УДК 004.43

JAVA ПЛАТФОРМАСЫНДА КРИПТОГРАФИЯЛЫҚ ҚЫЗМЕТТЕРДІ ЖЕТКІЗУШІ ПРОВАЙДЕРЛЕРМЕН ЖҰМЫС

Құлмахан Дархан Әбділғазизұлы
darkhan_kulmakhan_95@mail.ru
7М06111- «Ақпараттық қауіпсіздіктің әдістері мен технологиялары»
мамандығының 2-курс магистранты
Л. Н. Гумилев атындағы Еуразия Ұлттық Университеті, Нұр-Сұлтан қаласы, Қазақстан.
Ғылыми жетекші – Ж.С. Сауханова

Java платформасы қауіпсіздікке, соның ішінде тіл қауіпсіздігі, криптография, ашық кілттердің инфрақұрылымы, аутентификация, қауіпсіз байланыс және қол жеткізуді басқаруға үлкен мән береді.

JCA (Java cryptography architecture) - бұл Java платформасы үшін криптографиялық функционалдылыққа қол жеткізуге және дамытуға мүмкіндік беретін фреймворк. JCA құрамына хабарлама дайджесттері (хэштер), сандық қолтаңбаларға арналған интерфейс жиынтығы, сертификаттар мен куәліктерді тексеру, шифрлау (симметриялық/ асимметриялық блок/ағындық шифрлар), кілттерді құру және басқару, кездейсоқ сандарды генерациялау криптографиялық қызметтері кіреді. Java платформасының криптографиялық архитектурасы келесі принциптерге негізделген [1]:

- Криптографиялық қызметтерді көрсетудің тәуелсіздігі: қосымшаларға қауіпсіздік алгоритмдерін енгізу қажет емес. Керісінше, олар Java платформасынан қауіпсіздік қызметін сұрай алады. Қауіпсіздік қызметтері Java платформасына стандартты интерфейс арқылы қосылатын провайдерлерде - криптографиялық қызмет жеткізушілер көмегімен - жүзеге асырылады. Бағдарлама қауіпсіздікті қамтамасыз ету мақсатынды бірнеше тарапқа – провайдерлерге сенім арта алады.
- Іске асырудың өзара үйлесімділігі: провайдерлер қосымшалар кодында өзара әрекеттесе алады. Атап айтқанда, бағдарлама белгілі бір провайдермен байланыстырылмаған, ал провайдер белгілі бір қосымшамен байланыстырылмаған. Провайдерлерді таңдау қосымша құрушының құзырында.
- Қауіпсіздікті қамтамасыз етуде архитектураның жаңа алгоритмдерімен кеңею мүмкіндігі: Java платформасына қазіргі уақытта кеңінен қолданылатын қауіпсіздік қызметтерінің негізгі жиынтығын іске асыратын бірқатар провайдерлер кірістірілген. Алайда, кейбір қосымшалар жаңа стандарттарға немесе өз қызметтеріне сенуі мүмкін. Java платформасы осындай қызметтерді жүзеге асыратын арнайы провайдерлерді орнатуды қолдайды.

Архитектураның іске асырудан және алгоритмнен тәуелсіздігі бірін-бірі толықтырып отырады. Мысалы, цифрлық қолтаңбалар және хабарламалар дайджесттері сияқты криптографиялық қызметтерді іске асырудың егжей-тегжейін немесе осы тұжырымдамалардың негізін құрайтын алгоритмдер туралы алаңдамай-ақ пайдалануға болады. Алгоритмнен толық тәуелсіз болу мүмкін болмағанымен, Java платформасының криптографиялық архитектурасы қосымша әзірлеушіні стандартталған, алгоритмге негізделген API-мен қамтамасыз етеді. Іске асыруда тәуелсіздік қажет емес кезде, JCA әзірлеушілерге нақты орындалуды көрсетуге мүмкіндік береді.

Алгоритмнің тәуелсіздігіне криптографиялық «қозғалтқыштардың» (қызметтердің) түрлерін және осы криптографиялық «қозғалтқыштардың» функционалдығын қамтамасыз ететін кластарды анықтау арқылы қол жеткізіледі. Бұл «қозғалтқыш» кластары мысалдары: MessageDigest, Signature, KeyFactory, KeyPairGenerator және шифр кластары.

Іске асыру тәуелсіздігі провайдерлерге негізделген архитектураның көмегімен жүзеге асырылады. «Провайдер» ұғымы «криптографиялық қызметтерді жеткізуші» (Cryptography Service Provider, CSP) ұғымын білдіреді. Бұл ұғым цифрлық қолтаңба алгоритмдері, хабарлама дайджесттері сияқты бір немесе бірнеше криптографиялық қызметтерді жүзеге асыратын пакетке немесе жиынтыққа қатысты айтылады. Бағдарлама белгілі бір қызметті (мысалы, Digital Signature Algorithm – DSA, цифрлық қол қою алгоритмі) жүзеге асыратын және орнатылған провайдерлердің біреуінен объектінің белгілі бір түрін мысалы, Signature класының объектісін, құра алады. Қажет болса, бағдарлама орнына белгілі бір провайдерден орындалуын талап етуі мүмкін. Провайдерлер бағдарлама үшін ашық түрде жаңартылуы мүмкін, мысалы, жылдам немесе қауіпсіз нұсқалары болған кезде.

Іске асырудың үйлесімділігі дегеніміз ол - қосымшада әртүрлі провайдерлер бір-бірімен жұмыс істей алатындығы, бір-бірінің кілттерін немесе бір-бірінің цифрлық қолдарын

тексеруге мүмкіндіктерінің бар болатындығы. Бұл, мысалы, әртүрлі провайдерлер ұсынған алгоритмдер үшін бір провайдер жасаған кілтті екіншісі қолданады және бір провайдер жасаған қолтаңбаны екіншісі тексереді.

Архитектураның жаңа алгоритмдермен кеңеюі дегеніміз қолдау көрсетілетін «қозғалтқыш» кластарының жаңа алгоритмдерді оңай қосуға мүмкіндік беретіндігін білдіреді.

java.security.Provider класы Java платформасында қауіпсіздік провайдері тұжырымдамасын қамтиды. Әзірлеуші өзінің жобасында бір уақытта бірнеше провайдерді қолдана алады. Қауіпсіздік қызметі сұралғанда, java.security файлында анықталған провайдерлер тізімінен басымдығы жоғары провайдер таңдалады. Қосымша әзірлеушісі провайдерден қауіпсіздік қызметін сұрау үшін сәйкес getInstance() әдісін қолданады.

Келесі мысалда кірістірілген провайдерлер тізімін алу программасы келтірілген:

```
import java.security.Provider;
import java.security.Security;
public class MainProv {
        public static void main(String[] args) {
                Provider[] providers = Security.getProviders();
          for (Provider p : providers) {
            System.out.println(p.getName());
          }
        }
}
```

Архитектураға жаңа провайдерді кірістіруді криптографиялық мүмкіндіктері орасан зор Bouncy Castle кітапханасын қосу мысалында көрсетуге болады [2]:

- алдымен http://www.bouncycastle.org/latest_releases.html сілтемесі бойынша қажет JAR-файлды жүктеліп, jre/lib/ext каталогына орнатылады;
- jre/lib/security каталогында орналасқан java.security файлын редакциялау үшін ашылады, 1-суретте java.security файлынан үзінді келтірілген;
- 1-суретке сәйкес провайдерлер тізіміне Bouncy Castle кітіпханасы келесі жолды теру арқылы қосылады:

```
security.provider.11=
          org.bouncycastle.jce.provider.BouncyCastleProvider
```

```
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=apple.security.AppleProvider
```

1-сурет. java.security файлынан үзінді

Bouncy Castle провайдерінің дұрыс орнатылғандығын тексеру үшін келесі программаны жүргізуге болады:

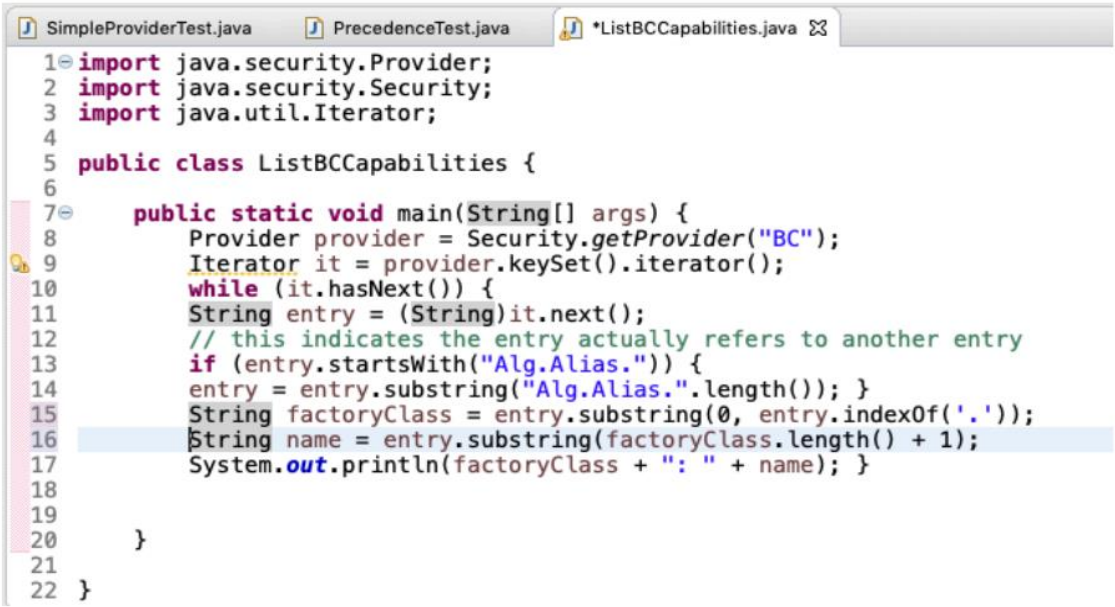
```
import java.security.Security;
public class SimpleProviderTest {
        public static void main(String[] args) {
                String providerName=”BC”;
                if (Security.getProvider(providerName)==null){
```

```
            System.out.println (providerName+" provider not installed")}
            else ){
            System.out.println (providerName+" provider installed")}
        }
}
```

Java орындау ортасы сұрату бойынша сәйкес қызметті көрсете алатын java.security файлындағы бірінші криптографиялық қызметтерді жеткізушіні таңдайды, төмендегі 2-суретте көрсетілген кодтың орындалу нәтижесінен көруге болады [3]:

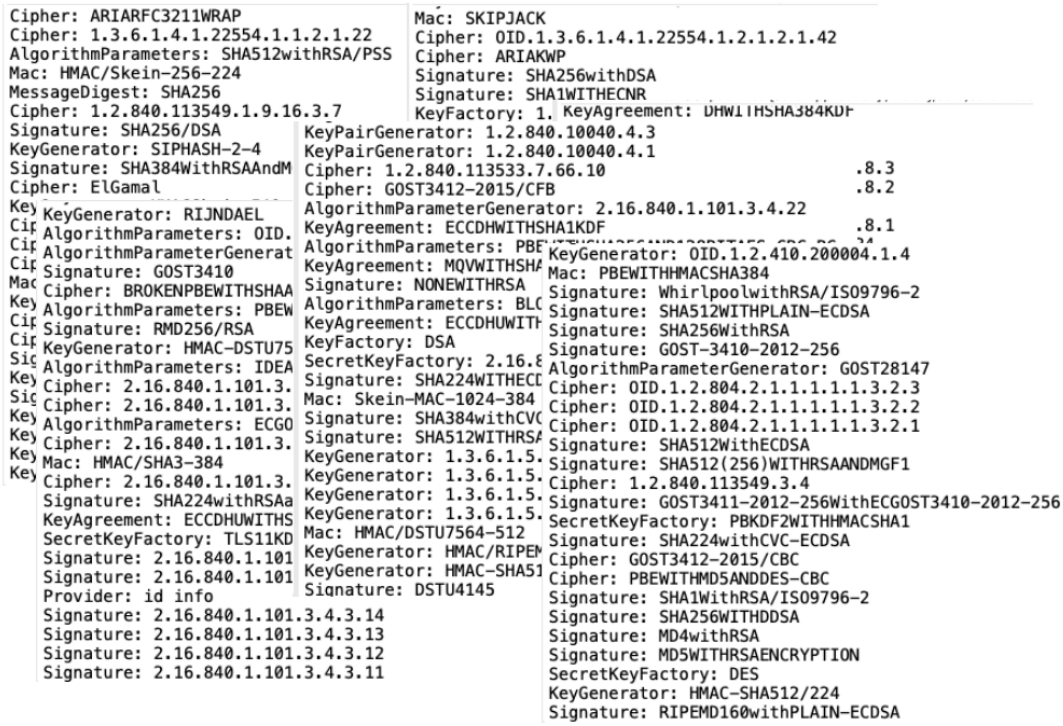
2-сурет. Криптографиялық қызметтерді жеткізушіні таңдау реті

Bouncy Castle провайдерінің мүмкіндіктерін, яғни жүзеге асыра алатын алгоритмдер тізімін шығару үшін 3-суретте көрсетілген программаны орындауға болады:



3-сурет Bouncy Castle провайдерінің мүмкіндіктерін баспаға шығару программасы

Программаны орындау барысында көлемі үлкен тізім аламыз. 4-суретте Bouncy Castle провайдерінің мүмкіндіктерін келтірілген.



4-суретте Bouncy Castle провайдерінің мүмкіндіктері

Мақалада Java платформасында криптографиялық қызметтерді көрсетуді жеткізуші провайдерлермен жұмыс қарастырылды. Архитектураға жаңа провайдерді кірістіру, қосымшада провайдерлерді таңдау реті, провайдерден қауіпсіздік қызметін сұрату үшін сәйкес әдістерді шақыру, провайдердің мүмкіндіктерін анықтау қарапайым кодтары келтірілді. Келтірілген кодтар Java платформасының қауіпсіздікті қамтамасыз ету технологияларын қолдану барысында провайдерлермен жұмыс істеу дағдысын қалыптастыруда маңызды орын алады.

Қолданылған әдебиеттер

1. Abhay Bhargav, B.V. Kumar, Secure Java For Web Application Development. Taylor & Francis Group. 2011
2. David Hook, Beginning Cryptography in Java. John Wiley & Sons. 2005
3. https://www.programcreek.com/java-api-examples/?class=java.security.Security&method=getProvider
4. https://docs.oracle.com/en/java/javase/11/security/java-security-overview1.html#GUID-2EF91196-D468-4D0F-8FDC-DA2BEA165D10