LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

ELIPTISKĀS LĪKNES BĀZĒTA KRIPTOGRĀFISKA IETVARA IZSTRĀDE

KVALIFIKĀCIJAS DARBS

Autors: **Oļegs Pešudovs**

Studenta apliecības Nr.: op14002

Darba vadītājs: dipl.mat. Imants Pops

Rīga, 2016

ANOTĀCIJA

Projekta mērķis ir piedāvāt izstrādātājiem personalizētu kriptogrāfisku sistēmu klienta-servera arhitektūras risinājumiem. Ietvars realizēts izmantojot Latvijā relatīvi jaunu un mazpazīstamu tehnoloģiju, kas fundamentāli balstīta uz eliptisko līkņu pār galīgiem pirmskaitļu laukiem matemātiskajām īpašībām.

Projekta ietvaros tiek izveidots personalizēts informācijas šifrēšanas ietvars un lietotne, kas nodrošina ietvara funkcionalitātes demonstrāciju.

Atslēgas vārdi: Java, JavaScript, API, SAPUI5, ECC, ECDH, RSA, kriptogrāfija

ABSTRACT

The goal of the project is to provide developers with a personalized cryptographic system for client-server architecture solutions. Framework is created using a relatively new and little-known technology within Latvia, which is fundamentally based on mathematical properties of elliptical curves over prime fields.

A personalized information encryption framework and an application that provides demonstration of the product functionality, were created during the project.

Key words: Java, JavaScript, API, SAPUI5, ECC, ECDH, RSA, cryptography

ΠΕΡΙΛΗΨΗ

Ο σκοπός του έργου - να παρέχει στους προγραμματιστές ένα εξατομικευμένο κρυπτογραφικό σύστημα, για την αντιμετώπιση της αρχιτεκτονικής client-server. Το σύστημα έχει δημιουργηθεί με τη χρήση μιας σχετικά νέας τεχνολογίας και ελάχιστα γνωστό στη Λετονία, η οποία βασίζεται στις μαθηματικές ιδιότητες των ελλειπτικών καμπυλών πάνω σε πεπερασμένα πεδία.

Το έργο δημιουργεί ένα εξατομικευμένο σύστημα για την κρυπτογράφηση των πληροφοριών και μια εφαρμογή που παρέχει μια επίδειξη των λειτουργιών του συστήματος

Λέξεις-κλειδιά : Java, JavaScript, API, SAPUI5, ECC, ECDH, RSA, κρυπτογράφηση

SATURS

VĀRDNĪCA

Sesija – stabils savienojums starp lietotāja iekārtu un serveri, kas ietver datu apmaiņu.

Eliptiska līkne – kopa, kuras elementi ir punkti (x, y) kas apmierina $y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0$ kur a, b ir tās līknes parametri.

MySQL – atvērtā pirmkoda datu bāzes vadības sistēma.

SQL – datu bāzu vaicājumu valoda.

DB – datu bāze.

SAPUI5 – objekt orientētais lietotāja saskarnes izstrādes ietvars.

ECDH – Elliptic curve Diffie–Hellman – atslēgu apmaiņas metode eliptiska līknes kriptogrāfijā.

API – iepriekš definēta klašu un procedūru sistēma, kuru var izmantot ārējas programmas.

JSON – JavaScript Object Notation – datu apmaiņas forma.

PPS – Programmatūras prasību specifikācija.

PPA – Programmatūras projektējuma apraksts.

HEX – Heksadecimālā skaitīšanas sistēma.

CVV – Card Verification Value – kredīt kartes verifikācijas numurs

Stack – datu tips, kas ļauj glabāt sevī datus, un atgriezt tos pēc "kas pēdējais ienāca, pirmais izies" principa.

Domēna parametri – eliptiskas līknes parametri.

Abela grupa – komutatīva grupa – šajā grupā, saskatīšanas operācijas rezultāts nav atkarīgs no tās argumentu secības

IEVADS

Šajā darbā tiek aprakstīta eliptiskās līknes bāzēta kriptogrāfiska ietvara izstrāde, kura galvenais mērķis ir dot izstrādātājiem iespēju aizsargāt datus, kas tiek pārsūtīti neaizsargātā tīklā, ar personalizētu šifrēšanas algoritmu. Kā arī tiek aprakstīta testa lietotne, ar kuras palīdzību tiek demonstrēta šifrēšanas algoritma izmantošana.

Izstrādātā ietvara datu šifrēšanas algoritmi fundamentāli balstās uz eliptisko līkņu kriptogrāfiju. Datu šifrēšanai tiek izmantots lietotāja un servera kopējā atslēga, kas tiek ģenerēta asimetriskā veidā pēc *ECDH* principa [1] [2]. Rezultātā, šifrētie dati nav salasāmi bez atslēgas.

Klienta lietotne izstrādāta izmantojot JavaScript programmēšanas valodu un *SAPUI5* biznesa aplikāciju izstrādes ietvaru. Servera lietotne izstrādāta Java programmēšanas valodā. Klienta un servera lietotnes sazinās ar *Jersey REST API* tehnoloģijas palīdzību. Lietotāja un servera dati tiek saglabāti *MySQL* datu bāzē.

Testa lietotnē datu pārsūtīšana laikā tiek šifrēti visi lietotāja personīgie dati, tādi kā lietotāja e-pasts, parole, kredīt kartes informācijā, utt.

Projekta mērķis

Interneta lietojumu daudzveidība un to lietotāju skaits ar katru dienu pieaug, līdz ar to pieaug arī pārsūtītās informācijas apjoms, kas bieži vien satur konfidenciālus datus. Internetā nav centralizētas pārvaldības institūcijas, kas atbild par datu drošību, tāpēc katrs atbild par saviem datiem pats. Eksistē ļoti daudz dažādu datu aizsardzības risinājumu, tomēr tie nenodrošina pilnīgu kontroli par to drošības parametriem.

Projekta mērķis ir piedāvāt izstrādātājiem personalizētu risinājumu datu šifrēšanai, kuru viņi var pilnīgi kontrolēt.

Izstrādātajā ietvarā tiek piedāvāta sava šifrēšanas metode, kura galvenā atšķirība ir punkta M aprēķināšanas algoritms no patvaļīga simbola m. Šajā algoritmā tiek izmantota lietotāja un servera kopēja atslēga S, un nejaušais skaitlis k, lai aprēķināt punktu $M \equiv S \ll m \pmod p$. Tas, salīdzinot ar pašlaik visvairāk izmantoto risinājumu $m \circ P$, teorētiski palielina ātrdarbību jo binära nobīde ir ātrāka par reizinājumu.

Eliptiskās līknes kriptogrāfijā ir diezgan jauna nozare, bet strauji attīstās. Galvenā priekšrocība, salīdzinot ar mūsdienās biežāk izmantoto šifrēšanas algoritmu *RSA*, lai iegūtu tādu pat drošības līmenī, eliptiskās līknes kriptogrāfijā tiek izmantota mazāka bitu garuma atslēgas, kas ļauj to ģenerēt ātrāk (sk. tabula 0.1) [1].

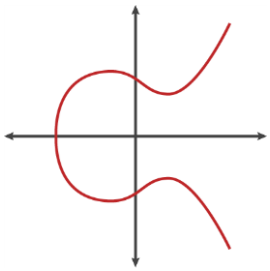
tabula 0.1 **ECC un RSA salīdzinājums**

Key Size (bits)		Generation time (seconds)	
ECC	RSA	ECC	RSA
163	1024	0.08	0.16
233	2240	0.18	7.47
283	3072	0.27	9.89
409	7680	0.64	133.90
571	15360	1.44	679.06

Teorētiskais pamatojums

Par eliptisko līkni

Eliptiska līkne E ir plaknes punktu kopa, kuru definē vienādojums $y^2 = x^3 + ax + b$ (sk. att. 0.1 **Vispārīga eliptiska līkne**att. 0.1), kur a, b nosaka līknes ģeometrisko formu un īpašības [1] [3] [10].



att. 0.1 **Vispārīga eliptiska līkne**

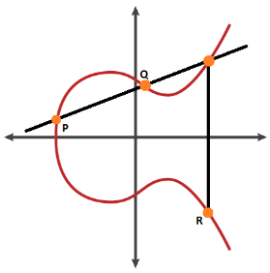
Tiek pieņemts, ja p ir pirmskaitlis, tad veselie skaitļi pēc moduļa p veido lauku $\mathbb{F}_p$ ar pakāpi $\#\mathbb{F}_p = p$. $\mathbb{F}_p$ lauka elementiem ir definētas saskaitīšanas un reizināšanas operācijas:

$$a + b \equiv c \ (mod\ p) \ \text{ kur } a, b, c \ \in \ \mathbb{F}_p$$

$$a * b \equiv c \ (mod\ p) \text{ kur } a, b, c \ \in \ \mathbb{F}_p$$

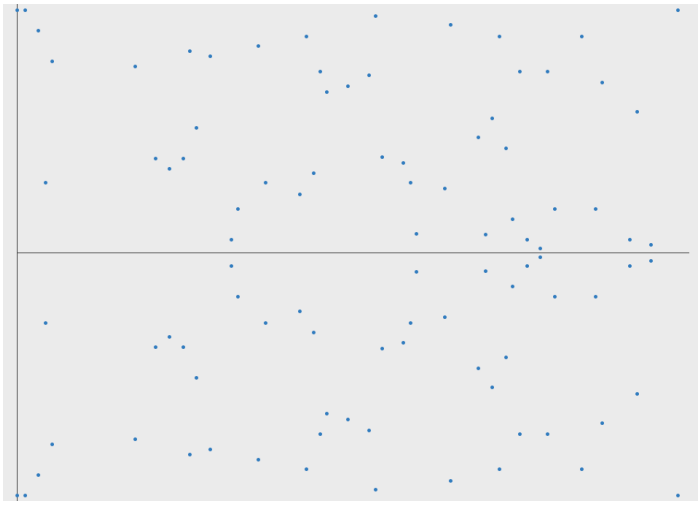
Ja E ir eliptiska līkne un $\mathbb{F}_p$ ir galīgs pirmskaitļa lauks, tad par eliptisko līkni pār galīgu pirmskaitļa lauku sauc sekojošu $\mathbb{F}_p$ veidotās plaknes punktu kopu $E(\mathbb{F}_p) = \{(x, y) |\ y^2 \equiv x^3 + ax + b \ (mod\ p) \wedge (x, y) \in \mathbb{F}_p^2 \ , a, b \in \mathbb{F}_p \wedge 4a^3 + 27b^2 \not\equiv 0 \ (mod\ p)\} \cup \{\mathcal{O}\}$.

$\mathcal{O}$ ir teorētisks punkts pie bezgalības, kur krustojas divas paralēlās taisnes. Definējot to, ir iespējams pārvērst $E(\mathbb{F}_p)$ Abela grupā, kur ir definēta punktu saskaitīšana $P + \mathcal{O} = \mathcal{O} + P = P$ un $P + Q = R$ (sk. att. 0.2) .



att. 0.2 **Punktu saskaitīšana ģeometriski**

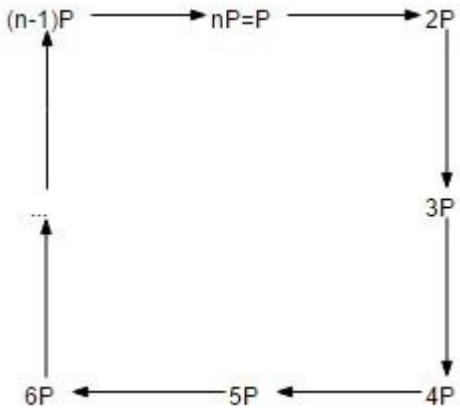
Plaknē eliptiska līkne pār galīgu pirmskaitļa lauku izskatīsies citādāk nekā pati eliptiska līkne (sk. att. 0.3)



att. 0.3 **Līkne** $y^2 = x^3 - x + 1$ **ar** $p = 97$

Par eliptiskas līknes šifrēšanu

Izmantojot eliptisko līkni pār galīgu pirmskaitļa lauku $E(\mathbb{F}_p)$ var izveidot ciklisko grupu. Varbūtiski tiek izvelēts punkts P uz līknes tā, lai tas veidotu ciklisko grupu skalāri reizinot P ar veseliem skaitļiem (sk. att. 0.4).



att. 0.4 **Cikliskas grupas piemērs**

Katrs šifrēšanas algoritm balstās uz kaut kādu problēmu, kuru ir grūti atrisināt ja nav ‘atslēgas’. Eliptiskas līknes kriptogrāfija balstās uz diskrēta logaritma problēmu (ECDLP): zinot $E(\mathbb{F}_p)$, un punktu $Q \in \mathbb{F}_p$, ir ļoti grūti atrast unikālo $x \in \mathbb{Z}$, tā lai $Q = xP$ [1] [10].

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1. Ievads

1.1.1. Nolūks

Programmatūras prasību specifikācija (PPS) paredzēta sistēmas izstrādātājiem, lai precīzi definētu izstrādājamās sistēmas funkcionālās un nefunkcionālās prasības.

1.1.2. Darbības sfēra

Programmatūras produktam jānodrošina:

- publisko un privāto atslēgu ģenerēšanu;
- atslēgu apmaiņas procedūru;
- simbolu virkņu šifrēšanas un dešifrēšanas procedūru;
- datu pārsūtīšanu starp klientu un serveri šifrētā veidā.

1.1.3. Saistība ar citiem dokumentiem

Dokumenta noformēšanā ievērotas standarta LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" prasības.

1.1.4. Definīcijas

tabula 1.1 **PPS definīcijas**

Saīsinājums	**Skaidrojums**
p	pirmskaitlis
$\mathbb{F}_p$	$\mathbb{F}_p = \mathbb{Z}_p$
a, b	eliptiskas līknes koeficienti $a, b \in \mathbb{F}_p$
plaknes punkts	punkts $= \{\forall(x, y) \mid x, y \in \mathbb{F}_p\}$
līknes punkts	plaknes punkts, kur $y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0$
P	Cikliskas apakšgrupas primitīvais elements - līknes punkts, kas tiek izmantots kā ģenerators cikliskai grupai
n	$n = \# < P, +.\circ >$
privāta atslēga	$d \in [0, n-1] \mid d \in \mathbb{F}_p$, tiek noslēpta un netiek publicēta
publiska atslēga	līknes punkts, kas tiek aprēķināts izpildot skalāras reizināšanas operāciju ar privāto atslēgu un cikliskas apakšgrupas primitīvo elementu

kopēja atslēga	līknes punkts, kas tiek aprēķināts reizinot viena lietotajā privāto atslēgu ar cita lietotāja publisko, un netiek publicēts
DB	datu bāze
Q, P	dotie plaknes punkti
simbols	ASCII tabulas elements
teksts	patvaļīga simbolu virkne
šifrēts teksts	saraksts ar plaknes punktu koordinātām un nobīdes vērtību, kas veido trijnieku (k, x, y)

1.1.5. Pārskats

Šis dokuments sastāv no 5 nodaļām un pielikuma.

1. nodaļā ir ievads, kurā tiek aprakstīts dokumenta mērķis un dokumenta struktūra.
2. nodaļā ir dots izstrādājamās programmas vispārīgs apraksts.
3. nodaļā ir detalizēti aprakstītas sistēmas funkcijas.
4. nodaļā ir noradītas saskarnes prasības.
5. nodaļā ir aprakstītas sistēmas nefunkcionālās prasības.

Pielikums sastāv no papildinošas informācijas, kas tiek minēta dokumentā.

1.2. Vispārējais apraksts

1.2.1. Produkta perspektīva

Produkts palīdz citiem izstrādātājiem, piedāvājot personalizētu risinājumu datu šifrēšanai, kurā izstrādātājs var pats nodefinēt parametrus. Šo produktu varēs izmantot, lai aizsargātu pārsūtamo informāciju neaizsargātā tīklā.

1.2.2. Produkta funkcijas

Produktam jāspēj ģenerēt privātas (skaitlis) un publiskas (punkts) kriptogrāfiskas atslēgas balstoties uz dotiem eliptiskas līknes kriptosistēmas domēna parametriem.

Implementējamajam datu aizsardzības risinājumam jānodrošina servera puses un klienta puses publisko atslēgu apmaiņas procedūra un atbilstošo, kopējo simetrisko atslēgu ģenerēšana.

Jānodrošina simbolu virkņu šifrēšanas un dešifrēšanas procedūra, izmantojot kopējo, simetrisko atslēgu un atbilstošas formulas, lai virknes saturs nemainās pēc šifrēšanas un dešifrēšanas.

Lietotāju personīgie dati un atbilstoša formāta metadati, kas tiek pārsūtīti starp klienta sistēmu un serveri, tiek šifrēti tā, lai bez atbilstošās kriptogrāfiskās atslēgas, to atšifrēšana nozīmētu algoritmiski sarežģītās diskrētā logaritma problēmas eliptiskajām līknēm atrisināšanu.

1.2.3. Lietotāju raksturiezīmes

Lietotājiem nepieciešama prasme darbam ar datoru, uzstādīto interneta pārlūkprogrammu, ar *REST API* pieprasījumiem.

Tiek sagaidīta padziļināta izpratne datu aizsardzības un kriptogrāfijas problemātikā.

Ja lietotājs ir izstrādātājs, tad jābūt zināšanām par datu šifrēšanu un iespējams arī datu šifrēšanas algoritmiem.

1.2.4. Vispārējie ierobežojumi

Klienta puses Sistēmas darbība tiek nodrošināta, izmantojot tīmekļa pārlūkprogrammu *Chrome 50.0.2661.94 m,* kas atbalsta *JavaScript, HTML5, CSS*. Sistēma izmanto *SAP* biznesa lietotņu izstrādes ietvaru *SAPUI5*.

Servera puses sistēmas darbību nodrošina *Java EE* lietotne un *Apache Tomcat* tīmekļa serveris.

Saziņai starp klienta puses sistēmu un servera puses sistēmu jābūt realizētai kā *REST* pieprasījumu servisiem.

1.2.4.1. Pieņēmumi un atkarības

Sistēma atbalsta datu šifrēšanu ar vienas eliptiskas līknes parametriem, kurus izstrādātājs var izvelēties ietvara ieviešanas laikā. Ja rodas nepieciešamība ieviest vairākas līknes un ļaut lietotājam izvelēties, sistēmu būs nepieciešams papildināt.

Sistēma spēj strādāt ar 512 bitu gariem domēna parametriem.

1.3. Funkcionālās prasības

1.3.1. Sistēmas funkcijas

1.3.1.1. Prasības identifikators: setKeyLength

Sistēmas stāvokļa S nosacījumi:

- Nav definēts atslēgu garums

Operācijas ieejas dati:

- $len \in \mathbb{N}, len > 0$

Operācijas izejas dati:

- $\emptyset$

Datu objektu transformācijas:

- $\text{setKeyLength}: \{S, len\} \rightarrow \{S'\}, S' = S \cup len$

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
    - Atslēgu garums jau ir definēts $len \neq null$
- Nekorekti ieejas dati
    - $len = null$
    - $len < 0$

1.3.1.2. Prasības identifikators: setPrimeField

Sistēmas stāvokļa S nosacījumi:

- Nav definēts galīgs lauks $\mathbb{F}_p = \emptyset$
- Definēts publisko atslēgu bitu garums *len* (prasība *1.3.1.1. setKeyLength*)

Operācijas ieejas dati:

- $p \in \mathbb{N}, p - pirmskaitlis, \log_2 p > len$

Operācijas izejas dati:

- $\mathbb{F}_p$

Datu objektu transformācijas:

- $\mathbb{F}_p = \mathbb{Z}_p$
- $\text{setPrimeField}: \{S, p\} \rightarrow \{S', \mathbb{F}_p\}, S' = S \cup \mathbb{F}_p$

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
    - Lauks jau ir definēts $\mathbb{F}_p \neq \emptyset$
    - Nav definēts publisko atslēgu garums *len=null*

- Nekorekti ieejas dati
  - p nav pirmskaitlis
  - $\log_2 p < len$

1.3.1.3. Prasības identifikators: setEllipticCurve

Sistēmas stāvokļa S nosacījumi:

- Nav definēta eliptiska līkne $E = \emptyset$

Operācijas ieejas dati:

- $p \in \mathbb{N}, p - \text{pirmskaitlis}$
- $a \in \mathbb{Z}$
- $b \in \mathbb{Z}$
- $P \in \mathbb{F}_p \times \mathbb{F}_p$
- $n \in \mathbb{N}, n - \text{pirmskaitlis}, n < p$
- $h \in \mathbb{N}$

Operācijas izejas dati:

- $E = (p, a, b, P, n, h)$

Datu objektu transformācijas:

- $setEllipticCurve: \{S, p, a, b, P, n, h\} \rightarrow \{S', E\}, S' = S \cup E$

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati

1.3.1.4. Prasības identifikators: pointSum

Sistēmas stāvokļa S nosacījumi:

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.4. setEllipticCurve*)
- Definēta Eliptiskās līknes punktu kopa $E(\mathbb{F}_p) \neq \emptyset$ (prasība *setEllipticCurvePoints*)

Operācijas ieejas dati:

- Plaknes punkts $Q1 = (q1_x, q1_y)$: $q1_x, q1_y \in \mathbb{Z}_p \wedge Q1 \in E(\mathbb{F}_p)$

- Plaknes punkts $Q2 = (q2_x, q2_y)$: $q2_x, q2_y \in \mathbb{Z}_p \wedge Q2 \in E(\mathbb{F}_p)$

Operācijas izejas dati:

- Plaknes punkts $Q = (q_x, q_y)$: $q_x, q_y \in \mathbb{Z}_p \wedge Q \in E(\mathbb{F}_p)$

Datu objektu transformācijas:

- $pointSum: \{S, Q1, Q2\} \rightarrow \{S', Q\}, S = S'$
- funkcijas $\{Q1, Q2\} \rightarrow Q$ deklarācija:
  - **ja** $Q1 = Q2$
    1. $L \equiv (3 * {Q2_x}^2 + a) / (2 * Q2_y) \pmod p$
    2. $TQ_x \equiv L^2 - Q2_x - Q2_x \pmod p$
    3. $TQ_y \equiv L(Q2_x - TQ_x) - Q2_y \pmod p$
    4. $Q = (TQ_x ; TQ_y)$
  - **ja** $Q1 \neq Q2$
    1. $L \equiv (Q1_y - Q2_y) / (Q1_x - Q2_x) \pmod p$
    2. $TQ_x \equiv L^2 - Q2_x - Q1_x \pmod p$
    3. $TQ_y \equiv L(Q2_x - TQ_x) - Q2_y \pmod p$
    4. $Q = (TQ_x ; TQ_y)$

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati
  - Nekorekts $Q1$
  - Nekorekts $Q2$
  - Nekorekts $Q1$ un $Q2$

1.3.1.5. Prasības identifikators: pointMultiply

Sistēmas stāvokļa S nosacījumi:

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.3. setEllipticCurve*)

Operācijas ieejas dati:

- $n \in \mathbb{N}, n > 0$
- Plaknes punkts $Q1 = (q1_x, q1_y)$: $q1_x, q1_y \in \mathbb{Z}_p \wedge Q1 \in E(\mathbb{F}_p)$

Operācijas izejas dati:

- Plaknes punkts $Q = (q_x, q_y)$: $q_x, q_y \in \mathbb{Z}_p \wedge Q \in E(\mathbb{F}_p)$

Datu objektu transformācijas:

- $pointMultiply: \{S, n, Q1\} \rightarrow \{S', Q\}, S = S'$
- funkcijas $\{n, Q1\} \rightarrow Q$ deklarācija:

1. $Q = Q1$
2. $n\%2$
   - ja $n\%2 == 0$, stack saglabā 2
   - ja $n\%2 == 1$, stack saglabā 1
3. $n = n/2$
4. atkārtojam kamēr n!=0
5. **izmetam no *stack* pirmo skaitli, kas vienmēr ir 1, jo 2/2 ir 1 ņemam no *stack* 2 vai 1:**
   - ja $2: Q = Q + Q$
   - ja $1: Q = Q + Q1$ *(ja* $Q = Q1$, *tad izmanto* $Q + Q$*)*
6. atkārtojam, kamēr *stack* nav tukšs

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati
  - Nekorekts n
    - $n \leq 0$
    - $n = null$
  - Nekorekts $Q1$
  - Nekorekts n un $Q1$

1.3.1.6. Prasības identifikators: mapCharToPoint

Sistēmas stāvokļa S nosacījumi:

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.3. setEllipticCurve*)

Operācijas ieejas dati:

- ASCII simbols m

- Plaknes punkts $Q = (q_x, q_y)$: $q_x, q_y \in \mathbb{Z}_p \wedge Q \in E(\mathbb{F}_p)$

Operācijas izejas dati:

- Plaknes punkts $M = (m_x, m_y)$: $m_x, m_y \in \mathbb{Z}_p \wedge M \in E(\mathbb{F}_p)$

Datu objektu transformācijas:

- $mapCharToPoint: \{S, m, Q\} \rightarrow \{S', M\}, S = S'$
- funkcijas $\{m, Q\} \rightarrow \{M\}$ deklarācija:

1. $M \equiv Q << m \pmod p$

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati
  - Nekorekts m
  - Nekorekts Q
  - Nekorekts m un Q

1.3.1.7. Prasības identifikators: encryptMessage

Sistēmas stāvokļa S nosacījumi :

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.3. setEllipticCurve*)

Operācijas ieejas dati:

- Patvaļīgs teksts s
- Kopēja atslēga $Q = (q_x, q_y)$: $q_x, q_y \in \mathbb{Z}_p \wedge Q \in E(\mathbb{F}_p)$

Operācijas izejas dati:

- Masīvs H no elementiem:
  - $k \in [0, n-1]$
  - Plaknes punkts $C = (c_x, c_y)$: $c_x, c_y \in \mathbb{Z}_p \wedge C \in E(\mathbb{F}_p)$

Datu objektu transformācijas:

- $encryptMessage: \{S, s, Q\} \rightarrow \{S', H\}, S = S'$
- funkcijas $\{s, Q\} \rightarrow \{H\}$ deklarācija :
  - ar katru simbolu m no teksta s:

1. **simbols m tiek pārveidots punktā M (prasība *1.3.1.6. mapCharToPoint*)**

2. **tiek ģenerēts nejaušs punkts no kopējā noslēpuma:**
3. $TS_x = k * Q_x$
4. $TS_y = Q_y$
5. $TS = (TS_x ; TS_y)$
6. **nejaušais punkts tiek saskatīts ar simbola punktu:**
7. $C = M + TS$
8. C tiek pievienots masīvam H

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati
  - $s = null$
  - $Q = null$

1.3.1.8. Prasības identifikators: decryptMessage

Sistēmas stāvokļa S nosacījumi:

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.3. setEllipticCurve*)

Operācijas ieejas dati:

- Masīvs H no elementiem:
  - $k \in [0, n-1]$
  - Plaknes punkts $C = (c_x, c_y)$: $c_x, c_y \in \mathbb{Z}_p \wedge C \in E(\mathbb{F}_p)$
- Kopēja atslēga $Q = (q_x, q_y)$: $q_x, q_y \in \mathbb{Z}_p \wedge Q \in E(\mathbb{F}_p)$

Operācijas izejas dati:

- Patvaļīgs teksts s

Datu objektu transformācijas:

- $decryptMessage: \{S, H, Q\} \rightarrow \{S', s\}, S = S'$
- funkcijas $\{H, Q\} \rightarrow \{s\}$ deklarācija:
  - **ar katru H elementu :**

    1. **tiek ģenerēts punkts no dotā k un kopējā noslēpuma** Q**:**
    2. $TS_x = k * Q_x$
    3. $TS_y = Q_y$
    4. $TS = (TS_x ; TS_y)$

5. **tiek noskaidrots simbola punkts:**
6. $M = C - TS$
7. **punkts** M **tiek pārveidots simbolā** m **(prasība** *1.3.1.9. mapPointToChar***)**
8. **simbols** m **tiek pievienots tekstam** s

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati
  - $H = null$
  - H nesatur vajadzīgus elementus
  - $Q = null$

1.3.1.9. Prasības identifikators: mapPointToChar

Sistēmas stāvokļa S nosacījumi:

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.3. setEllipticCurve*)

Operācijas ieejas dati:

- Plaknes punkts $C = (c_x, c_y)$: $c_x, c_y \in \mathbb{Z}_p \wedge C \in E(\mathbb{F}_p)$
- Plaknes punkts $Q = (q_x, q_y)$: $q_x, q_y \in \mathbb{Z}_p \wedge Q \in E(\mathbb{F}_p)$

Operācijas izejas dati:

- ASCII simbols m

Datu objektu transformācijas:

- $mapPointToChar: \{S, C, Q\} \rightarrow \{S', m\}, S = S'$
- funkcijas $\{C, Q\} \rightarrow \{m\}$ deklarācija :

1. $i = 0$
2. **kamēr** $TS \neq C$**:**
   - $TS \equiv Q << i \ (mod\ p)$
   - $i++$
3. **saņemtais** i **ir simbola vērtība**
4. $m = (char)i$
5. **simbols** m **ir atrasts**

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju
- Nekorekti ieejas dati
    - Nekorekts C
    - Nekorekts Q
    - Nekorekts C un Q

1.3.1.10. Prasības identifikators: generateKeyPair

Sistēmas stāvokļa S nosacījumi :

- Definēts galīgs lauks $\mathbb{F}_p \neq \emptyset$ (prasība *1.3.1.2. setPrimeField*)
- Definēta Eliptiskā līkne $E = (p, a, b, P, n, h)$ (prasība *1.3.1.3. setEllipticCurve*)

Operācijas ieejas dati:

- $\emptyset$

Operācijas izejas dati:

- Skaitlis $d \in [0, n]$
- Plaknes punkts $X = (x_1, x_2)$: $x_1, x_2 \in \mathbb{Z}_p \wedge X \in E(\mathbb{F}_p)$

Datu objektu transformācijas:

- $generateKeyPair: \{S\} \rightarrow \{S', d, X\}, S = S'$

Definētie izņēmumi:

- Nekorekts stāvoklis izsaucot operāciju

1.4. Ārējās saskarnes prasības

1.4.1. Sistēmas saskarnes

Komunikācija ar serveri norisinās ar *REST API* pieprasījumiem un atbildēm, kas satur informāciju *JSON* formātā atbilstoši izsauktajām metodēm.

1.4.1.1. connect

Šī metode pieprasa ECC parametrus no DB, kas tiek vēlāk izmantoti visiem aprēķiniem šifrēšanas/dešifrēšanas laikā un atslēgu ģenerēšanai.

Method	POST	
URL	/rest/api/connect	
Content-Type	application/json	
POST parameters	None	
Success response	response: Success!	
	id	eliptiskas līknes id
	name	eliptiskas līknes nosaukums
	p	eliptiskas līknes primārs skaitlis, kas tiek izmantots lai noskaidrot līknes plaknes robežas
	a	eliptiskas līknes parametrs
	b	eliptiskas līknes parametrs
	n	eliptiskas līknes primārs skaitlis, kas tiek izmantots lai noskaidrot līknes cikliskas grupas robežas
	h	eliptiskas līknes kofaktors
	Px	cikliskas apakšgrupas primitīva elementa x koordināta
	Py	cikliskas apakšgrupas primitīva elementa x koordināta
Error response	response: Connection error!	

1.4.1.2. getSession

Šī metode pieprasījumā nodod sistēmai lietotāja publisko atslēgu, kas tiek saglabāta un sasaistīta ar pieprasījuma sesijas identifikatoru, un saņem sesijas identifikatoru ar servera publisko atslēgu.

Method	POST	
URL	/rest/api/getSession	
Content-Type	application/json	
POST parameters	x	Lietotāja publiskas atslēgas x koordināta
	y	Lietotāja publiskas atslēgas y koordināta
Success response	response: Success!	
	session	sesijas id
	x	servera publiskas atslēgas x koordināta
	y	servera publiskas atslēgas y koordināta
Error response	response: Server failed to establish session!	

1.4.1.3. register

Šī metode pieprasa DB izveidot jauno lietotāju ar dotajiem parametriem.

Pieprasījums ir pilnīgi nošifrēts pirms sūtīšanas!

Method	POST	
URL	/rest/api/register	
Content-Type	application/json	
POST parameters	email	Lietotāja e-pasts
	message	Lietotāja parole, iekodēta ar SHA1 algoritmu
Success response	response: User created successfully!	
Error response	response: User already exists!	

1.4.1.4. login

Šī metode pieprasa DB pārbaudīt lietotāju datus.

Pieprasījums ir pilnīgi nošifrēts pirms sūtīšanas!

Method	POST	
URL	/rest/api/login	
Content-Type	application/json	
POST parameters	email	Lietotāja e-pasts
	message	Lietotāja parole, iekodēta ar SHA1 algoritmu
Success response	response: Success!	Lietotāja identifikators, piesaistīts viņa sesijas identifikatoram
Error response	response: User does not exist or wrong data!	

1.4.1.5. getInfo

Šī metode pieprasījumā nodod sistēmai lietotāja e-pastu, saņem lietotāja saglabāto informāciju. Sistēma arī pārbauda, vai pieprasījuma autoram ir atļauja šos datus saņemt.

Pieprasījums ir pilnīgi nošifrēts pirms sūtīšanas!

Method	POST	
URL	/rest/api/getInfo	
Content-Type	application/json	
POST parameters	email	Lietotāja e-pasts
Success response	response: Success!	
	message	Lietotāja saglabātā informācija. **Šī informācijā tiek saņemta šifrētā veidā.**
Error response	response	kļūdas paziņojums ar identifikatoru

1.4.1.6. setInfo

Šī metode pieprasījumā nodod sistēmai lietotāja e-pastu un informāciju, ko viņš grib saglabāt. Sistēma arī pārbauda, vai pieprasījuma autoram ir atļauja šos datus saglabāt.

Pieprasījums ir pilnīgi nošifrēts pirms sūtīšanas!

Method	POST	
URL	/rest/api/setInfo	
Content-Type	application/json	
POST parameters	email	Lietotāja e-pasts
	message	Lietotāja saglabājamā informācija
Success response	response: Success!	
Error response	response	kļūdas paziņojums ar identifikatoru

1.4.1.7. getCard

Šī metode pieprasījumā nodod sistēmai lietotāja e-pastu, saņem lietotāja saglabāto kredīt kartes informāciju. Sistēma arī pārbauda, vai pieprasījuma autoram ir atļauja šos datus saņemt.

Pieprasījums ir pilnīgi nošifrēts pirms sūtīšanas!

Method	POST	
URL	/rest/api/getCard	
Content-Type	application/json	
POST parameters	email	Lietotāja e-pasts
Success response	response: Success!	
	message ○ cardName ○ cardNumber ○ cardExpire ○ cardCVV	Lietotāja saglabātā informācija. **Šī informācijā tiek saņemta šifrētā veidā un satur parametrus *JSON* formātā.**
Error response	response	kļūdas paziņojums ar identifikatoru

1.4.1.8. setCard

Šī metode pieprasījumā nodod sistēmai lietotāja e-pastu un kredīt kartes informāciju, ko viņš grib saglabāt. Sistēma arī pārbauda, vai pieprasījuma autoram ir atļauja šos datus saglabāt.

Pieprasījums ir pilnīgi nošifrēts pirms sūtīšana!

Method	POST
URL	/rest/api/setInfo

Content-Type	application/json	
POST parameters	email	Lietotāja e-pasts
	cardName	Lietotāja kredīt kartes nosaukums
	cardNumber	Lietotāja kredīt kartes numurs
	cardExpire	Lietotāja kredīt kartes derīguma termiņš
	cardCVV	Lietotāja kredīt kartes verifikācijas numurs
Success response	response: Success!	
Error response	response	kļūdas paziņojums ar identifikatoru

1.4.1.9. testCurve

Šī metode ir izveidota, lai pārbaudītu veiktspēju dažādam līknēm. Tiek noteikts privātās un publiskās atslēgas ģenerēšanas laiks.

Method	POST	
URL	/rest/api/testCurve	
Content-Type	application/json	
POST parameters	times	cik reizes pārbaudīt
	p	eliptiskas līknes primārs skaitlis, kas tiek izmantots lai noskaidrotu līknes plaknes robežas
	a	eliptiskas līknes parametrs
	b	eliptiskas līknes parametrs
	n	eliptiskas līknes primārs skaitlis, kas tiek izmantots lai noskaidrotu līknes cikliskās grupas robežas
	h	eliptiskas līknes kofaktors
	px	cikliskas apakšgrupas primitīva elementa x koordināta
	py	cikliskas apakšgrupas primitīva elementa x koordināta

Success response	response: [PriveKey,PubKey,Total]	kopa, kur elementi satur privātās un publiskās atslēgas ģenerēšanas laiku un to summu.
Error response	response: null	

1.4.1.10. addCurve

Šī metode ļauj saglabāt notestētas līknes parametrus datu bāzē ar patvaļīgu nosaukumu.

Method	POST	
URL	/rest/api/addCurve	
Content-Type	application/json	
POST parameters	name	nosaukums, ar kuru saglabāt
	p	eliptiskas līknes primārs skaitlis, kas tiek izmantots lai noskaidrotu līknes plaknes robežas
	a	eliptiskas līknes parametrs
	b	eliptiskas līknes parametrs
	n	eliptiskas līknes primārs skaitlis, kas tiek izmantots lai noskaidrotu līknes cikliskās grupas robežas
	h	eliptiskas līknes kofaktors
	px	cikliskas apakšgrupas primitīva elementa x koordināta
	py	cikliskas apakšgrupas primitīva elementa x koordināta
Success response	response: Success!	
Error response	response	kļūdas paziņojums ar identifikatoru

1.4.2. Lietotāja saskarnes

Komunikācija ar lietotāju noris ar vienkāršas, viegli saprotamas *SAPUI5* lietotnes palīdzību.

Lietotājs var ielogoties un saglabāt kaut kādu informāciju. Informācija glabāsies šifrētā veidā, bet administrators var to redzēt atklātā veidā.

1.5. Nefunkcionālās prasības

1.5.1. Veiktspējas prasības

Sistēmai jāveic šifrēšanas/dešifrēšanas operācija laikā, kas nepārsniedz 5 sekundes, pie nosacījuma, ka šifrējamā simbolu virkne nepārsniedz 5000 simbolu, tiek izmantota 256 bitu garuma atslēgas un procedūra noris uz sistēmas ar Intel® Core™ i5-4460 @3.20GHz procesoru.

1.5.2. Drošības prasības

Informācija, kuru vajag aizsargāt, tiek šifrēta:

- ar lietotāja un servera kopējo atslēgu, kad tiek pārsūtīta;
- ar servera atslēgu, kad tiek saglabāta datu bāzē.

Jānodrošina, lai piekļuve lietotāju informācijai būtu tikai pašam lietotājam un administratoram.

Jānodrošina arī informācijas metadatu, tādu kā formāta apraksts, šifrēšana.

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1. Ievads

2.1.1. Dokumenta nolūks

Šis dokuments ir projekta "Eliptiskās līknes bāzēta kriptogrāfiska ietvara izstrāde" izstrādes un ieviešanas projekta nodevums.

Šajā dokumentā tiek aprakstīta programmatūras prasību specifikācijā definēto prasību implementācija konkrētās izstrādes vidēs un tehnoloģijās, kā arī tiek parādīts, kā programmatūras sistēma tiek strukturēta.

2.1.2. Darbības sfēra

Programmatūras darbības sfēra ir aprakstīta *Programmatūras prasību specifikācijā* nodaļā *Darbības sfēra*.

2.1.3. Saistība ar citiem dokumentiem

Projekta "Eliptiskās līknes bāzēta kriptogrāfiska ietvara izstrāde" *Programmatūras projektējuma apraksts* lietojams kopā ar produkta *Programmatūras prasību specifikāciju*.

PPA satura organizācija balstās uz standarta LVS 72:1996 "Ieteicamā prakse programmatūras projektējumā aprakstīšanai" prasībām.

2.1.4. Pārskats

Projekta "Eliptiskās līknes bāzēta kriptogrāfiska ietvara izstrāde" PPA sastāv no 4 nodaļām:

Ievads, kas satur vispārīgu informāciju par dokumenta nolūku, darbības sfēru un saistību ar citiem dokumentiem;

Tehniskie risinājumi, kas iekļauj informāciju par projektā izmantotam tehnoloģijām un izstrādes vidi;

Dekompozīcijas apraksts, kas satur projekta konceptam atbilstošo funkciju aprakstu;

Prasību trasējamības matrica, kas attēlo PPS prasību realizāciju PPA.

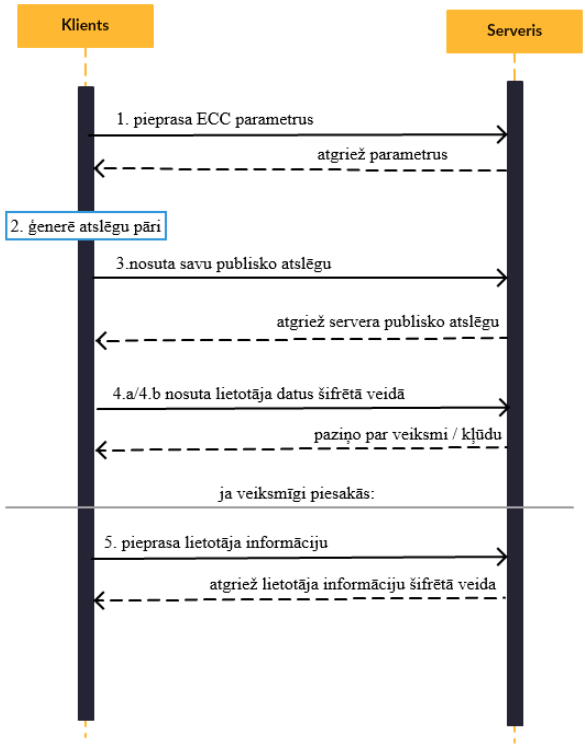
2.2. Tehniskie risinājumi

Projekts "Eliptiskās līknes bāzēta kriptogrāfiska ietvara izstrāde" tiek izstrādāts izmantojot *Java* un *JavaScript* programmēšanas valodas. Datubāzes funkcionalitāti nodrošina *MySQL* 5.7.11. Visas datu bāzes tabulas izmanto *InnoDB* zema līmeņa datu glabāšanas apakšsistēmu.

Klienta lietotnes izstrādē tiek pielietots *JavaScript* lietotņu izstrādes ietvars *SAPUI5 1.36.9*, kas nodrošina MVC arhitektūras koncepcijai atbilstošu risinājumu implementācijas iespējas.

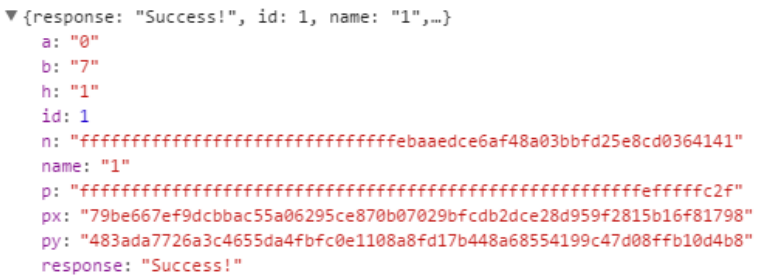
Servera lietotnes arējas saskarnes izstrādē tiek pielietots *Jersey REST API*, kas ļauj servera un klienta lietotnēm komunicēties ar *REST API* pieprasījumiem *JSON* formātā.

2.2.1. Arhitektūra



att. 2.1 **klienta-servera komunikācijas secība**

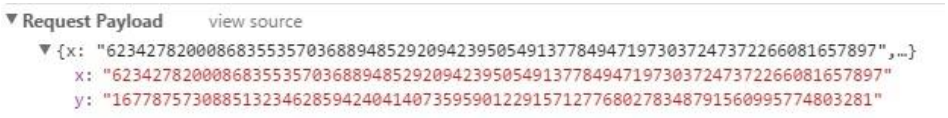
1. Uzsākot saziņu, klienta – servera sistēma vienojas par domēna parametriem. Lietotājs inicializē lietotni, serverim tiek pieprasīta ECC domēna parametru informācija, kas tiek glabāta servera puses datu bāzē (sk. att. 2.2).



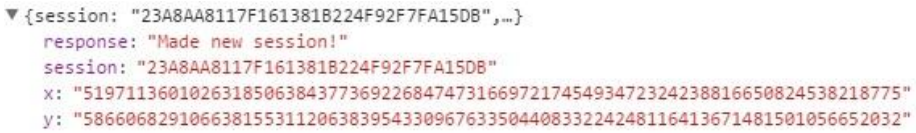
att. 2.2 **Saņemtie ECC domēna parametri**

2. Izmantojot saņemtos kriptosistēmas domēna parametrus, klients ģenerē kriptogrāfisko atslēgu pāri.

3. Lietotāja publiska atslēga tiek aizsūtīta uz serveri (sk. att. 2.3). Kā atbilde tiek saņemta servera publiskā atslēga un sesijas identifikators (sk. att. 2.4).



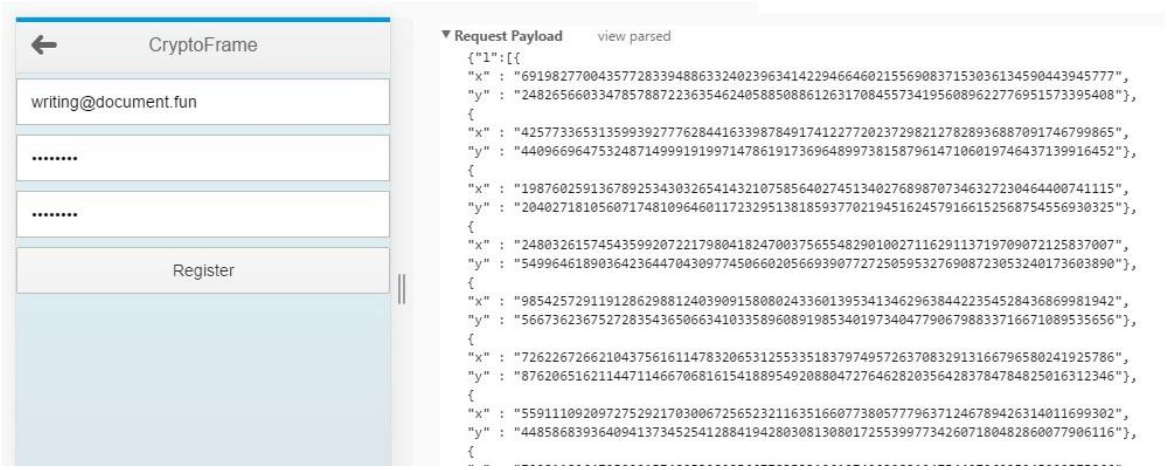
att. 2.3 **Lietotāja aizsūtītie dati**



att. 2.4 **Lietotāja saņemtie dati**

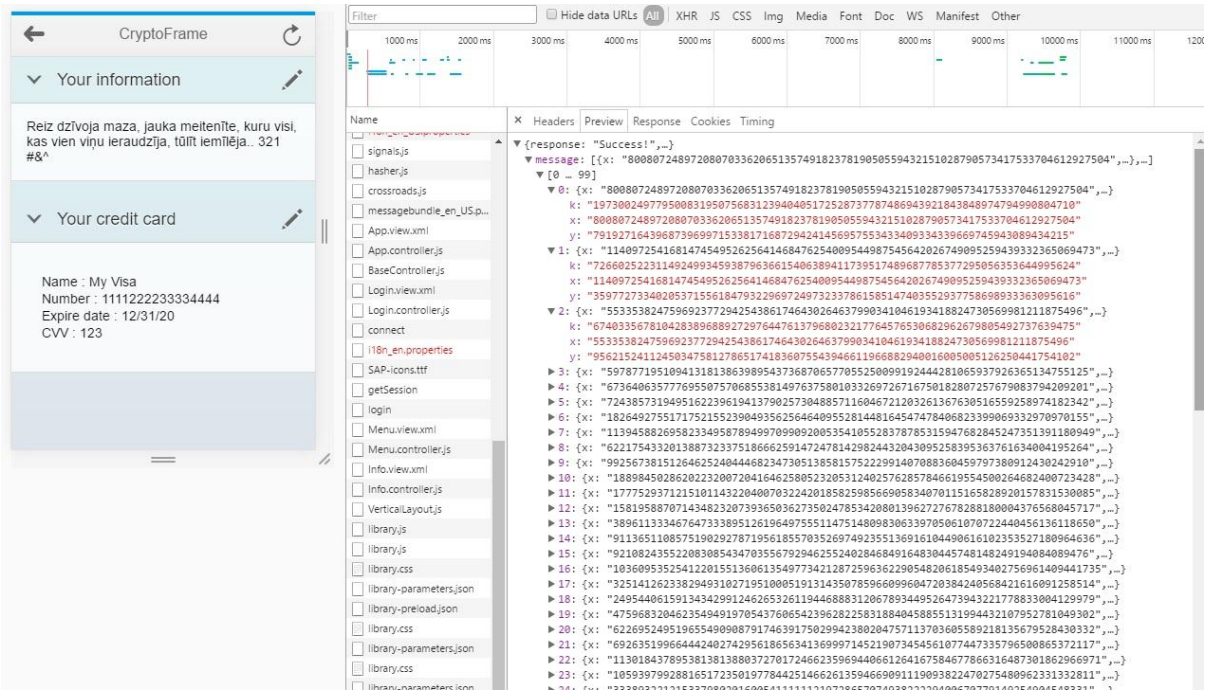
4. Lietotāja lietotne ir inicializēta.
    a. Lietotājs piesakās – tiek aizsūtīti lietotāja e-pasts un parole uz serveri šifrētā veidā. Ja serveris pēc dešifrēšanas secināja, ka dati ir pareizi, datu bāzē pie sesijas identifikatora tiek pierakstīts lietotājs un serveris atbild par veiksmi. Pretējā gadījumā serveris atbild par kļūdu.

    b. Lietotājs reģistrējas – tiek aizsūtīti lietotāja e-pasts un parole uz serveri šifrētā veidā (sk. att. 2.5). Ja serveris pēc dešifrēšanas secināja, ka dati ir pareizi, serveris saglabā tos datu bāzē un atbild par veiksmi. Pretējā gadījumā serveris atbild par kļūdu.



att. 2.5 **Lietotāja dati šifrētā veidā**

5. Lietotājs pieprasa savu informāciju – tiek aizsūtīts lietotāja e-pasts uz serveri šifrētā veidā. Serveris dešifrē e-pastu, pārbauda vai ielogotam lietotājam ir tiesības piekļūt pieprasītajai lietotāja informācijai un atsuta informāciju šifrētā veidā (sk. att. 2.6).



att. 2.6 **Lietotāja kredīt kartes un cita saglabāta informācija tiek šifrēta**

2.3. Dekompozīcijas apraksts

Šeit tiek aprakstītas pamata funkcijas. Tiek pieņemts ka *Java* un *JavaScript* funkcijas atšķiras tikai sintaktiski (valodu atšķirības dēļ), tāpēc funkcijas netiek dublētas, kas atvieglo lasāmību.

Funkcijām tiek ģenerēta automātiska dokumentācija ar:

- *Java* funkcijām – ar *JavaDoc* palīdzību (sk. Pielikums 4. JavaDoc)
- *JavaScript* funkcijām – ar *JSDoc* palīdzību (sk. Pielikums 5. JSDoc)

2.3.1. Ecc klase

Ecc klase ir galvenā klase, kas satur visus nepieciešamos līknes parametrus, matemātiskas funkcijas un šifrēšanas algoritmus. Konstruktorā tiek definēti visi eliptiskās līknes parametri, bez kuriem klases metodes nestrādās.

Metode		
Ecc		
Metodes identifikators		
2.3.1.1. ecc.constructor		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
sp	String	eliptiskas līknes p parametrs
sa	String	eliptiskas līknes a parametrs
sb	String	eliptiskas līknes b parametrs
sPx	String	eliptiskas līknes cikliskas apakšgrupas primitīva elementa P x koordināta
sPy	String	eliptiskas līknes cikliskas apakšgrupas primitīva elementa P y koordināta
sn	String	eliptiskas līknes n parametrs

Metode
Ecc.pointMultiply
Metodes identifikators
2.3.1.2. ecc.pointMultiply

Metodes apraksts		
Aprēķina punkta reizinājumu ar skaitli, izmantojot eliptiskas līknes punktu reizinājuma formulu (sk. Pielikums 1. Formulu lapa - Formula Nr.2 - Punktu reizinājums)		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
point	Point	punkts, kuru vajag reizināt
d	BigInteger	skaitlis, ar kuru reizina punktu
p	BigInteger	pirmskaitlis, eliptiskas līknes parametrs
Izvada dati		
	Point	jauns punkts

Metode		
Ecc.pointSum		
Metodes identifikators		
2.3.1.3. ecc.pointSum		
Metodes apraksts		
Aprēķina divu punktu summu, izmantojos eliptiskas līknes punktu saskaitīšanas formulu (sk. Pielikums 1. Formulu lapa - Formula Nr.1 - Punktu saskaitīšana)		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
point1	Point	punkti, kurus vajag saskaitīt
point2	Point	
Izvada dati		
	Point	jauns punkts

Metode		
Ecc.encryptMessage		
Metodes identifikators		
2.3.1.4. ecc.encryptMessage		
Metodes apraksts		
Šifrē doto simbolu virkni, pielietojot šifrēšanas formulas (sk. Pielikums 1. Formulu lapa - Formula Nr.4 - Simbola šifrēšana) katram virknes simbolam, izmantojot doto atslēgu		
Ievad dati		

Nosaukums	Datu tips	Apraksts
message	String	teksts, kuru vajag nošifrēt
key	Point	atslēga, ar kuru šifrēt
Izvada dati		
	EncodedMsg	šifrēts teksts

Metode		
Ecc.decryptMessage		
Metodes identifikators		
2.3.1.5. ecc.decryptMessage		
Metodes apraksts		
Dešifrē doto šifrēto tekstu pielietojot dešifrēšanas formulas (sk. Pielikums 1. Formulu lapa - Formula Nr.5 - Punkta atšifrēšana) katram elementam, izmantojot doto atslēgu		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
encMessage	EncodedMsg	šifrēts teksts
key	Point	atslēga, ar kuru atšifrēt
Izvada dati		
	String	teksts

Metode		
Ecc.mapCharToPoint		
Metodes identifikators		
2.3.1.6. ecc.mapCharToPoint		
Metodes apraksts		
Pārvērš doto simbolu punktā, ar dotās atslēgas palīdzību (sk. Pielikums 1. Formulu lapa - Formula Nr.3 - Simbola pārveidošana punktā (mapping))		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
c	Character	simbols, kuru vajag pārvērst par punktu
key	Point	dotā atslēga
Izvada dati		
	Point	rezultējošs punkts

Metode		
Ecc.mapPointToChar		
Metodes identifikators		
2.3.1.7. ecc.mapPointToChar		
Metodes apraksts		
Pārvērš doto punktu simbolā, ar dotās atslēgas palīdzību (sk. Pielikums 1. Formulu lapa - Formula Nr.6 - Punkta pārveidošana simbolā (unmapping))		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
c	Point	punkts, kuru vajag pārvērst par simbolu
key	Point	dotā atslēga
Izvada dati		
	Character	rezultējošs simbols

Metode		
Ecc.rndBigInt		
Metodes identifikators		
2.3.1.8. ecc.rndBigInt		
Metodes apraksts		
Ģenerē nejaušo skaitli (BigInteger tipa) no 0 līdz dotajam maksimumam		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
max	BigInteger	maksimāla vērtība
Izvada dati		
	BigInteger	nejauš skaitlis no 0 līdz maksimālai vērtībai

2.3.2. Point klase

Palīg klase, kas reprezentē punktu plaknē.

Metode
Point
Metodes identifikators
2.3.2.1. point.constructor

Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
x	BigInteger	x koordināta
y	BigInteger	y koordināta

Metode		
Point.compare		
Metodes identifikators		
2.3.2.2. point.compare		
Metodes apraksts		
Salīdzina konstruēto punktu ar doto. Punkti ir vienādi, ja abu koordinātu vērtības sakrīt		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
point	Point	punkts, ar kuru salīdzināt konstruēto
Izvada dati		
	Boolean	

2.3.3. EncodedMsg klase

Palīg klase, kas reprezentē šifrēto tekstu. Šifrēts tests sastāv no diviem sarakstiem (List), kur vienā glabājas visi šifrētā teksta punkti, un otrā glabājas visas atslēgu nobīdes. Svarīgi, lai punkta un tam atbilstošās punkta nobīdes indekss sakrīt.

Metode
EncodedMsg.
Metodes identifikators
2.3.3.1. encodedMsg.constructor
Metodes apraksts
Tiek izveidoti jaunie (tukši) saraksti ar *Point* un *BigInteger* tipiem

Metode
EncodedMsg.add
Metodes identifikators

2.3.3.2. encodedMsg.add		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
c	Point	punkts, kas reprezentē šifrēto simbolu
kt	BigInteger	atslēgu nobīde

2.3.4. Ecccharmap klase

Palīg klase, kas paātrina dešifrēšanas procesu. Klasē tiek saglabāts saraksts ar simboliem un tiem atbilstošo punktu x koordinātas (kas tiek aprēķināti no šifrētā punkta, izmantojot atbilstošu atslēgu ar nobīdi). Ja dešifrēšanas procesā tiek aprēķināts punkts, kura x koordināta jau ir pierakstīta klasē, klase atgriež simbola vērtību un ļauj izlaist daļu no *mapPointToChar* matemātiska procesa.

Metode
Ecccharmap
Metodes identifikators
2.3.4.1. ecccharmap.constructor

Metode		
Ecccharmap.addChar		
Metodes identifikators		
2.3.4.2. ecccharmap.addChar		
Metodes apraksts		
Pievieno simbolu un attiecīgo punktu x koordinātu klases sarakstam		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
c	Character	simbols
x	BigInteger	x koordināta

Metode
Ecccharmap.existsChar
Metodes identifikators
2.3.4.3. ecccharmap.existsChar

Metodes apraksts		
Pārbauda vai klases sarakstā ir dots simbols		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
c	Character	simbols
Izvada dati		
	Boolean	

Metode		
Ecccharmap.hasKey		
Metodes identifikators		
2.3.4.4. ecccharmap.hasKey		
Metodes apraksts		
Pārbauda vai klases sarakstā ir dota x koordinātas vērtība		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
x	BigInteger	x koordināta
Izvada dati		
	Boolean	

Metode		
Ecccharmap.getChar		
Metodes identifikators		
2.3.4.5. ecccharmap.getChar		
Metodes apraksts		
Atgriež no saraksta simbolu, kas ir saistīts ar doto x koordināti (ja tāds eksistē)		
Ievad dati		
Nosaukums	**Datu tips**	**Apraksts**
x	BigInteger	x koordināta
Izvada dati		
	Character	attiecīgs simbols (ja neeksistē tad *null*)

2.4. Prasību trasējamības matrica

Prasību trasējamības matricā tiek paradīts, kā projektējumā tiek nodrošinātas PPS funkcionālās prasības.

PPS Funkcionālās prasības identifikators	**Projektēta metode**
1.3.1.1. setKeyLength	ecc.constructor
1.3.1.2. setPrimeField	
1.3.1.3. setEllipticCurve	
1.3.1.4. pointSum	ecc.pointSum
1.3.1.5 pointMultiply	ecc.pointMultiply
1.3.1.6. mapCharToPoint	ecc.mapCharToPoint
1.3.1.7. encryptMessage	ecc.encryptMessage
1.3.1.8. decryptMessage	ecc.decryptMessage
1.3.1.9. mapPointToChar	ecc.mapPointToChar
1.3.1.10.generateKeyPair	ecc.rndBigInt
	ecc.pointMultiply

3. DATU BĀZES PROJEKTĒJUMS

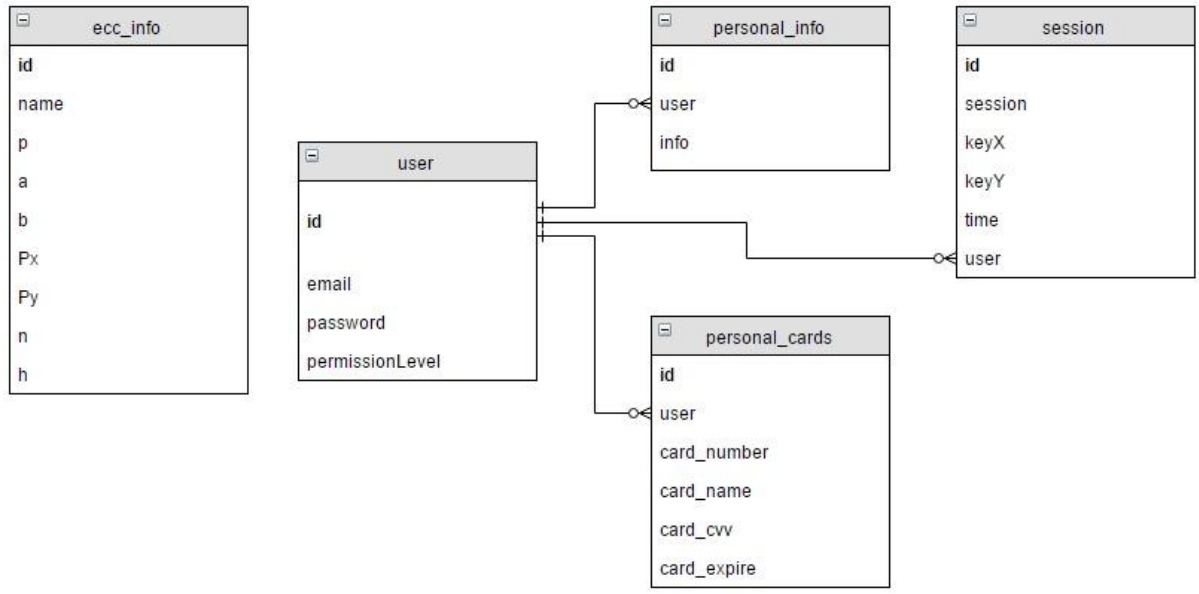
Šajā nodaļā tiek aprakstīts testa lietotnes datu modelis un tā realizācija MySQL datu bāzē.

Datu bāzes tabulas tiek aprakstītas ar sekojošo struktūru:

- Lauka nosaukums
- Lauka datu tips
- Vai lauks var būt NULL
- Citi atribūti
    - PK – Primāra atslēga
    - FK – Ārēja atslēga
    - UQ – Unikāls
    - AI – Automātiski palielinās
- Lauka apraksts

Lai saglabāt FK saistību, ir vajadzīgs lietotājs kas izmantots ka jaunas sesijas lietotājs, kamēr pats lietotājs nepieteiksies.

3.1. Konceptuālais ER modelis



3.2. Tabulu apraksts

3.2.1. Tabula “ecc_info”

Tabulā glabājas eliptiskās līknes parametri, kurus izmanto projekta algoritmi. Tabula ir strukturēta tā, lai pēc nepieciešamības (un projekta papildināšanas) varētu glabāt vairāk par vienu līkni.

Nosaukums	**Datu tips**	**NULL**	**Atribūti**	**Apraksts**
id	Int(11)	NOT NULL	PK, AI	
name	Varchar(45)	NOT NULL	UQ	Līknes nosaukums
p	Varchar(200)	NOT NULL		primārs skaitlis, HEX formātā
a	Varchar(200)	NOT NULL		Līknes parametrs
b	Varchar(200)	NOT NULL		Līknes parametrs
Px	Varchar(200)	NOT NULL		Cikliskas apakšgrupas primitīva elementa x koordināta, HEX formātā
Py	Varchar(200)	NOT NULL		Cikliskas apakšgrupas primitīva elementa y koordināta, HEX formātā
n	Varchar(200)	NOT NULL		Elementu skaits cikliskā grupā
h	Varchar(200)	NOT NULL		Kofaktors

3.2.2. Tabula “user”

Tabulā glabājas lietotāja autentifikācijas un autorizācijas informācija.

Nosaukums	Datu tips	NULL	Atribūti	Apraksts
id	Int(11)	NOT NULL	PK, AI	
email	Varchar(250)	NOT NULL	UQ	Lietotāja e-pasts
password	Longtext	NOT NULL		Lietotāja parole, šifrētā veidā
permissionLevel	Int(11)	NULL		Lietotāja administratora līmenis 0 – parasts lietotājs, >=5 – administrators

3.2.3. Tabula “session”

Tabulā glabājas informācija par lietotāju sesijām un sesijas publiskam atslēgām.

Nosaukums	Datu tips	NULL	Atribūti	Apraksts
id	Int(11)	NOT NULL	PK, AI	
session	Varchar(45)	NOT NULL	UQ	Sesijas identifikators
keyX	Varchar(200)	NOT NULL		Sesijas publiskās atslēgas X koordināta
keyY	Varchar(200)	NOT NULL		Sesijas publiskās atslēgas Y koordināta
time	datetime			Ģenerācijas laiks
user	Int(11)		FK(user.id)	Lietotāja identifikators, kas izmanto sesiju. 1 – ja jauna sesija un lietotājs vēl nav pieteicies

3.2.4. Tabula “personal_info”

Tabulā glabājas lietotāja ievadītā informācija šifrētā veidā.

Nosaukums	Datu tips	NULL	Atribūti	Apraksts
id	Int(11)	NOT NULL	PK, AI	
user	Int(11)	NOT NULL	FK(user.id)	Lietotāja identifikators
info	longtext	NULL		Lietotāja informācija šifrētā veidā

3.2.5. Tabula “personal_cards”

Tabulā glabājas lietotāja ievadītā kredīt karšu informācija šifrētā veidā.

Nosaukums	Datu tips	NULL	Atribūti	Apraksts
id	Int(11)	NOT NULL	PK, AI	
user	Int(11)	NOT NULL	FK(user.id)	Lietotāja identifikators
card_number	longtext	NULL		Lietotāja kartes numurs šifrētā veidā
card_name	longtext	NULL		Lietotāja kartes nosaukums šifrētā veidā
card_cvv	longtext	NULL		Lietotāja kartes CVV šifrētā veidā
card_expire	longtext	NULL		Lietotāja kartes termiņš šifrētā veidā

4. TESTĒŠANAS DOKUMENTĀCIJA

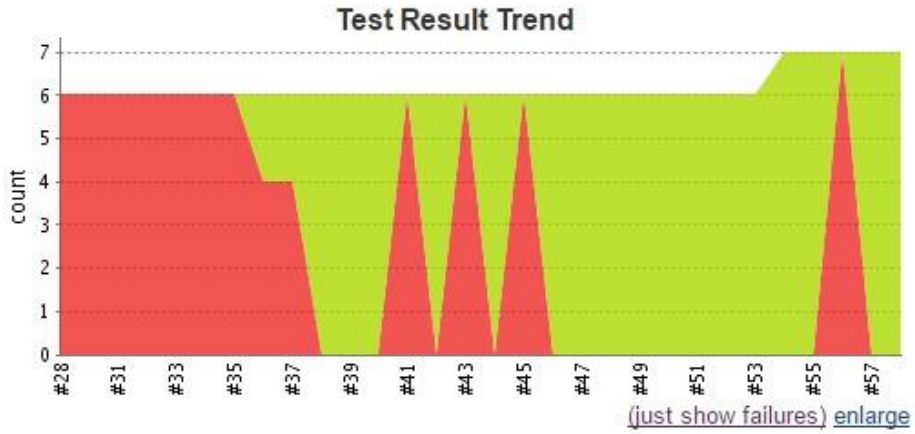
4.1. Automātiskie vienībtesti

Projekta funkcionalitāte tiek testēta pēc katra būvējuma, izmantojot automatizētos testēšanas rīkus. Testēšanai izmantoti divi ietvari : *JUnit* – java funkcionalitātes testēšanai; *QUnit* – JavaScript funkcionalitātes testēšanai.

Ar testiem tika pārbaudītas līknes matemātiskās funkcijas, atslēgu ģenerēšanas un apmaiņas funkcijas un ka dati nemainās pēc šifrēšanas-dešifrēšanas procesa.
Automātiskie testi ļāva uzlabot programmatūras kvalitāti, novērst radušās nepilnības pēc kārtējām izmaiņām.

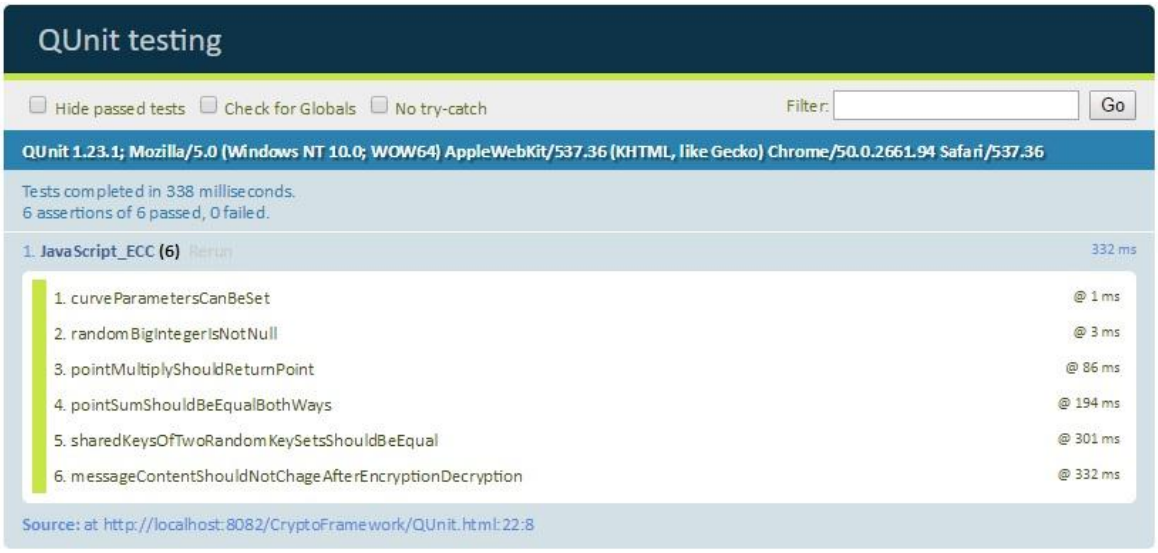
4.2. Vienībtestēšanas rezultāti

Pēc testēšanas *Jenkins* integrācijas rīks attēlo kādi *JUnit* testi bija neveiksmīgi, kā arī attēlo grafiku ar iepriekšējo testu rezultātiem (sk. att. 4.1).



att. 4.1 **JUnit testu rezultātu attēlošana Jenkins rīkā**

QUnit testu rezultāti tiek attēloti atsevišķā projekta tīmekļu lapā (sk. att. 4.2).



att. 4.2 **QUnit testa rezultāti**

4.2.1. Java Ecc

4.2.1.1. curveParametersShouldBeSet

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.1	Tiek uzstādīti visi eliptiskas līknes parametri no datu bāzes	Tiek inicializēta *ecc* klase, kas paņem parametrus no datu bāzes un veiksmīgi atgriež tos.	Atbilst prasībām

4.2.1.2. pointMultiplyShouldReturnPoint

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.2	Tiek ģenerēts jauns līknes punkts no nejauša skaitļa	Rezultējošais punkts nav *NULL*	Atbilst prasībām

4.2.1.3. pointSumShouldBeEqualBothWays

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.3	Tiek ģenerēti divi nejaušie lielie skaitļi. Tiek ģenerēti punkti A un B no šiem	Punkti veiksmīgi ģenerējas un viņu summa nemainās, mainot tos vietām.	Atbilst prasībām

	skaitļiem. Tiek pārbaudīts vai A+B=B+A		

4.2.1.4. sharedKeysOfTwoRandomKeySetsShouldBeEqual

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.4	Tiek ģenerēti divi nevienādi atslēgu pāri (dA, HA) un (dB, HB) un pārbaudīts vai kopējā atslēga tiek pareizi aprēķināta	Kopēja atslēga sakrīt dA*HB = S = dB*HA	Atbilst prasībām

4.2.1.5. messageContentShouldNotChageAfterEncryptionDecryption

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.5	Tiek pārbaudīts, vai testa simbolu virkne nemainās pēc šifrēšanas -dešifrēšanas procesa izmantojot testa atslēgu.	Rezultējošā simbolu virkne sakrīt ar sākotnējo.	Atbilst prasībām

4.2.1.6. messageContentShouldNotChageAfterEncryptionDecryptionASCII255

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.6	Tiek pārbaudīts, vai simbolu virkne no pirmajiem 255 ASCII tabulas simboliem nemainās pēc šifrēšanas - dešifrēšanas procesa izmantojot testa atslēgu.	Rezultējošā simbolu virkne sakrīt ar sākotnējo.	Atbilst prasībām

4.2.1.7. randomMessageContentShouldNotChageAfterEncryptionDecryptionWithRandomKey

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.java.7	Tiek pārbaudīts, vai nejauša simbolu virkne nemainās pēc šifrēšanas - dešifrēšanas procesa izmantojot nejaušo atslēgu.	Rezultējošā simbolu virkne sakrīt ar sākotnējo.	Atbilst prasībām

4.2.2. JavaScript_Ecc

4.2.2.1. curveParametersCanBeSet

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.js.1	Tiek uzstādīti visi eliptiskas līknes parametri un pārbaudīts vai izveidojas klase	Tiek inicializēta *ecc* klase ar testa parametriem.	Atbilst prasībām

4.2.2.2. randomBigIntegereIsNotNull

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.js.2	Tiek ģenerēts jauns nejauš BigInteger klases skaitlis ar uzstādīto maksimumu	Rezultējošais skaitlis nav *NULL*	Atbilst prasībām

4.2.2.3. pointMultiplyShouldReturnPoint

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.js.3	Tiek ģenerēts jauns līknes punkts no nejauša skaitļa	Rezultējošais punkts nav *NULL*	Atbilst prasībām

4.2.2.4. pointSumShouldBeEqualBothWays

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.js.4	Tiek ģenerēti divi nejaušie lielie skaitļi. Tiek ģenerēti punkti A un B no šiem skaitļiem. Tiek pārbaudīts vai A+B=B+A	Punkti veiksmīgi ģenerējas un viņu summa nemainās, mainot tos vietām.	Atbilst prasībām

4.2.2.5. sharedKeysOfTwoRandomKeySetsShouldBeEqual

Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.js.5	Tiek ģenerēti divi nevienādie atslēgu pāri (dA, HA) un (dB, HB) un pārbaudīts, vai kopējā atslēga tiek pareizi aprēķināta	Kopējā atslēga sakrīt dA*HB = S = dB*HA	Atbilst prasībām

4.2.2.6. messageContentShouldNotChageAfterEncryptionDecryption

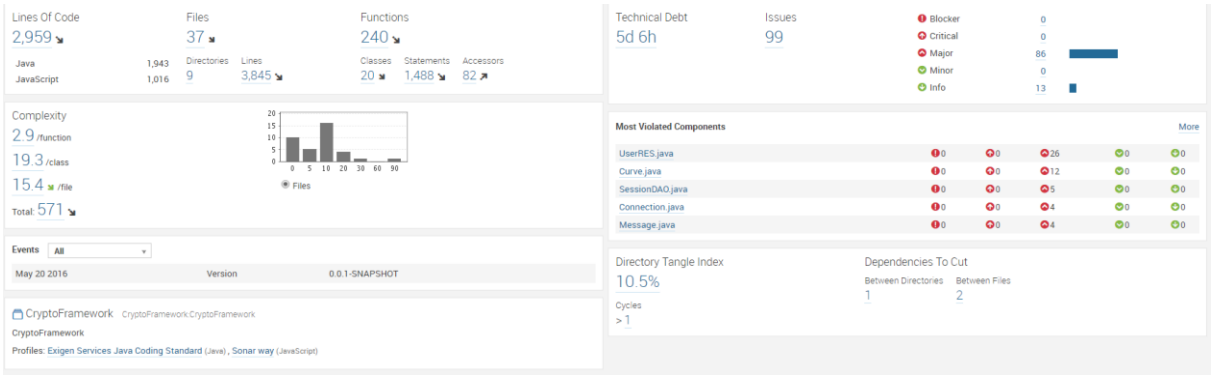
Testa identifikators	Apraksts	Sagaidāmais rezultāts	Rezultāts
test.js.6	Tiek pārbaudīts vai testa simbolu virkne nemainās pēc šifrēšanas - dešifrēšanas procesa izmantojos testa atslēgu.	Rezultējošā simbolu virkne sakrīt ar sākotnējo.	Atbilst prasībām

4.3. Testēšanas trasējamības matrica

Projektēta metode	**Testa identifikators**
ecc.constructor	test.java.1
	test.js.1
ecc.pointSum	test.java.3
	test.java.4
	test.js.4
	test.js.5
ecc.pointMultiply	test.java.2
	test.java.4
	test.js.3
	test.js.5
ecc.mapCharToPoint	test.java.5
ecc.encryptMessage	test.java.6
ecc.decryptMessage	test.java.7
ecc.mapPointToChar	test.js.6
ecc.rndBigInt	test.js.2

4.4. Koda kvalitāte

Koda kvalitātes nodrošināšanai tika izmantots *SonarQube*, kas attēloja sintaktiskas, lasāmības un citas koda kļūdas (sk. att. 4.3).



att. 4.3 **SonarQube**

5. PROJEKTA ORGANIZĀCIJA

Projekta izstrāde tika organizēta pēc iteratīvas pieejas principa. Tika veiktas intensīvas sarunas ar pasūtītāju, lai noskaidrotu nepieciešamo funkcionalitāti, prasības un ierobežojumus. Pabeidzot izvirzīto prasību implementāciju, rezultāts tika iesniegts pasūtītāja novērtēšanai. Pēc tam sekoja jaunas sarunas, lai noskaidrotu tālākās prasības un pārrunātu projekta tālāko attīstību. Bija vairākas iterācijas, tiekoties gan klātienē, gan izmantojot uzņēmuma piedāvāto infrastruktūru (e-pasts, Skype, wiki, JIRA).

Projekta pārvaldībai tika izmantots *SVN* versiju kontrole un *Jenkins* integrācijas rīks. Šāda pieejā ļāva pārbaudīt katru jauno versiju ar automatizētiem testiem un, kļūdainas versijas gadījumā, bez problēmām atgriezt projektu uz pēdējo veiksmīgo versiju un pakāpeniski atrast un novērst kļūdas.

Kļūdu atrašanai tika izmantoti arī automatizētie testēšanas rīki *JUnit* (*Java* vienībtestēšanai) un *QUnit* (*JavaScript* vienībtestēšanai), ar kuriem tika pārbaudīti projekta galvenie algoritmi. Tas ļāva ātrāk atrast kļūdas, kas radās mainot kādu no algoritma daļām.

Programmas kods tika ari pārbaudīts ar *SonarQube,* lai paaugstinātu koda kvalitāti.

Sistēmas programmēšana tika veikta izstrādes vidē *Eclipse Luna Java EE IDE for SAPUI5 Web Developers*. Datu bāzes pārvaldība tika veikta ar *MySQL Workbench 6.3 CE*

Projekta dokumentācijas izveidošanai tika lietots wiki un *Microsoft Word 2016*. Projekta funkciju aprakstiem tiek automātiski ģenerēta *JavaDoc* un *JSDoc* dokumentācija.

6. DARBIETILPĪBAS NOVĒRTĒJUMS

6.1. Sākotnējais darbietilpības novērtējums

Sākotnējais darbietilpības novērtējums bija noteikts balstoties uz klienta un pieredzējušo kolēģu komentāriem, ka arī izstrādātāja pieredzi.

Izstrādātājs uzņēmās izstrādāt nepieciešamo dokumentāciju un prasīto funkcionalitāti triju mēnešu laikā.

6.2. Darbietilpības novērtējums pēc sistēmas nodošanas

Projekta pārvaldība un patērēto dienu uzskaite īstenota izmantojot *AS Exigen Services Latvia* iekšējo rīku *Startrack*. Tiek pieņemts, ka 1 persondiena ir 8 stundas, nedēļā ir 5 darbdienas un mēnesī ir 4 nedēļas, kas ir aptuvens novērtējums, jo darbam veltīto stundu skaits konkrētā dienā var mainīties. Tomēr iegūtie dati ir pietiekoši ticami un nozīmīgi lai tos izmantotu projektu darbietilpības novērtēšanai.

Izpildītais darbs	**Dienu skaits**
Dokumentācijas izstrāde	**20**
Sarunas ar pasūtītāju	5
Prasību specificēšana	6
Projektējuma izveide	6
Vienībtestēšanas rezultātu apspriešana	3
Datubāzes izstrāde	**6**
Sarunas ar pasūtītāju	1
Datubāzes komunikācijas konfigurēšana	2
Tabulu izveide un rediģēšana	3
Java ietvara izstrāde	**18**
Pamata klašu definēšana	5
Eliptiskas līknes funkciju izstrāde un labošana	10
API klašu un funkciju rakstīšana	3
JavaScript lietotnes izveide	**28**
SAPUI5 Ietvara konfigurēšana un pielāgošana	10
Eliptiskas līknes funkciju pielāgošanā *JavaScript* videi	5

Lietotnes projektēšana	3
Lietotāju funkciju izveidošana	8
Dizaina izveide un labošana	2
Vienībtestu izveide un testēšana	**4**
Java vidē	*2,5*
curveParametersShouldBeSet	
pointMultiplyShouldReturnPoint	
pointSumShouldBeEqualBothWays	
sharedKeysOfTwoRandomKey SetsShouldBeEqual	
messageContentShouldNotChageAfter EncryptionDecryption	
messageContentShouldNotChageAfter EncryptionDecryptionASCII255	
randomMessageContentShouldNot ChageAfterEncryptionDecryption WithRandomKey	
JavaScript vidē	*1,5*
curveParametersCanBeSet	
randomBigIntegerIsNotNull	
pointMultiplyShouldReturnPoint	
pointSumShouldBeEqualBothWays	
sharedKeysOfTwoRandomKey SetsShouldBeEqual	
messageContentShouldNotChageAfter EncryptionDecryption	

Kopā 76 persondienas kas ir ~3,8 personmēneši.

NOBEIGUMS

Projekta mērķis tika sasniegts. Pašlaik algoritms var ātri šifrēt un dešifrēt informāciju ja ir atslēga. Ja kāds ļaundaris pārķers informācijas paketi, informācija nebūs atklātā tekstā un tādējādi gandrīz neiespējams atšifrēt informāciju, neatrisinot diskrētā logaritma problēmu eliptiskām līknēm.

Neskatoties uz to, ka galvenais mērķis bija palīdzēt izstrādātajiem aizsargāt informāciju, kas tiek pārsūtīta caur neaizsargātu tīklu, izstrādātājs var arī izmantot šo ietvaru ar aizsargāto *HTTPS* protokolu - tas tikai palielinās drošību.

Vērts ari pieminēt ka, pašreizējais šifrētās informācijas attēlojums (ar redzamiem parametru k, x, y nosaukumiem (sk. att. 2.6)) ir izdarīts, lai atvieglotu demonstrāciju. Realizācijas laikā izstrādātājs var to attēlot citādi kā arī var pielietot kādu testa kompresijas algoritmu, lai samazināt šifrētas informācijas izmēru.

LITERATŪRAS SARAKSTS

1. Krists Magons – “Applications and Benefits of Elliptic Curve Cryptography” – Pieejams: http://ceur-ws.org/Vol-1548/032-Magons.pdf

2. Andrea Corbellini – “Elliptic Curve Cryptography: ECDH and ECDSA” [tiešsaite] – Pieejams: http://andrea.corbellini.name/2015/05/30/elliptic-curve-cryptography-ecdh-and-ecdsa/

3. Wikipedia – “Elliptic Curve Cryptography” [tiešsaite] – Pieejams: https://en.wikipedia.org/wiki/Elliptic_curve_cryptography

4. Stackexchange – “Mapping points between elliptic curves and integers” [tiešsaite] – Pieejams: http://crypto.stackexchange.com/questions/310/mapping-points-between-elliptic-curves-and-the-integers

5. Stackexchange – “Mapping of message onto elliptic curve and reverse it” [tiešsaite] – Pieejams: http://crypto.stackexchange.com/questions/14955/mapping-of-message-onto-elliptic-curve-and-reverse-it

6. Stackexchange – “ElGamal with elliptic curves” [tiešsaite] – Pieejams: http://crypto.stackexchange.com/questions/9987/elgamal-with-elliptic-curves

7. Padma Bh et. al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 05, 2010, 1904-1907 “Encoding And Decoding of a Message in the Implementation of Elliptic Curve Cryptography using Koblitz’s Method” – Pieejams: http://www.enggjournals.com/ijcse/doc/IJCSE10-02-05-08.pdf

8. Kefa Rabah – “Elliptic Curve ElGamal Encryption and Signature Schemes” – Pieejams: https://www.researchgate.net/publication/45949429_Elliptic_Curve_ElGamal_Encryption_and_Signature_Schemes

9. Nicolae Constantinescu – “Elliptic Curve Cryptosystems and Scalar Multiplication” – Pieejams: http://inf.ucv.ro/~ami/index.php/ami/article/viewFile/305/292

10. Mandy Zandra Seet – “ELLIPTIC CURVE CRYPTOGRAPHY Improving the Pollard-Rho Algorithm”, The University of New South Wales, bakalaura darbs 02.11.2007 – Pieejams: https://www.maths.unsw.edu.au/sites/default/files/mandyseetthesis.pdf

11. Come on code on – “Modular Multiplicative Inverse” [tiešsaite] – Pieejams: https://comeoncodeon.wordpress.com/2011/10/09/modular-multiplicative-inverse/

PIELIKUMI

Pielikums 1. Formulu lapa

Apzīmējumi

P	Cikliskas apakšgrupas primitīvais elements - līknes punkts, kas tiek izmantots kā ģenerators cikliskai grupai
p	pirmskaitlis, līknes parametrs
a	līknes funkcijas parametrs
$Q1, Q2$	dotie punkti (pieder Eliptiskās līknes punkta P cikliskajai apakškopai)
C	šifrēts punkts (pieder Eliptiskās līknes punkta P cikliskajai apakškopai)
L	formulas starp skaitlis
n, i	skaitlis $n, i \in \mathbb{N}$
k	nejaušs skaitlis $k \in \mathbb{N}$
m	dotais simbols
M	punkts, kas reprezentē simbolu (pieder, ar primāro skaitli ierobežotai, Eliptiskās līknes plaknei)
S	punkts, kas ir 2 personu kopējā atslēga-noslēpums (pieder Eliptiskās līknes punkta P cikliskajai apakškopai)
T	prefikss, kas nozīmē pagaidu vērtību
x, y	postfikss, kas nozīme koordinātu vērtību
$<<$	bināra nobīde pa kreisi

Formula Nr.1 - Punktu saskaitīšana

ja $Q1 = Q2$

$$L \equiv (3 * {Q2_x}^2 + a) / (2 * Q2_y) \pmod{p}$$

$$TQ_x \equiv L^2 - Q2_x - Q2_x \pmod{p}$$

$$TQ_y \equiv L(Q2_x - TQ_x) - Q2_y \pmod{p}$$

$$Q = (TQ_x ; TQ_y)$$

ja $Q2 \neq Q1$

$$L \equiv (Q1_y - Q2_y) / (Q1_x - Q2_x) \pmod{p}$$

$$TQ_x \equiv L^2 - Q2_x - Q1_x \pmod{p}$$

$$TQ_y \equiv L(Q2_x - TQ_x) - Q2_y \pmod{p}$$

$$Q = (TQ_x ; TQ_y)$$

Formula Nr.2 - Punktu reizinājums

$$Q = Q1$$

n tiek dalīts uz 2 un 1 kas tiek saglabāts stack:

$n\%2$

ja $n\%2 == 0$, stack saglabā 2

ja $n\%2 == 1$, stack saglabā 1

$n = n/2$

atkārtojam kamēr n!=0

izmetam no stack pirmo skaitli, kas vienmēr ir 1, jo 2/2 ir 1
ņemam no stack 2 vai 1:

ja 2 : $Q = Q + Q$

ja 1 : $Q = Q + Q1$ *(ja* Q = Q1, *tad izmanto* Q + Q*)*

atkārtojam, kamēr stack nav tukšs

Formula Nr.3 - Simbola pārveidošana punktā (mapping)

$M \equiv S << m \pmod{p}$

punkts *M* tiek nodots šifrēšanai

Formula Nr.4 - Simbola šifrēšana

simbols m tiek pārveidots punktā *M*

tiek ģenerēts nejaušs punkts no kopējā noslēpuma:

$$TS_x = k * S_x$$

$$TS_y = S_y$$

$$TS = (TS_x ; TS_y)$$

nejaušs punkts tiek saskatīts ar simbola punktu:

$$C = M + TS$$

(C, k) **tiek nosūtīts**

Formula Nr.5 - Punkta atšifrēšana

tiek saņemts (C, k)

tiek ģenerēts punkts no dotā k **un kopēja noslēpuma:**

$$TS_x = k * S_x$$

$$TS_y = S_y$$

$$TS = (TS_x ; TS_y)$$

tiek noskaidrots simbola punkts:

$$M = C - TS$$

Formula Nr.6 - Punkta pārveidošana simbolā (unmapping)

$$i = 0$$

kamēr $TS \neq \boldsymbol{M}$**:**

$$TS \equiv S \ll i \ (mod\ p)$$

$$i++$$

saņemtais i ir simbola vērtība

$$m = (char)i$$

simbols m ir atrasts

Pielikums 2. Java koda gabals

```
/**
     * Performs point multiplication d*point based on elliptic curve math
functions
     * @param pointX x coord of {@link Point} to multiply
     * @param pointY y coord of {@link Point} to multiply
     * @param d {@link BigInteger} to multiply {@link Point} with
     * @param p elliptic curve parameter (prime)
     * @return new {@link Point}
     * @see Point
     * @see #pointSum(Point, Point)
     * */
    public static Point pointMultiply(BigInteger pointX,BigInteger
pointY,BigInteger d,BigInteger p){
        Stack<Integer> st = new Stack<Integer>();
        int t;  // Stack is used to store actions (add or multiply) that
are then performed to required point
        BigInteger resultX=pointX;
        BigInteger resultY=pointY;
        BigInteger tempX;
        BigInteger tempY;
        BigInteger i = d;
        while(i.compareTo(BigInteger.ZERO)>0){
            if((i.mod(TWO)).compareTo(BigInteger.ONE)==0){
                st.push(1);
                i=i.subtract(BigInteger.ONE);
            }
            else {
                st.push(2);
                i=i.divide(TWO);
            }
        }
        // Since 2/2 leaves 1, and it is stored, we need to throw it out!
(otherwise if for ex.  n=2, nP = 2(P+P) )
        if(!st.isEmpty()){
            t = st.pop();
        }   while(!st.isEmpty()){
            t = st.pop();
            if(t==1){
                //Q = Q + P;
                    Point temp = pointSum(resultX,resultY,pointX,pointY);
                    tempX = temp.getX();
                    tempY = temp.getY();

            }
            else{

                //Q = Q + Q;
                    Point temp = pointSum(resultX,resultY,resultX,resultY);
                    tempX = temp.getX();
                    tempY = temp.getY();

            }
            resultX= tempX.mod(p);
            resultY= tempY.mod(p);
        }
        return new Point(resultX,resultY);
         }

/**
     * Performs {@link Point} addition point1+point2 based on elliptic
curve math functions
```

```
     * @param point1X first {@link Point} x coord
     * @param point1Y first {@link Point} y coord
     * @param point2X second {@link Point} x coord
     * @param point2Y second {@link Point} y coord
     * @return new {@link Point}
     * @see Point
     * @see #pointMultiply(Point,BigInteger,BigInteger)
     * */
    public static Point pointSum(BigInteger point1X, BigInteger point1Y,
BigInteger point2X, BigInteger point2Y){
        BigInteger l;
        BigInteger resultX;
        BigInteger resultY;

        if(point1X.compareTo(point2X)!=0){   //if point1X != point2X
                    BigInteger ltemp1 =
point1Y.subtract(point2Y).mod(Ecc.getP());
                    BigInteger ltemp2 =
point1X.subtract(point2X).modInverse(p);
                    l= ltemp1.multiply(ltemp2); // L=(point1Y-
point2Y)/(point1X-point2X)
                }
                else {//if point1X == point2X
                    BigInteger ltemp1 =
point2X.pow(2).multiply(THREE).add(a).mod(p);
                    BigInteger ltemp2 =
point2Y.multiply(TWO).modInverse(p);
                    l= ltemp1.multiply(ltemp2); //
L=(3*point2X^2+a)/(2*point2Y)
                }
                l=l.mod(p);

        resultX = l.pow(2).subtract(point2X).subtract(point1X).mod(p); //
resultX = L^2 - point2X - point1X

        resultY =
l.multiply(point2X.subtract(resultX)).subtract(point2Y).mod(p); // resultY
= L(point2X-resultX)-point2Y

        return new Point(resultX,resultY);
         }
```

Pielikums 3. JavaScript koda gabals

```
/**
* Returns a sum of two points
* @param {BigInteger} Cx x coord of first Point
* @param {BigInteger} Cy y coord of first Point
* @param {BigInteger} Px x coord of second Point
* @param {BigInteger} Py y coord of second Point
* @return {Point} (x,y)
* @function
*/
ecc.prototype.pointSum = function (Cx, Cy, Px, Py){
    var L, TQx, TQy, Ltemp1, Ltemp2,subTemp;
    if(Cx.compareTo(Px)!==0 && Cy.compareTo(Py)!==0){
        /*that is if Q!=P */
                Ltemp1 = Cy.subtract(Py).mod(this.p);
                subTemp = Cx.subtract(Px);
        if(subTemp.signum()<0){
                subTemp = subTemp.negate();
                subTemp = subTemp.mod(this.p);
                subTemp = this.p.subtract(subTemp);
        }
            Ltemp2 = subTemp.modInverse(this.p);
            /*that is  L=(Qy-Py)/(Qx-Px) */
            L= Ltemp1.multiply(Ltemp2);
    }
    else {
        /* if Q==P */
            Ltemp1 =
Px.pow(2).multiply(this.THREE).add(this.a).mod(this.p);
            subTemp = Py.multiply(this.TWO);
            if(subTemp.signum()<0){
                subTemp = subTemp.negate();
                subTemp = subTemp.mod(this.p);
                subTemp = this.p.subtract(subTemp);
            }
            Ltemp2 = subTemp.modInverse(this.p);
            L= Ltemp1.multiply(Ltemp2);
    }
    L=L.mod(this.p);
    /*that is  TQx = L^2 - Px - Qx */
    TQx = L.pow(2).subtract(Px).subtract(Cx).mod(this.p);
    /*that is  TQy = L(Px-TQx)-Py */
    TQy = L.multiply(Px.subtract(TQx)).subtract(Py).mod(this.p);
    return new point(TQx,TQy);
};
/**
 * Performs point multiplication d*P based on elliptic curve math functions
 * @param {BigInteger} Px x coord of Point to multiply
 * @param {BigInteger} Py y coord of Point to multiply
 * @param {BigInteger} d to multiply Point with
 * @return {Point}
 * @function
 * */
ecc.prototype.pointMultiply = function (Px,Py,d){
    // Stack is used to store actions (add or multiply) that are then
performed to required point
    var st = [];
    var t;
    var Qx=Px;
    var Qy=Py;
    var TQx;
    var TQy;
```

```
    var i = d;
    var Temp;
    while(i.compareTo(this.ZERO)>0){
        if((i.mod(this.TWO)).compareTo(this.ONE)===0){
            st.push(1);
            i=i.subtract(this.ONE);
        }
        else {
            st.push(2);
            i=i.divide(this.TWO);
        }
    }
    // Since 2/2 leaves 1, and it is stored, we need to throw it out!
(otherwise if for ex.  n=2, nP = 2(P+P) )
    if(st[0] != null){
        t = st.pop();
    }
    while(st[0] != null){
        t = st.pop();
        if(t===1){
            /*that is Q = Q + P */
                Temp = this.pointSum(Qx,Qy,Px,Py);
                TQx = Temp.Hx;
                TQy = Temp.Hy;
        }
        else{
            /*that is Q = Q + Q */
                Temp = this.pointSum(Qx,Qy,Qx,Qy);
                TQx = Temp.Hx;
                TQy = Temp.Hy;
        }
        Qx= TQx.mod(this.p);
        Qy= TQy.mod(this.p);
    }
    var TQ = new point(Qx,Qy);
    return TQ;
};
```

Pielikums 4. JavaDoc

Class Ecc

java.lang.Object

main.java.ecc.Ecc

```
public class Ecc
extends java.lang.Object
```

ECC class is created for fast and reliable encryption/decryption of information using Elliptic Curve Cryptography methods.

Author:

opesudovs

Field Summary

Fields

Modifier and Type	Field and Description
`private static java.math.BigInteger`	**a** Elliptic curve parameter
`private static java.math.BigInteger`	**b** Elliptic curve parameter
`private static Ecccharmap`	**ECCCHARMAP** Some decrypted characters are stored in **Ecccharmap** to speed up decryption of repeating characters
`private static int`	**h** Elliptic curve parameter
`private static java.math.BigInteger`	**n** Elliptic curve parameter, nP!=0
`private static java.math.BigInteger`	**p** Elliptic curve parameter
`private static java.math.BigInteger`	**Px**

	Elliptic curve parameter, generator point X coordinate
`private static java.math.BigInteger`	**`Py`** Elliptic curve parameter, generator point Y coordinate
`private static java.math.BigInteger`	**`THREE`** Pre-init `BigInteger` with value 3 for later use
`private static java.math.BigInteger`	**`TWO`** Pre-init `BigInteger` with value 2 for later use

Constructor Summary

Constructors

Constructor and Description
`Ecc``(java.lang.Boolean useConnect)` Creates ECC instance and sets ECC parameters from database if useConnect == true
`Ecc``(java.lang.String curve)` Creates ECC instance and sets ECC parameters from database
`Ecc``(java.lang.String sp, java.lang.String sa, java.lang.String sb, java.lang.String sPx, java.lang.String sPy, java.lang.String sn, int sh)` Creates ECC instance and manually sets curve parameters

Method Summary

All Methods

Modifier and Type	Method and Description
`static java.lang.String`	**`decryptMessage`**`(`**`EncodedMsg`**` encMessage, `**`Point`**` key)` Decrypts given encrypted message with given key (`Point`)
`static` **`EncodedMsg`**	**`encryptMessage`**`(java.lang.String message, java.math.BigInteger keyX, java.math.BigInteger keyY)`

	Encrypts given string with given key (**Point**)
static **EncodedMsg**	**encryptMessage**(java.lang.String message, **Point** key) Encrypts given string with given key (**Point**)
static java.math.BigInteger	**getA**()
static java.math.BigInteger	**getB**()
static int	**getH**()
static java.math.BigInteger	**getN**()
static java.math.BigInteger	**getP**()
static java.math.BigInteger	**getPx**()
static java.math.BigInteger	**getPy**()
static **Point**	**mapCharToPoint**(char c, **Point** key) Maps given Character to a **Point** using key (**Point**)
static char	**mapPointToChar**(**Point** charPoint, **Point** key) Maps given **Point** to a Character using key (**Point**)
static **Point**	**pointMultiply**(java.math.BigInteger pointX, java.math.BigInteger pointY, java.math.BigInteger d, java.math.BigInteger p) Performs point multiplication d*point based on elliptic curve math functions
static **Point**	**pointMultiply**(**Point** point, java.math.BigInteger d, java.math.BigInteger p) Performs point multiplication d*point based on elliptic curve math functions
static **Point**	**pointSum**(java.math.BigInteger point1X, java.math.BigInteger point1Y, java.math.BigInteger point2X, java.math.BigInteger point2Y)

	Performs `Point` addition point1+point2 based on elliptic curve math functions
static **Point**	**pointSum**(**Point** point1, **Point** point2) Performs `Point` addition point1+point2 based on elliptic curve math functions
static java.math.BigInteger	**rndBigInt**(java.math.BigInteger max) Generates a random `BigInteger` of same bit length
static void	**setA**(java.math.BigInteger a)
static void	**setB**(java.math.BigInteger b)
static void	**setH**(int h)
static void	**setN**(java.math.BigInteger n)
static void	**setP**(java.math.BigInteger sp)
static void	**setPx**(java.math.BigInteger px)
static void	**setPy**(java.math.BigInteger py)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

a

private static java.math.BigInteger a

Elliptic curve parameter

b

private static java.math.BigInteger b

Elliptic curve parameter

ECCCHARMAP

private static final Ecccharmap ECCCHARMAP

Some decrypted characters are stored in `Ecccharmap` to speed up decryption of repeating characters

h

```
private static int h
```

Elliptic curve parameter

n

```
private static java.math.BigInteger n
```

Elliptic curve parameter, nP!=0

p

```
private static java.math.BigInteger p
```

Elliptic curve parameter

Px

```
private static java.math.BigInteger Px
```

Elliptic curve parameter, generator point X coordinate

Py

```
private static java.math.BigInteger Py
```

Elliptic curve parameter, generator point Y coordinate

THREE

```
private static final java.math.BigInteger THREE
```

Pre-init `BigInteger` with value 3 for later use

TWO

```
private static final java.math.BigInteger TWO
```

Pre-init `BigInteger` with value 2 for later use

Constructor Detail

Ecc

```
public Ecc(java.lang.Boolean useConnect)
```

Creates ECC instance and sets ECC parameters from database if useConnect == true

Parameters:

`useConnect - use database?`

Ecc

```
public Ecc(java.lang.String curve)
```

Creates ECC instance and sets ECC parameters from database

Parameters:

`curve - name of table entry in database`

Ecc

```
public Ecc(java.lang.String sp,
           java.lang.String sa,
           java.lang.String sb,
           java.lang.String sPx,
           java.lang.String sPy,
           java.lang.String sn,
              int sh)
```

Creates ECC instance and manually sets curve parameters

Parameters:

`sp - this.p`

`sa - this.a`

`sb - this.b`

`sPx - this.Px`

`sPy - this.Py`

`sn - this.n`

`sh - this.h`

Method Detail

decryptMessage

```
public static java.lang.String decryptMessage(EncodedMsg encMessage,
                                               Point key)
```

Decrypts given encrypted message with given key (`Point`)

Parameters:

`encMessage - EncodedMsg to be encrypted`

`key - key`

Returns:

`String`

See Also:

`encryptMessage(String,Point)`

encryptMessage

```
public static EncodedMsg encryptMessage(java.lang.String message,
                                        java.math.BigInteger keyX,
                                           java.math.BigInteger keyY)
```

Encrypts given string with given key (`Point`)

Parameters:

`message - String to be encrypted`

`keyX - key x coord`

`keyY - key y coord`

Returns:

`EncodedMsg`

See Also:

`decryptMessage(EncodedMsg,Point)`

encryptMessage

```
public static EncodedMsg encryptMessage(java.lang.String message,
                                        Point key)
```

Encrypts given string with given key (`Point`)

Parameters:

`message - String to be encrypted`

`key - key`

Returns:

`EncodedMsg`

See Also:

`decryptMessage(EncodedMsg,Point)`

getA

```
public static java.math.BigInteger getA()
```

getB

```
public static java.math.BigInteger getB()
```

getH

```
public static int getH()
```

getN

```
public static java.math.BigInteger getN()
```

getP

```
public static java.math.BigInteger getP()
```

getPx

```
public static java.math.BigInteger getPx()
```

getPy

```
public static java.math.BigInteger getPy()
```

mapCharToPoint

```
public static Point mapCharToPoint(char c,
                                   Point key)
```

Maps given `Character` to a `Point` using key (`Point`)

Parameters:

c - Character to be mapped

key - key

Returns:

Point

See Also:

Point, mapPointToChar(Point, Point)

mapPointToChar

```
public static char mapPointToChar(Point charPoint,
                                  Point key)
```

Maps given Point to a Character using key (Point)

Parameters:

charPoint - Point that represents a character

key - key

Returns:

Character

See Also:

Point, mapCharToPoint(char, Point)

pointMultiply

```
public static Point pointMultiply(java.math.BigInteger pointX,
                                  java.math.BigInteger pointY,
                                  java.math.BigInteger d,
                                     java.math.BigInteger p)
```

Performs point multiplication d*point based on elliptic curve math functions

Parameters:

pointX - x coord of Point to multiply

pointY - y coord of Point to multiply

d - BigInteger to multiply Point with

p - elliptic curve parameter (prime)

Returns:

new Point

See Also:

Point, pointSum(Point, Point)

pointMultiply

```
public static Point pointMultiply(Point point,
                                  java.math.BigInteger d,
                                     java.math.BigInteger p)
```

Performs point multiplication d*point based on elliptic curve math functions

Parameters:

point - Point to multiply

d - BigInteger to multiply Point with

p - elliptic curve parameter (prime)

Returns:

new Point

See Also:

Point, pointSum(Point, Point)

pointSum

```
public static Point pointSum(java.math.BigInteger point1X,
                             java.math.BigInteger point1Y,
                             java.math.BigInteger point2X,
                                java.math.BigInteger point2Y)
```

Performs Point addition point1+point2 based on elliptic curve math functions

Parameters:

point1X - first Point x coord

point1Y - first Point y coord

point2X - second Point x coord

point2Y - second Point y coord

Returns:

new Point

See Also:

Point, pointMultiply(Point,BigInteger,BigInteger)

pointSum

```
public static Point pointSum(Point point1,
                             Point point2)
```

Performs Point addition point1+point2 based on elliptic curve math functions

Parameters:

point1 - first Point

point2 - second Point

Returns:

new Point

See Also:

Point, pointMultiply(Point,BigInteger,BigInteger)

rndBigInt

```
public static java.math.BigInteger rndBigInt(java.math.BigInteger max)
```

Generates a random BigInteger of same bit length

Parameters:

max - maximum BigInteger value

Returns:

new BigInteger

setA

```
public static void setA(java.math.BigInteger a)
```

setB

```
public static void setB(java.math.BigInteger b)
```

setH

```
public static void setH(int h)
```

setN

```
public static void setN(java.math.BigInteger n)
```

setP

```
public static void setP(java.math.BigInteger sp)
```

setPx

```
public static void setPx(java.math.BigInteger px)
```

setPy

```
public static void setPy(java.math.BigInteger py)
```

Class Point

java.lang.Object

main.java.ecc.Point

```
public class Point
extends java.lang.Object
```

This class represents a point on 2D plane and is made from two `BigInteger` as x and y coordinates.

See Also:

`Ecc.mapCharToPoint(char, Point)`, `Ecc.mapPointToChar(Point, Point)`

Field Summary

Fields

Modifier and Type	Field and Description
`private java.math.BigInteger`	**x** Point's x coordinate
`private java.math.BigInteger`	**y** Point's y coordinate

Constructor Summary

Constructors

Constructor and Description
Point`()` Creaters new instance of **`Point`**
Point`(java.math.BigInteger d)` Constructs a point on elliptic curve
Point`(java.math.BigInteger x, java.math.BigInteger y)` Constructs a point using given coordinates
Point`(java.lang.String x, java.lang.String y)`

Constructs a point using given coordinates

Method Summary

All Methods

Modifier and Type	Method and Description
`boolean`	**compare** (**Point** `point`) Compares two points
`java.math.BigInteger`	**getX** () Returns x coordinate of this **Point**
`java.math.BigInteger`	**getY** () Returns y coordinate of this **Point**
`void`	**setX** (`java.math.BigInteger x`) Sets x coordinate for this **Point**
`void`	**setY** (`java.math.BigInteger y`) Sets y coordinate for this **Point**

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

x

`private java.math.BigInteger x`

Point's x coordinate

y

`private java.math.BigInteger y`

Point's y coordinate

Constructor Detail

Point

```
public Point()
```

Creaters new instance of `Point`

See Also:

`Point`

Point

```
public Point(java.math.BigInteger d)
```

Constructs a point on elliptic curve

Parameters:

`d` - multiplier for elliptic curve generator point

See Also:

`Point`

Point

```
public Point(java.math.BigInteger x,
             java.math.BigInteger y)
```

Constructs a point using given coordinates

Parameters:

`x` - x coord

`y` - y coord

See Also:

`Point`

Point

```
public Point(java.lang.String x,
             java.lang.String y)
```

Constructs a point using given coordinates

Parameters:

`x` - x coord

`y` - y coord

See Also:

`Point`

Method Detail

compare

```
public boolean compare(Point point)
```

Compares two points

Parameters:

K - Point to compare with

Returns:

boolean

See Also:

Point

getX

```
public java.math.BigInteger getX()
```

Returns x coordinate of this Point

Returns:

x coord

getY

```
public java.math.BigInteger getY()
```

Returns y coordinate of this Point

Returns:

y coord

setX

```
public void setX(java.math.BigInteger x)
```

Sets x coordinate for this Point

Parameters:

x - x coord

setY

```
public void setY(java.math.BigInteger y)
```

Sets y coordinate for this Point

Parameters:

y - y coord

Class EncodedMsg

java.lang.Object

main.java.ecc.EncodedMsg

```
public class EncodedMsg
extends java.lang.Object
```

This class represents encrypted message, that is made from `Point` coordinates and random `BigInteger` (k-value) that was used in encryption process.

To correctly decrypt a message, you must use same key as that was used in encryption process.

See Also:

`Ecc.encryptMessage(String, Point)`, `Ecc.decryptMessage(EncodedMsg, Point)`

Field Summary

Fields

Modifier and Type	Field and Description
`private java.util.List<java.math.BigInteger>`	**`k`** array of random BigInteger
`private java.util.List<Point>`	**`l`** array of Points that are mapped to character

Constructor Summary

Constructors

Constructor and Description
`EncodedMsg``()`
`EncodedMsg``(java.util.List<Message> message)` converts `List` of **`Message`** to this EncodedMsg
`EncodedMsg``(java.util.List<Point> lt, java.util.List<java.math.BigInteger> kt)`

Method Summary

All Methods

Modifier and Type	Method and Description
`void`	**`add`**`(`**`Point`** `c, java.math.BigInteger kt)` Adds encrypted character
`Point`	**`get`**`(int i)` returns certain **`Point`**
`java.util.List<java.math.BigInteger>`	**`getK`**`()`
`java.math.BigInteger`	**`getK`**`(int i)`
`java.util.List<`**`Point`**`>`	**`getL`**`()` Returns `List` of **`Point`** from this instance of EncodedMsg
`void`	**`setK`**`(java.util.List<java.math.BigInteger> k)`
`void`	**`setL`**`(java.util.List<`**`Point`**`> lt)` Sets `List` of **`Point`** for this instance of EncodedMsg
`java.util.List<`**`Message`**`>`	**`sgetMessageList`**`()` convert this EncodedMsg to `List` of **`Message`**
`int`	**`size`**`()`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

k

`private java.util.List<java.math.BigInteger> k`

array of random BigInteger

l

```
private java.util.List<Point> l
```

array of Points that are mapped to character

Constructor Detail

EncodedMsg

```
public EncodedMsg()
```

See Also:

`EncodedMsg`

EncodedMsg

```
public EncodedMsg(java.util.List<Message> message)
```

converts `List` of `Message` to this EncodedMsg

Parameters:

`message` - list of messages

See Also:

`EncodedMsg`

EncodedMsg

```
public EncodedMsg(java.util.List<Point> lt,
                  java.util.List<java.math.BigInteger> kt)
```

Parameters:

`lt` - list of points

`kt` - list of k-value

See Also:

`Ecc.encryptMessage(String, Point)`, `EncodedMsg`

Method Detail

add

```
public void add(Point c,
                java.math.BigInteger kt)
```

Adds encrypted character

Parameters:

`c` - `Point` to add

`kt` - k-value for point C

See Also:

`EncodedMsg`

get

```
public Point get(int i)
```

returns certain `Point`

Parameters:

`i - id of Point to return`

Returns:

getK

```
public java.util.List<java.math.BigInteger> getK()
```

getK

```
public java.math.BigInteger getK(int i)
```

Parameters:

`i - order number`

Returns:

`k-value[i]`

See Also:

`EncodedMsg`

getL

```
public java.util.List<Point> getL()
```

Returns `List` of `Point` from this instance of EncodedMsg

Returns:

`List of Point`

See Also:

`EncodedMsg`

setK

```
public void setK(java.util.List<java.math.BigInteger> k)
```

setL

```
public void setL(java.util.List<Point> lt)
```

Sets `List` of `Point` for this instance of EncodedMsg

Parameters:

`lt - List of Point`

See Also:

`EncodedMsg`

sgetMessageList

```
public java.util.List<Message> sgetMessageList()
```

convert this EncodedMsg to `List` of `Message`

Returns:

List

See Also:

EncodedMsg

size

```
public int size()
```

Returns:

size of the lists

See Also:

EncodedMsg

Class Ecccharmap

java.lang.Object

main.java.ecc.Ecccharmap

```
public class Ecccharmap
extends java.lang.Object
```

This class is used to store 'already calculated' `Character` and their respective `Point` x coord values, for increased performance in case if message contains multiple occurance of single character.

Note : `Point` x coord values are dependant on key that was used in decryption process.

Field Summary

Fields

Modifier and Type	Field and Description
`private java.util.Map<java.lang.Character, java.math.BigInteger>`	**`map`** `Map` that stores characters and x coord that represents this character

Constructor Summary

Constructors

Constructor and Description
`Ecccharmap``()`

Method Summary

All Methods

Modifier and Type	Method and Description
`void`	**`addChar`**`(char c, java.math.BigInteger x)` Adds point to map

void	**clear**() Removes all entries in this map
boolean	**existsChar**(char c) Check if character is already in map
char	**getChar**(java.math.BigInteger x) Retrieves character with given x coord
private char	**getKey**(java.math.BigInteger value)
boolean	**hasKey**(java.math.BigInteger x) Check if x coord is already in map

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

map

```
private java.util.Map<java.lang.Character,java.math.BigInteger> map
```

Map that stores characters and x coord that represents this character

Constructor Detail

Ecccharmap

```
public Ecccharmap()
```

See Also:

Ecccharmap

Method Detail

addChar

```
public void addChar(char c,
                    java.math.BigInteger x)
```

Adds point to map

Parameters:

c - character

x - x coord that represents character c

clear

```
public void clear()
```

Removes all entries in this map

existsChar

```
public boolean existsChar(char c)
```

Check if character is already in map

Parameters:

`c` - character

Returns:

boolean

getChar

```
public char getChar(java.math.BigInteger x)
```

Retrieves character with given x coord

Parameters:

`x` - x coord that represents character c

Returns:

character

getKey

```
private char getKey(java.math.BigInteger value)
```

See Also:

`getChar(BigInteger)`

hasKey

```
public boolean hasKey(java.math.BigInteger x)
```

Check if x coord is already in map

Parameters:

`x` - x coord that represents character c

Returns:

boolean

Class ecc

Defined in: ecc.js.

Class Summary

	ecc(p, a, b, Px, Py, n, h) ECC class is created for fast and reliable encryption/decryption of information using Elliptic Curve Cryptography methods.

Method Summary

	decryptMessage(L, S) Decrypts given encrypted message with given key (Point)
	encryptMessage(M, S) Encrypts given string with given key (Point)
	mapCharToPoint(m, S) Maps given Character to a Point using key (Point)
	mapPointToChar(M, S) Maps given Point to a Character using key (Point)
	pointMultiply(Px, Py, d) Performs point multiplication d*P based on elliptic curve math functions
	pointSum(Cx, Cy, Px, Py) Returns a sum of two points
	rndBigInt(max) Generates a random BigInteger of about same length

Class Detail

ecc(p, a, b, Px, Py, n, h)
ECC class is created for fast and reliable encryption/decryption of information using Elliptic Curve Cryptography methods.
Author: opesudovs.

Parameters:

p
a
b
Px
Py
n
h

Method Detail

{String} **decryptMessage**(L, S)
Decrypts given encrypted message with given key (Point)

Parameters:

{EncodedMsg} **L**
to be encrypted
{Point} **S**
key
Returns:
{String}

{EncodedMsg} **encryptMessage**(M, S)
Encrypts given string with given key (Point)
Parameters:
{String} **M**
to be encrypted
{Point} **S**
key
Returns:
{EncodedMsg}

{Point} **mapCharToPoint**(m, S)
Maps given Character to a Point using key (Point)
Parameters:
{Character} **m**
to be mapped
{Point} **S**
key
Returns:
{Point}

{Character} **mapPointToChar**(M, S)
Maps given Point to a Character using key (Point)
Parameters:
{Point} **M**
that represents a character
{Point} **S**
key
Returns:
{Character}

{Point} **pointMultiply**(Px, Py, d)
Performs point multiplication d*P based on elliptic curve math functions
Parameters:
{BigInteger} **Px**
x coord of Point to multiply
{BigInteger} **Py**
y coord of Point to multiply
{BigInteger} **d**
to multiply Point with
Returns:
{Point}

{Point} **pointSum**(Cx, Cy, Px, Py)
Returns a sum of two points
Parameters:
{BigInteger} **Cx**
x coord of first Point
{BigInteger} **Cy**
y coord of first Point
{BigInteger} **Px**
x coord of second Point
{BigInteger} **Py**
y coord of second Point
Returns:
{Point} (x,y)

{BigInteger} **rndBigInt**(max)

Generates a random BigInteger of about same length

Parameters:

{BigInteger} **max**

maximum BigInteger value

Returns:

{BigInteger}

Kvalifikācijas darbs „***Eliptiskās līknes bāzēta kriptogrāfiska ietvara izstrāde***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Oļegs, Pešudovs*** _____ **___.05.2016**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs/a: ***dipl.mat. Imants Pops*** _____ **___.05.2016.**

Recenzents: ***M.soc.zin. Lauris Raipulis***

Darbs iesniegts 30.05.2016.
Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2016. prot. Nr. _____

Komisijas sekretārs(-e): _____