = B$. Seega peaks Alice ühisosa protokollis suutma tõestada, et $|A| \leq k$.

Ülesanne 4 [50] *Freedman-Nissim-Pinkase ühisosa leidmise protokoll [FNP04] kasutab arvutuste optimiseerimiseks jaga-ja-valitse meetodit. Ühesuunaliste räsifunktsioonide abil jagatakse hulk A kuni viie elemendilisteks hulkadeks $\mathcal{A}_1, \ldots, \mathcal{A}_m$. Protokoll garanteerib, et hulgad $\mathcal{A}_1, \ldots, \mathcal{A}_m$ ei leki. Kuid tõestuses $|A| \leq k$, avalikustab Alice suurused $d_i = |\mathcal{A}_i|$. Veidi lihtsustatud protokollis kasutatakse avalikku ühesuunalist funktsiooni $f : \mathcal{X} \to \{1, \ldots, k\}$ hulga A tükeldamiseks, st. $\mathcal{A}_i = \{a \in A : f(a) = i\}$.*

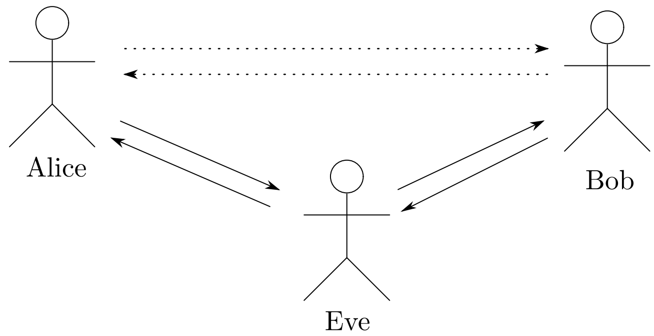
(a) *Näidata, et FNP protokoll on ebaturvaline.*

(b) *Leida efektiivne rünne, mis paljastab A, kui protokolli rakendatakse mitu korda samale sisendile A.*

(c) *Üldistada eelmisi tulemusi FPN protokollis tegelikult kasutatavale räsiskeemile.*

2 Men-in-the-middle-attack ehk vahendusrünne

Krüptograafilised vahendid tagavad turvalisuse vaid teatud eeldustel. Neist üks olulisemaid eeldusi on jagatud *saladuse* olemasolu. Näiliselt võimaldab avaliku võtme krüptograafia sellest loobuda: avaliku signeerimisvõtme või krüpteerimisvõtme teadmisel saab luua turvalise terviklikust tagava sidekanali kahe osapoole vahel ilma jagatava *saladuseta*. Tegemist on illusiooniga — "saladuseks" on siin avalik võti. Avaliku võtme krüptograafia maandab vaid riske: "avaliku saladuse" ilmsiks tulek ei aita ründajat, küll aga peab olema tagatud avaliku võtme autentsus. Vastasel korral saab süsteemi rünnata kasutades vahendusrünnet, kus Eve teeskleb Alice'ile, et on Bob, ja Bobile, et on Alice.



Järgnevas vaatlemegi kui turvalised on hetkel kasutatavad tehnoloogiad. Sealjuures teeme *ketserliku* eelduse, et me ei oma arvuti üle täielikku kontrolli. See on üsna loomulik, sest mitmed levinud operatsioonisüsteemid on suletud koodiga. Ka avatud koodiga tarkvara ei anna kasutajale täielikku kontrolli — ükski inimene ei suuda kogu vastavat koodi läbi analüüsida (seada on üle 10 miljoni koodirea). Teiseks ründavad meie arvuteid viirused ja pahalased. Järgnevas oleme mõõdukalt paranoilised ning eeldame, et arvutikasutaja omab arvuti üle lühiajalist kontrolli. See tähendab:

- kasutaja suudab tuvastada programmide ebaharilikku käitumist;
- enamus tarkvarast töötab korrektselt;
- kuid võib arvutis olla spetsiaalseid tagauksi.

Vaatleme esmalt kahte levinut OpenSSH klienti programme ssh (Linux) ja PuTTY (Windows). Mõlema programmi puhul salvestatakse serveritele vastavad võtmed ülekirjutatavale meediale: ssh salvestab avalikud võtmed faili `/.ssh/known_hosts` ja PuTTY Windowsi registrisse. Failide ja registri terviklikust ei garanteerita.

Ülesanne 5 [15] *Kas antud eeldustel on OpenSSH turvaline vahendusrünnete vastu? Kuidas tuleks pahalasel võtmete nimistut muuta, et oleks võimalik valikuline vahendusrünne läbi võltsitud DNS päringute? Veenduda, et samad probleemid on ka veebilehitsejatel* `https` *(SSL) turvalisuse tagamisel. Millised võimalikud lahendused?*

Ülesanne 6 [5] *E-pangandus on turvaline vaid siis, kui ühendus luuakse läbi SSL kanali. Mõelda välja tähelepanematusel põhinev rünne juhuks, kui pangal on olemas ka `http`-aadressil olev koduleht. Kontrollida mitmel Eesti pangal on ikka veel aktiivses kasutuses `http`-aadress.*

Seega reaalses elus näib "avalike saladuste" turvaline säilitamine ja jagamine oluliselt keerulisem kui alul arvata võis. Paljudes visioonides on ID-kaardil põhinev autentimine lahenduseks enamikele turvaprobleemidele. Kuid kas see ikka on nii?

Et sellele küsimusele vastata, kirjeldame ID-kaardi käsutikku ja ehitust. Avalikult on dokumenteeritud kolm peamist krüptograafilist operatsiooni (pole välistatud nö. salajased lisafunktsioonid).

[AUT] *Signeerimine autentimisvõtmega*. Teatest m genereeritakse erikujul avatekst m', mis signeeritakse salajase autentimisvõtmega. Küsitakse PIN-AUT koodi.

[SIG] *Digitaalallkirja andmine*. Dokumendi räsist m genereeritakse erikujuline avatekst m', mis signeeritakse salajase signeerimisvõtmega. On võimalik dokument ka otse kaardil räsida ja signeerida. Küsitakse PIN-SIG koodi.

[DEC] *Krüptogrammi dekrüpteerimine*. Krüptogramm c dekrüpteeritakse ja kontrollitakse kas vastav avatekst on õigel erikujul m'' (teine kuju võrreldes signeerimisega). Kasutada võib nii signeerimis kui autentimisvõtit. Küsitakse vastavalt kas PIN-SIG või PIN-AUT koodi.

Ülesanne 7 [15] *Üks lihtsamaid rakendusi on ID-kaardi abil arvutisse sissel logimine: kasutajalt küsitakse PIN-AUT kood ning ID-kaardiga tehakse AUT operatsioon. Kuidas saab seda protokolli kasutada ID-kaardi kaaperdamiseks? Viimase tähendab näiteks pahalase kontrolli all oleva SSL ühenduse loomist e-pangaga ja ennast vastava ID-kaardi kasutajana autentimist. Hinnata ründe avastamise tõenäosust ning mõju koduarvuti ja serveri kontekstis.*

Ülesanne 8 [10] *SSH autentimisprotokolli standardis pole määratud, kuidas arvutatakse kiipkaardiga autentimisel vajalikku kutsungit. Mida peab kutsung rahuldama, et vahendusrünne oleks välistatud (tagauksi peale serveri vale avaliku võtme pole)?*

Ülesanne 9 [30*] *Näidata, et ID-kaarti kasutamine SSL ühenduse autentimisel (näiteks e-panganduses) pole turvaline, sest tehtud eeldustel saab ID-kaarti kergelt kaaperdada.*

(a) Mille vastu saaks üldse ID-kaarti antud oludes kaitsta?

(b) Milline peaks olema ID-kaardi käsustik, mis garanteeriks turvalisuse ka lühiajalise kontrolli mudelis?

(c) Kas probleeme saaks lahendada erilise ID-kaardi lugejaga?

(d) Kas samad modifikatsioonid võimaldavad ka turvalist signeerimist?

3 One-time-pad ehk Vernami šiffer

1917. aastal patenteeris Vernam lihtsa krüptosüsteemi One-Time Pad'i, kus n-bitine avatekst XOR-itakse juhuslikult valitud võtmega $k \in \{0,1\}^n$. On tõestatud, et Vernami šiffer on absoluutselt turvaline. Samas kaasnevad antud šifriga mitmed probleemid:

- on raske genereerida ja hoida ülipikki võtmeid;
- šiffer kaotab turvalisuse kui võtit taaskasutada.

Esimese probleemi lahendamiseks kasutatakse krüptograafiliselt tugevaid juhuarvude generaatoreid, mis võimaldavad vähendada salvestusmahtu 128 bitini. Näiteks OFB ja Counter režiimid kasutavad turvalist blokkšifrit (nagu AES või IDEA) võtmejada k genereerimiseks ning krüpteerimine vastab otseselt Vernami šifrile.

Võtmete taaskasutamine on ootamatult levinud probleem, mis jääb turvaanalüüsides tihti märkamatuks. Nii näiteks õnnestus Inglismaa ja USA salateenistustel VENONA projekti (1943–1980) raames taastada osa Vernami šifriga krüpteeritud NSV Liidu diplomaatilisest korrespondentsist just tänu võtme taaskasutamisele. Teiseks tuntumaks näiteks on paljud failisüsteemide krüpteerimisvahendid [Saa04a, Saa04b].

Ülesanne 10 [10] *Näidata, et järgmine interaktiivne krüptosüsteem pole turvaline. Miks?*

EELDUS: Ründaja on passivne.
PRIVAATNE SISEND: Alice'i avatekst $m \in \{0,1\}^n$.

- Alice valib juhuslikult $r_1 \in \{0,1\}^n$ ja saadab Bobile $c_1 = m \oplus r_1$.
- Bob valib juhuslikult $r_2 \in \{0,1\}^n$ ja saadab Alicile $c_2 = c_1 \oplus r_2$
- Alice saadab $c_3 = m \oplus r_2 = c_2 \oplus r_1$ Bobile.
- Bob leiab $m = c_3 \oplus r_2$.

Vaatleme nüüd võtme taaskasutamisest põhjustatud probleeme failisüsteemide krüpteerimisel. Sealjuures on kasulik teada, et failisüsteemid koosnevad andmeblokkidest suurusega 1024, 2048 või 4096 bitti. Kuna enamus hetkel disainitud blokkšifritest E_k on 128 bitised, siis on loomulik jagada andmeblokk 128 bitisteks alamblokkideks, st. $d_i = (m_{i,1}\|\ldots\|m_{i,\ell})$. Arusaadavalt peab iga mõistlik krüpteeritud failisüsteem pakkuma kohest ligipääsu suvalisele andmeblokile.

Ülesanne 11 [20] *Vaatleme kolme traditsioonilist failisüsteemi krüpterimisviisi:*

[ECB] *andmed jagatakse 128 bitisteks blokkideks ja iga blokk krüpteritakse eraldi, st.* $c_{i,j} = E_k(m_{i,j})$*;*

[CBC] *iga blokk krüpteeritakse kasutades CBC režiimi, võttes algväärtuseks* IV *bloki numbri, st.* $c_{i,1} = E_k(i \oplus m_i)$ *ja* $c_{i,j} = E_k(c_{i,j-1} \oplus m_i)$*;*

[CNT] *blokkšifrit kasutatakse võtme genereerimiseks, st.* $c_{i,j} = E_k(\ell \cdot i + j) \oplus m_{i,j}$.

Veenduda, et kõik kolm nimetatud failisüsteemide krüpteerimisviisi on ebaturvalised. Leida vastavad ründed. Miks on ECB ja CNT režiimid eriti ebaturvalised?

Ülesanne 12 [30] *Disainida täiendus CBC failisüsteemile, mis muudab selle turvaliseks vastavate rünnete suhtes ning ei suurenda oluliselt failisüsteemi pöörduste aega. Formuleerida täpsed turvatingimused ja tõestada süsteemi turvalisus. Miks peab* `write` *pöörduste maht olema tõkestatud?*

Ülesanne 13 [30] *Eeldades, et mälu* swapimine *kui selline on hetkel aegunud tehnoloogia ja kõvaketta ruum pole probleem, disainida turvaline krüpteeritud failisüsteem, mis kasutab blokkšifrit CNT režiimis. Miks on saavutatav turvalisus on illusoorne, kui kasutada peavõtme peitmiseks paroolisüsteeme? Milliste vahenditega tuleks tagada tegelik turvalisus?*

4 Doominoefekt ja turvaline ühisarvutus

Krüptograafia üheks suuremaks eesmärgiks on doominoefekti vältimine — turvalisus peab säilima, kui leke on piisavalt väike. (Loomulikult pole võimalik turvalisust tagada kui salajased võtmed lekivad). Seetõttu on turvadefinitsioonid pahatihti väga kohmakad või "liiga" ranged. Ometi viivad katsed definitsioone lõdvendada enamasti doominoefektini. Iseseisvalt võib primitiiv tagada piisava turvalisuse, kuid teiste protokollide koosmõjul turvalisus hajub.

Vaatleme näitena 2002. aastal välja pakutud protokolli skalaarkorrutise arvutamiseks. Vektorid $\boldsymbol{x}$ ja $\boldsymbol{y}$ on $2n$-elemendilised ning $\boldsymbol{u_1}$ ja $\boldsymbol{v_i}$ n-elemendilised vektorid üle lõpliku korpuse $\mathbb{Z}_p$ (lihtsuse mõttes võib ette kujutada, et vektorid on üle $\mathbb{R}$).

PRIVAATSED SISENDID: Vektorid $\boldsymbol{x}$ ja $\boldsymbol{y}$
PRIVAATSED VÄLJUNDID: Arvud $s_1 + s_2 = \boldsymbol{x} \cdot \boldsymbol{y}$.

1. Alice ja Bob genereerivad juhusliku pööratava $2n \times 2n$ maatriksi M.
2. Alice leiab $(\boldsymbol{u_1} \| \boldsymbol{u_2}) = M\boldsymbol{x}$ ja saadab $\boldsymbol{u_2}$ Bobile.
3. Bob leiab $(\boldsymbol{v_1} \| \boldsymbol{v_2}) = M^{-\mathrm{T}}\boldsymbol{y}$ ja saadab $\boldsymbol{v_1}$ Alice'ile.
4. Alice arvutab $s_1 = \boldsymbol{u_1} \cdot \boldsymbol{v_1}$ ja Bob arvutab $s_2 = \boldsymbol{u_2} \cdot \boldsymbol{v_2}$

Protocol 1: Du-Zhan'i protokoll skalaarkorrutise leidmiseks

Ülesanne 14[10] *Veenduda, et vektorile $\boldsymbol{u_2}$ vastab p^n võimalikku sisendvektorit $\boldsymbol{x}$ ja vektorile $\boldsymbol{v_1}$ vastab p^n sisendvektorit $\boldsymbol{y}$. Kas protokoll on turvaline, kui $p^n > 2^{60}$? Mis juhtub kui protokolli rakendada kaks korda ühe ja sama sisendi peal?*

Def. **(MPC-I turvalisus — esimene katse, mitteformaalne)** Protokoll säilitab Alice'i privaatsuse, kui protokolli järgiv uudishimulik Bob ei suuda polünomiaalses ajas leida midagi, mida ei saaks arvutada vaid Bobi sisendist ja väljundist.

Ülesanne 15[20] *Veenduda, et Du-Atallah'i protokoll on turvaline MPC-I mõttes. Leida efektiivne rünne, kui on teada ℓ kordinaati. Kas protokoll on ikka turvaline?*

PRIVAATSED SISENDID: Vektorid $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^n$.
PRIVAATSED VÄLJUNDID: $s_1 + s_2 \equiv \boldsymbol{x} \cdot \boldsymbol{y} \mod p$.

1. Alice genereerib vektorid $\boldsymbol{v_1}, \ldots, \boldsymbol{v_m} \in \mathbb{Z}_p^n$, nii et $\boldsymbol{x} = \boldsymbol{v_1} + \cdots + \boldsymbol{v_\ell}$,
   ning saadab $\boldsymbol{v_1}, \ldots, \boldsymbol{v_m}$ juhuslikus järjekorras π Bobile.
   Bob arvutab $t_i = \boldsymbol{v_i} \cdot \boldsymbol{y}$.

3. For $i = 1, \ldots, \ell$
   Bob valib juhusliku $r_i \in \mathbb{Z}_p$ ja leiab tabeli $T_i = (t_1 - r_i, \ldots, t_m - r_i)$
   Alice leiab $\tau_i = t_{\pi(i)} - r_i$, tehes OT päringu tabelisse T_i.

4. Alice väljastab $s_i = \tau_1 + \ldots + \tau_\ell$.
   Bob väljastab $s_2 = r_1 + \ldots + r_\ell$.

Protocol 2: Du-Attalah skalaarkorrutise protokoll

Def. **(MPC-II turvalisus — teine katse, mitteformaalne)** Protokoll säilitab Alice'i privaatsuse, kui protokolli jälgiv uudishimulik Bob ei suuda polünomiaalses ajas teha vahet Alice ja simulaatori vahel. Simulaator teab vaid Bobi sisendit ning oodatavat väljundit. Sealjuures saab Bob valida endale sobiva Alice'i sisendi ja väljundi.

Ülesanne 16 *Näidata, et Du-Atallah'i protokoll pole MPC-II mõttes turvaline.*

Viited

[Dai04] Wei Dai. Crypto++ 5.2.1 benchmarks. `http://www.eskimo.com/~weidai/benchmarks.html`, 2004.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.

[Saa04a] Markku-Juhani O. Saarinen. Encrypted watermarks and linux laptop security. 2004. Leitav `www.tcs.hut.fi/~mjos/doc/wisa2004.pdf`.

[Saa04b] Markku-Juhani O. Saarinen. Vestlused sõbralikus õhkonnas, 2004.