IVXV võtmerakendus

Spetsifikatsioon

Versioon 1.0

20. september 2017

11 lk

Dok IVXV-SVR-1.0

Sisukord

PEATÜKK 1

Võtmerakendus

1.1 Sissejuhatus

Võtmerakendus on Korraldaja põhitööriist, millega genereeritakse iga hääletamise jaoks häälte salastamise ja häälte avamise võti. Võtmerakenduse abil toimub ka häälte lugemine ja tulemuse väljastamine.

Dokumendis spetsifitseeritakse võtmerakenduse tehnilised detailid.

1.2 Eritüübiliste võtmeosakute protokollide liidestamine

Võtmeosakute genereerimise protokolli liides

Klass *ee.ivxv.key.protocol.GenerationProtocol* defineerib liidese, mida ElGamali või RSA võtmeosakute genereerimise protokoll peab täitma. Liides on järgnev:

```
public interface GenerationProtocol {
    byte[] generateKey() throws ProtocolException;
}
```

Eritüübiliste võtmeosakute genereerimiseks peab implementeerima *generateKey()* meetodi, mis tagastab kodeeritud avaliku võtme X.509 sertifikaadina DER formaadis. Protokolli klass peab olema paki *ee.ivxv.key.protocol.generation* alampakis.

Protokolli parameetrid tuleb määrata protokolli klassiinstantsi initsialiseerimise ajal.

Dekrüpteerimise protokolli liides

Dekrüpteerimise ja võtmeosakute genereerimise protokoll ei pea olema üksüheses seoses, st. võtmeosakute genereerimise protokollile võib vastata mitu dekrüpteerimise protokolli. Seega on dekrüpteerimise protokolli liides defineeritud sõltumatult võtmeosakute genereerimise liidesest. Protokoll peab implementeerima *ee.ivxv.key.protocol.DecryptionProtocol* liidese:

```
public interface DecryptionProtocol {
    ElGamalDecryptionProof decryptMessage(byte[] msg) throws␣
↪ProtocolException;
}
```

Meetod *decryptMessage()* võtab sisendina krüptogrammi DER formaadis ning väljastab *ElGamalDecryptionProof* instantsi. Juhul kui protokoll ei toeta lugemistõendi väljastamist, siis on vastavad väljad väärtustatud tühiväärtusega (*null*).

Analoogselt võtmeosakute genereerimise protokollile tuleb protokolli parameetrid määrata klassiinstantsi initsialiseerimise ajal.

Allkirjastamise protokolli liides

Lisaks dekrüpteerimise protokollile saab implementeerida ka allkirjastamise protokolli. Sarnaselt dekrüpteerimisprotokollile võib ühele võtmegenereerimise meetodile vastata mitu allkirjastamise protokolli. Protokoll peab implementeerima *ee.ivxv.key.protocol.SigningProtocol* liidese:

```
public interface SigningProtocol {
    byte[] sign(byte[] msg) throws ProtocolException;
}
```

Meetod *sign()* võtab sisendina sõnumi, mida soovitakse allkirjastada, ja väljastab RSA-PSS allkirja järgnevate parameetritega:

- sõnumi räsifunktsioon: SHA2-256
- maski genereerimise funktsioon: MGF1, maski räsifunktsioon SHA2-256 ja maski pikkus 32 baiti
- soola pikkus: 32 baiti
- sababait: *0xbc*

Analoogselt võtmeosakute genereerimise protokollile tuleb protokolli parameetrid määrata klassiinstantsi initsialiseerimise ajal.

Toetatud protokollid

Hetkel on teostatud järgnevad võtmeosakute genereerimise protokollid:

- *ee.ivxv.key.protocol.generation.desmedt.DesmedtGeneration*: Võtmeosakud on sellised, et oleks võimalik kasutada *[DF89]* (page 11) hajutatud dekrüpteerimisprotokolli. Võtmeosakud salvestatakse otse PKCS15 liidest toetavale pääsmikule. Klassiinstantsi initsialiseerimise ajal saab anda järgnevaid argumente:
  - *PKCS15Card[] cards*: järjend objektides mis implementeerivad PKCS15Card liidest (nt. kiipkaardid või tarkvaralised pääsmikud).
  - *ElGamalParameters params*: ElGamali krüptosüsteemi parameetrid.
  - *ThresholdParameters tparams*: läviskeemi parameetrid.
  - *Rnd rnd*: juhuslikkuse sisend võtmeosakute genereerimisel
  - *byte[] cardShareAID*: võtmeosaku ligipääsuidentifikaator PKCS15 pääsmikul. Defineerib, millise ligipääsutunnusega pääseb võtmeosakule ligi.
  - *byte[] cardShareName*: võtmeosaku identifikaator PKCS15 pääsmikul.
- *ee.ivxv.key.protocol.generation.shoup.ShoupGeneration*: Võtmeosakud on sellised, et oleks võimalik kasutada *[Shoup00]* (page 11) põhinevat hajutatud allkirjastamisprotokolli. Võtmeosakud salvestatakse otse PKCS15 liidest toetavale pääsmikule. Klassiinstantsi initsialiseerimise ajal saab anda järgnevaid argumente:
  - *PKCS15Card[] cards*: järjend objektidest, mis implementeerivad PKCS15Card liidest (nt. kiipkaardid või tarkvaralised pääsmikud).
  - *int modLen*: RSA võtme pikkus bittides
  - *ThresholdParameters tparams*: läviskeemi parameetrid.
  - *Rnd rnd*: juhuslikkuse sisend võtmeosakute genereerimisel
  - *byte[] cardShareAID*: võtmeosaku ligipääsuidentifikaator PKCS15 pääsmikul. Defineerib, millise ligipääsutunnusega pääseb võtmeosakule ligi.
  - *byte[] cardShareName*: võtmeosaku identifikaator PKCS15 pääsmikul.

On teostatud järgnevad dekrüpteerimise protokollid:

- *ee.ivxv.key.protocol.decryption.recover.RecoverDecryption*: Loetakse PKCS15 liidest toetavatelt pääsmikelt võtmeosakud, rekonstrueeritakse nende abil operatiivmälus salajane võti ning teostatakse dekrüpteerimine lugemistõendiga. Klassiinstantsi initsialiseerimise ajal saab anda järgnevaid argumente:
  - *PKCS15Card[] cards*: järjend objektides mis implementeerivad PKCS15Card liidest (nt. kiipkaardid või tarkvaralised pääsmikud).
  - *ThresholdParameters tparams*: läviskeemi parameetrid.
  - *byte[] cardShareAID*: võtmeosaku ligipääsuidentifikaator PKCS15 pääsmikul. Defineerib, millise ligipääsutunnusega pääseb võtmeosakule ligi.
  - *byte[] cardShareName*: võtmeosaku identifikaator PKCS15 pääsmikul.

On teostatud järgnevad allkirjastamise protokollid:

- *ee.ivxv.key.protocol.signing.shoup.ShoupSigning*: Loetakse PKCS15 liidest toetavatelt pääsmikelt võtmeosakud, konstrueeritakse mälus allkirjastamise osakud ilma võtit rekonstrueerimata ning kombineeritakse allkirjastamise osakud RSA-PSS allkirjaks. Klassiinstantsi initsialiseerimise ajal saab anda järgnevaid argumente:

- *PKCS15Card[] cards*: järjend objektides mis implementeerivad PKCS15Card liidest (nt. kiipkaardid või tarkvaralised pääsmikud).
- *ThresholdParameters tparams*: läviskeemi parameetrid.
- *Rnd rnd*: juhuslikkuse sisend RSA-PSS allkirja soola genereerimisel.
- *byte[] cardShareAID*: võtmeosaku ligipääsuidentifikaator PKCS15 pääsmikul. Defineerib, millise ligipääsutunnusega pääseb võtmeosakule ligi.
- *byte[] cardShareName*: võtmeosaku identifikaator PKCS15 pääsmikul.

Protokollide võtmerakendusega liidestamine

Järgnev kirjeldus käib nii võtmeosakute genereerimise ja dekrüpteerimise protokollide kohta.

Märkus: Praegune kirjeldus on üldine. Kui konfi- ja argumentide parsimine on lõplikult välja töötatud ning protokollid võtmerakendusega liidestatud, siis tuleks järgnevat lõiku täiendada.

Liidestamaks uut protokolli võtmerakendusega, tuleb kõigepealt teostada vastava protokolli liidest täitev klass. Võtmerakendus peab töö alguses seadistuse töötlemise käigus aru saama kas käsureaargumentidest või seadistusfailist, millist protokolli soovitakse kasutada. Seejärel tuleb vastava klassi staatilise meetodi abil ülejäänud käsureaargumentide või seadistusfaili abil initsialiseerida uus protokolliklassi instants. Seejärel tuleb genereerida võti või dekrüpteerida sõnum.

Toetatud protokollide kirjeldused

Shamiri saladuse jagamise skeem

Olgu meil salajane väärtus $s = a_0$ ja soovime seda jagada n osapoole vahel selliselt, et vähemalt t osapoolt saaksid selle saladuse rekonstrueerida. Selleks valime koefitsiendid a_1 kuni a_{t-1} ning vaatame polünoomi muutuja x suhtes:

$$P(x) = a_{t-1}x^{t-1} + .. + a_1x + a_0$$

Olgu x_1 kuni x_n nullist erinevad unikaalsed väärtused (üldiselt 1 kuni n), sellisel juhul saame osakud $s_i = P(x_i)$ ning salajase väärtuse $s = P(0)$.

Geomeetriliselt vaadates on $P(x)$ polünoom ning osakud punktid sellel polünoomil. Põhikoolimatemaatikast teame, et $t - 1$ järku polünoomi joonistamiseks piisab meile t punktist (sirge jaoks kaks punkti, parabooli jaoks kolm punkti jne.). Salajane väärtus on selle polünoomi väärtus y-telje lõikepunktis.

Vaadates rekonstrueerimist arvuliselt, mitte geomeetriliselt, saame kasutades Lagrange interpoleerimise meetodit. Tähist $\prod$ kasutame me mitme liikmega korrutise tähistamiseks ja tähist $\sum$ kasutame me mitme liikmega summa tähistamiseks.

Nüüd, tähistame lisaks t osapoolt, kes osalevad salajase väärtuse rekonstrueerimisel tähisega U. Lagrange interpoleerimise valem ütleb:

$$\overline{P}(x) = \sum_{j \in U} s_j \frac{\prod\limits_{i \in U, j \neq i} x - x_i}{\prod\limits_{i \in U, j \neq i} x_j - x_i}$$

Tõepoolest: fikseerime j - paneme tähele, et kui $x = x_j$, siis murru väärtus on 1 (kuna lugejas ja nimetajas olevad kordajad taandavad üksteist) ja kui $x \neq x_j$, kuid $x = x_k$, mingi muu $k \in U$ korral, siis murd on 0 (kuna lugejas on $x_k - x_i = 0$ mingi $i \in U$ korral). Seega:

$$\overline{P}(x_j) = s_j + 0 \sum_{i \in U, i \neq j} s_i = s_j = P(x_j)$$

Kuna osapooled teavad väärtuseid $s_i = P(x_i)$ (osakud), siis kombineerides ning korrutades need läbi baaspolünoomiga

$$L(U, x, j) = \frac{\prod\limits_{i \in U, j \neq i} x - x_i}{\prod\limits_{i \in U, j \neq i} x_j - x_i}$$

ja fikseerides $x = 0$, saame jagatud saladuse.

ee.ivxv.key.protocol.generation.desmedt.DesmedtGeneration

Arvestades, et ElGamali võtmeparameetriks on rühm G koos generaatoriga g, siis salajaseks võtmeks valitakse x, mis on ülimalt $ord(g)$, st. g multiplikatiivne järk rühmas G. Vastavaks avalikuks võtmeks võetakse väärtus $y = g^x$. Salajase võtme komplektiks saab väärtus (G, g, x) ja avaliku võtme komplektiks väärtus (G, g, y). Rühm G valitakse selliselt, et tema järk on mingi algarv p selliselt, et kehtib $p = 2q + 1$, kus q on samuti algarv. Selliselt juhul kirjeldab G väärtust algarv p.

Algebrast teame, et kui G järk on $2q + 1$, siis iga selle rühma elemendi järk on kas 1, 2, q või $2q$. Me oleme huvitatud selliselt generaatorist, mille järk on q ja mis on ruutjääk, kuna see genereerib piisavalt suure alamrühma, mille kõik elemendid on ruutjäägid. Vastasel juhul võib toimuda ühe biti lekkimine krüpteeritud sõnumi kohta. Sellise generaatori leidmiseks vaatame me rühma suvalisi elemente ning kontrollime tema järku ning ruutjäägilisust kuni leiame sobiva elemendi. Sellise elemendi määrame generaatoriks.

Instants genereerib juhusliku $0 < x < q$ salajaseks võtmeks, jagab selle Shamiri ühissalastuse abil argumentidena antud osapoolte vahel. Iga salajase võtme osak kodeeritakse kui ühissalastamata salajase võtme komplekt.

Seejärel arvutatakse $y = g^x$ ning tagastatakse kodeeritud avaliku võtme komplekt.

ee.ivxv.key.protocol.generation.shoup.ShoupGeneration

RSA võtmepaar genereeritakse järgnevalt: genereeritakse kaks algarvu p ja q bitipikkusega *modLen*$/2$ ning võetakse $n = pq$. Avalik võti e võetakse selliselt et

$\gcd(e, \phi(n)) = 1$, kuid antud protokollis on e fikseeritud $e = 65537$. Seega tuleb valida p ja q nii pikalt kui see tingimus kehtib. Salajane võti d võetakse selliselt, et $de \equiv 1 \pmod{\phi(n)}$, kus ϕ on Euleri ϕ.

Arv $\phi(n)$ näitab, kui paljud arvudest $1 \leq m < n$ on sellised et $\gcd(m, n) = 1$, kus $\gcd(a, b)$ on kahe arvu a ja b suurim ühistegur. On ilmne, et kui p on algarv, siis $\phi(p) = p - 1$. Lisaks on lihtne näidata, et kui p ja q on algarvud, siis $\phi(pq) = \phi(p)\phi(q)$.

Euleri teoreem ütleb, et kui a ja n on ühistegurita, siis:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Seega, kui sõnumi m allkirjastamiseks tehakse $s \equiv m^d \pmod{n}$, siis verifitseerimiseks kontrollitakse kas $s^e \equiv m \pmod{n}$. Tõepoolest: $(m^d)^e \equiv m^{de} \equiv m^{k\phi(n)+1} \equiv m^{k\phi(n)}m \equiv 1^k m \equiv m \pmod{n}$.

Salajane võti d jagatakse Shamiri salastuse jagamisega osadeks, iga osa kodeeritakse kui jagamata salajase võtme komponent ning salvestatakse osapoolele. Avalik võtme komplekt kodeeritakse ning tagastatakse.

ee.ivxv.key.protocol.decryption.recover.RecoverDecryption

Protokoll toimib, rekonstrueerides ElGamali võtme ning dekrüpteerides sellega krüptogramme.

Täpsemalt, olgu U indeksid kaartidest, mis moodustavad argumendiks antud $cards$ muutuja. Instants loeb salajase võtme komplektid kaartidelt, kontrollib võtmekomplektide terviklust (st. rühma G ja generaatori g kirjelduse ühesust), dekodeerib igast komplektist salajase võtme x_i.

Seejärel arvutatakse salajane võti x kasutades Lagrange interpoleerimist:

$$x = P(0) = \sum_{j \in U} s_j \frac{\prod_{i \in U, j \neq i} -x_i}{\prod_{i \in U, j \neq i} x_j - x_i}$$

Krüptogrammi $c = (c_1, c_2) = (my^r, g^r)$ dekrüpteerimiseks arvutatakse:

$$d = \frac{c_1}{c_2^x}$$

Dekrüpteerimise lugemistõendi jaoks valitakse juhuslik r ning konstrueeritakse järgnevad pühendumused:

$$a = c_2^r$$
$$b = g^r$$

Seejärel arvutatakse Fiat-Shamiri pretensioon järgnevalt, kus *H* on räsifunktsioon *SHA2-256* ning *B2I* on meetod, mis teisendab baidijada täisarvuks ühtlaselt vahemikus:

```
K = H("DECRYPTION" || y || c || d || a || b)
k = B2I(K, q)
```

Nüüd arvutatakse lugemistõendi vastus:

$$s = kx + r$$

Kogu lugemistõend on komplekt (a, b, s). Tagastatakse $(d, (a, b, s))$.

ee.ivxv.key.protocol.signing.shoup.ShoupSigning

Antud protokollis ei toimu võtme rekonstrueerimist.

Olgu U indeksid kaartidest, mis moodustavad argumendiks antud $cards$ muutuja. Klassiinstants loeb salajase võtme komplektid ja kontrollib nende terviklust (st. mooduli ja avaliku võtme ühesus). Loetakse mällu võtme moodul n ja avalik võti e. Lisaks dekodeeritakse ja loetakse mällu salajased võtmed d_i. Allkirja genereerimiseks sõnumile m rakendatakse sellele EMSA-PSS kodeerimist *[RFC8017]* (page 11), kus on kasutusel varasemalt defineeritud *RSA-PSS parameetrid* (page 4), saades allkirjastamiseks sõnumi M.

Me tähistame tähisega $n!$ arvu n faktoriaali, st. $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$. Meenutame, et Lagrange interpolatsiooni baaspolünoom oli:

$$L(U, x, j) = \frac{\prod\limits_{i \in U, j \neq i} x - x_i}{\prod\limits_{i \in U, j \neq i} x_j - x_i}$$

Defineerime modifitseeritud Lagrange baaspolünoomi järgnevalt:

$$L'(U, x, j) = n! \frac{\prod\limits_{i \in U, j \neq i} x - x_i}{\prod\limits_{i \in U, j \neq i} x_j - x_i}$$

Kuna me teame, et punktid $1 \leq x_i, x_j \leq n$, siis $|x_j - x_i| < n$. Seega, korrutades Lagrange baaspolünoomi läbi $n!$, saame, et $L'(U, j)$ on alati täisarv.

Allkirja konstrueerimiseks arvutame:

$$s = \prod_{j \in U} (M^{x_j})^{L'(U,0,j)} = M^{\sum\limits_{j \in U} x_j L'(U,0,j)} = M^{n!d}$$

Kuna kasutasime modifitseeritud Lagrange interpoleerimist, siis võrreldes tavalise RSA allkirjaga on see astendatud $n!$-ga. Bezout' lemmast teame, et x ja y korral leiduvad a ja b selliselt, et $ax + by = \gcd(x, y)$. Veel enam, selliseid a ja b väärtuseid on võimalik leida laiendatud Eukleidese algoritmiga suurima ühisteguri leidmiseks.

Kasutades Eukleidese laiendatud algoritmi, leitakse a ja b, selliselt et $ae + bn! = \gcd(e, n!)$. Kuna avalik võti e on valitud algarv, siis $\gcd(e, n!) = 1$. Arvutame:

$$\sigma = M^a s^b$$

Arvestades, et $de = 1 \pmod{\phi(n)}$, on see tõesti korrektne allkiri:

$$\begin{aligned}
\sigma^e &= M^{ae} s^{be} \\
&= M^{ae} M^{n!dbe} \\
&= M^{ae} M^{n!bde} \\
&= M^{ae} M^{n!b} \\
&= M^{ae+bn!} \\
&= M^{\gcd(e,n!)} \\
&= M
\end{aligned}$$

Protokolli instants tagastab σ allkirjana.

1.3 Viited

Kirjandus

[DF89] Desmedt, Y. & Frankel, Y. Brassard, G. (Ed.). Threshold Cryptosystems. Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings, Springer, 1989, 435, 307-315

[Shoup00] Shoup, V. Practical Threshold Signatures Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding, 2000, 207-220

[RFC8017] PKCS #1: RSA Cryptography Specifications Version 2.2. https://tools.ietf.org/html/rfc8017