



UNIVERZITET U NOVOM SADU
**FAKULTET TEHNIČKIH NAUKA U
NOVOM SADU**



Milan Kostić

Stefan Karać

Milan Mutić

Nemanja Rogić

Sigurnost i bezbednost u smart grid sistemima

Public key infrastructure system

-Predmetni projekat-

Profesor: Docent dr. Imre Lendak

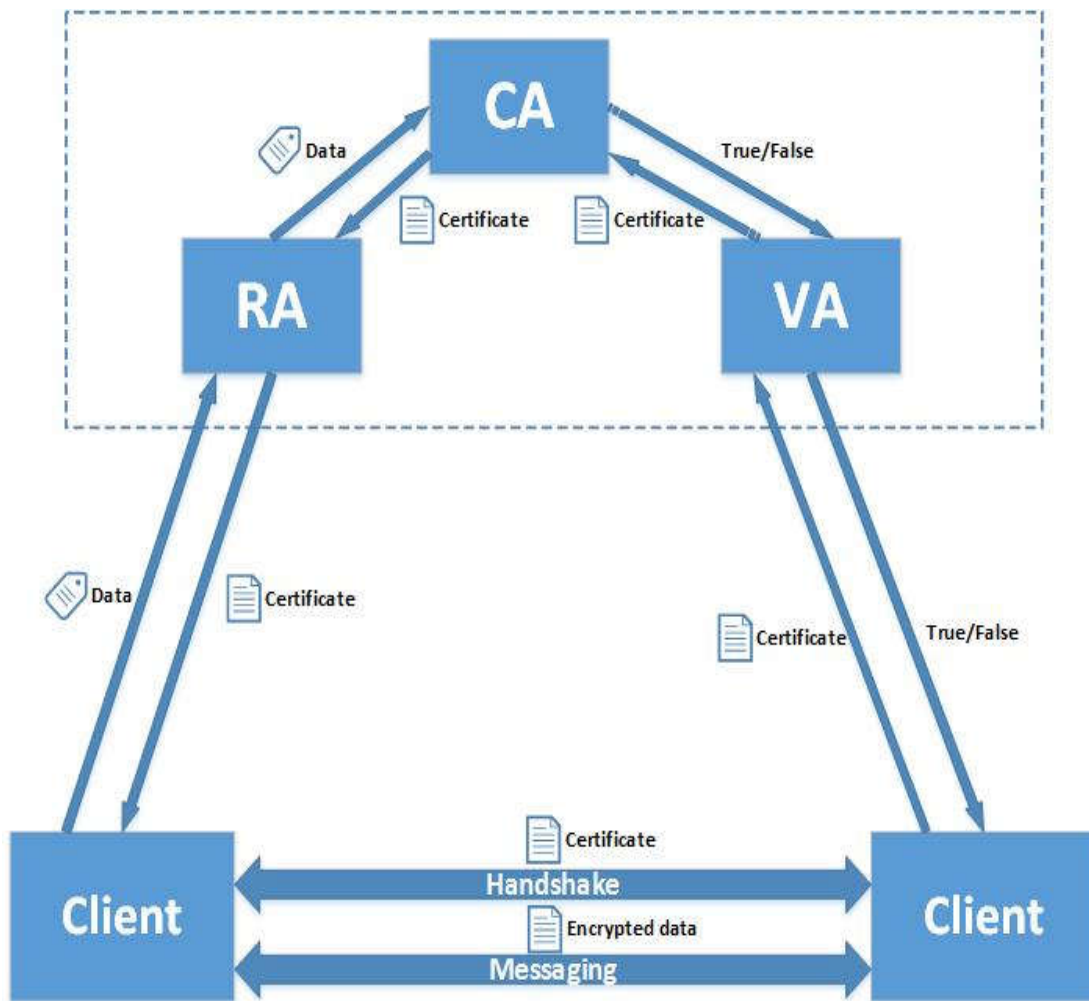
Asistent: Stefan Dejanović

Novi Sad, 2016

1. Arhitektura sistema

Tema rada je implementacija sistema (eng. Public key infrastructure system) za upravljanje sertifikatima. Sistem treba da omogući kreiranje sertifikata, izdavanje sertifikata klijentima (registracija klijenata) kao i validaciju sertifikata.

Sistem se sastoji iz tri komponente (Slika 1): Certification authority (u daljem tekstu CA), Registration authority (u daljem tekstu RA) i Validation authority (u daljem tekstu VA).



Slika 1. Public key infrastructure system

CA ima sledeće zadatke:

- Generisanje i izdavanje sertifikata
- Provera validnosti sertifikata
- Povlačenje sertifikata

CA sve izgenerisane sertifikate čuva trajno na masovnoj i u operativnoj memoriji. U slučaju da CA sistem postane nedostupan, postoji rezervni sistem sa istom funkcionalnošću koji se u najkraćem roku aktivira i stavlja na korišćenje. Svi podaci se zbog toga repliciraju sa aktivnog sistema na rezervni. CA određene akcije evidentira u Windows event log-u.

RA komponenta ima posrednu ulogu prilikom registracije klijenata. Vršiti početne provere klijenata kao i delegiranje zahteva ka CA komponenti, koja treba da izvrši registraciju klijenata.

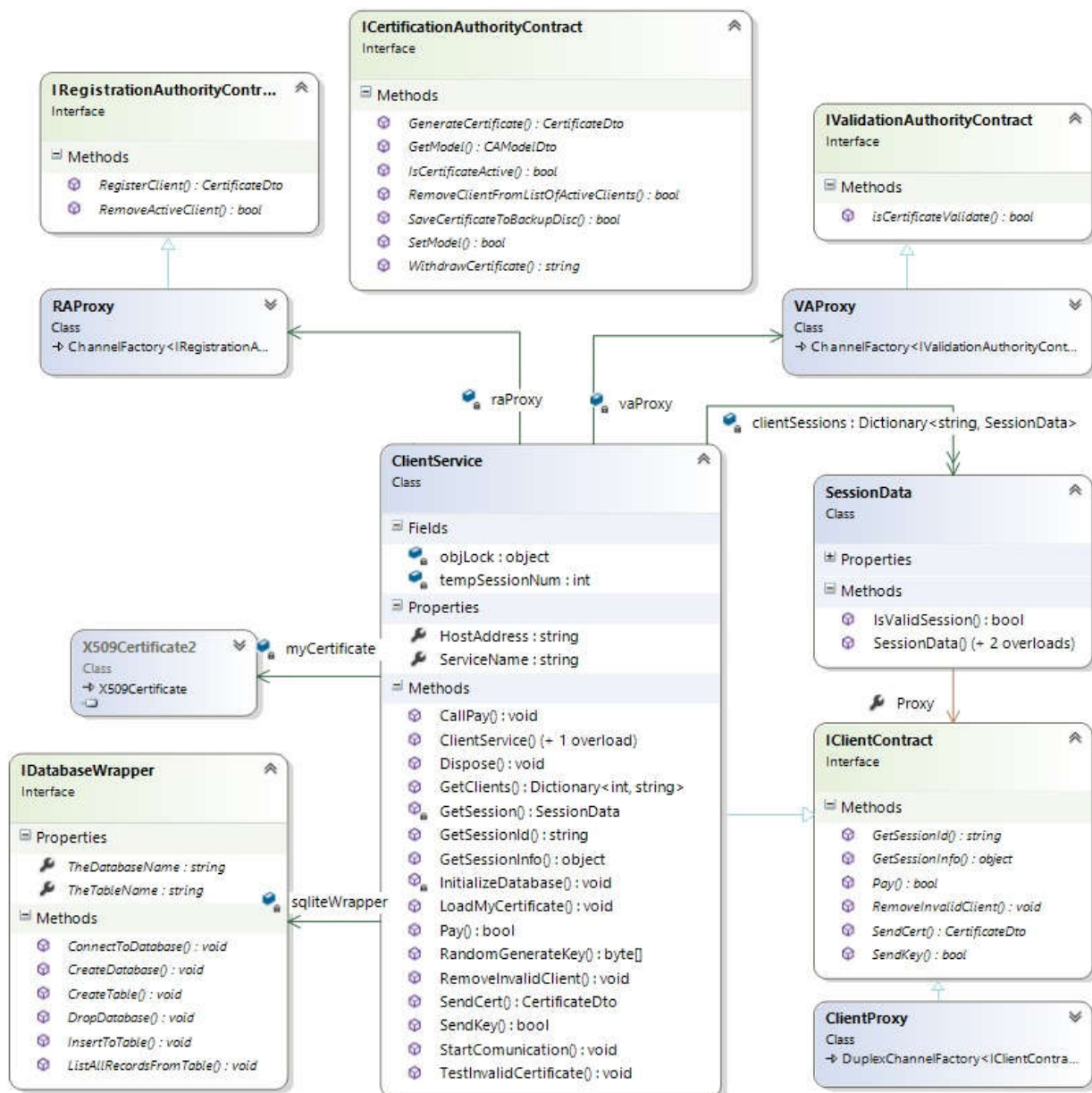
VA podsistem ima ulogu validacije klijenata, koji žele da uspostave međusobnu komunikaciju. Kao i RA komponenta VA prosleđuje klijentske zahteve ka CA podsistemu koji vrši proveru i validiranje klijenata.

Da bi klijenti uspešno komunicirali potrebno je da prvo budu registrovani. Za registraciju se obraćaju RA komponenti. Prilikom uspostave komunikacije vrše obostranu validaciju sertifikata, tako što se obraćaju VA komponenti sa zahtevom validiranja sertifikata druge strane. Nakon uspešne validacije, klijenti razmenjuju ključ sesije koji će se koristiti za kriptovanje i dekriptovanje poruka u toj sesiji. Ključ sesije se razmenjuje asimetričnim algoritmom, dok se dalja komunikacija obavlja kriptovanjem poruka simetričnim algoritmom. Za kriptovanje poruka nakon bezbednog otvaranja komunikacionog kanala se koristi AES (Advanced encryption standard) algoritam u ECB obliku. Za razmenu ključa sesije se koristi asimetrični RSA algoritam.

Svaku uspešnu uspostavu komunikacije klijenti beleže u SQLite bazi podataka. Prilikom toga se upisuju podaci o vremenu i identifikaciji klijenta sa kojim je uspostavljena komunikacija.

1. Dizajn sistema

Slika 2 prikazuje ključne klase koje implementiraju opisane funkcionalnosti u prethodnom poglavlju. Svaka od pomenutih komponenti su nezavisne (CA, RA, VA, klijentska komponenta) aplikacije, odnosno WCF servisi.



Slika 2. Class diagram

CA komponenta nudi `ICertificationAuthorityContract` interfejs. Najznačajnije metode ovog servisa su metode za generisanje sertifikata i proveru važenja sertifikata (Listing 1). Kao što se sa dijagrama zaključuje ove metode pozivaju RA servis i VA servis na zahtev klijenata za registraciju, odnosno validaciju sertifikata.

```

[ServiceContract]
public interface ICertificationAuthorityContract
{
    [OperationContract]
    CertificateDto GenerateCertificate(string subject, string address);

    [OperationContract]
    bool IsCertificateActive(X509Certificate2 certificate);

    .....
}

```

Listing 1. Interfejs CA komponente

RA klijentima stavlja na raspolaganje IRegistrationAuthorityContract interfejs. Na listingu 2 je prikazana jedina značajna metoda RA servisa koja služi za registrovanje klijenata.

```

[ServiceContract]
public interface IRegistrationAuthorityContract
{
    [OperationContract]
    CertificateDto RegisterClient(string address);
}

```

Listing 2. Interfejs RA komponente

Za proveru validnosti sertifikata koristi se VA servis, kroz interfejs IValidationAuthorityContract (Listing 3). Za validaciju sertifikata klijenti će pozvati IsCertificateValid() metodu.

```

[ServiceContract]
public interface IValidationAuthorityContract
{
    [OperationContract]
    bool IsCertificateValid(X509Certificate2 certificate);
}

```

Listing 3. Interfejs VA komponente

Svaka klijentska aplikacija može da se ponaša i kao klijent i kao servis u komunikaciji sa drugim klijentima. Za potrebe komunikacije klijent nudi IClientContract interfejs (Listing 4).

```

[ServiceContract(CallbackContract = typeof(IClientContract), SessionMode =
    SessionMode.Required)]
public interface IClientContract
{
    [OperationContract]
    bool Pay(byte[] message);

    [OperationContract]
    CertificateDto SendCert(CertificateDto cert);
}

```

```

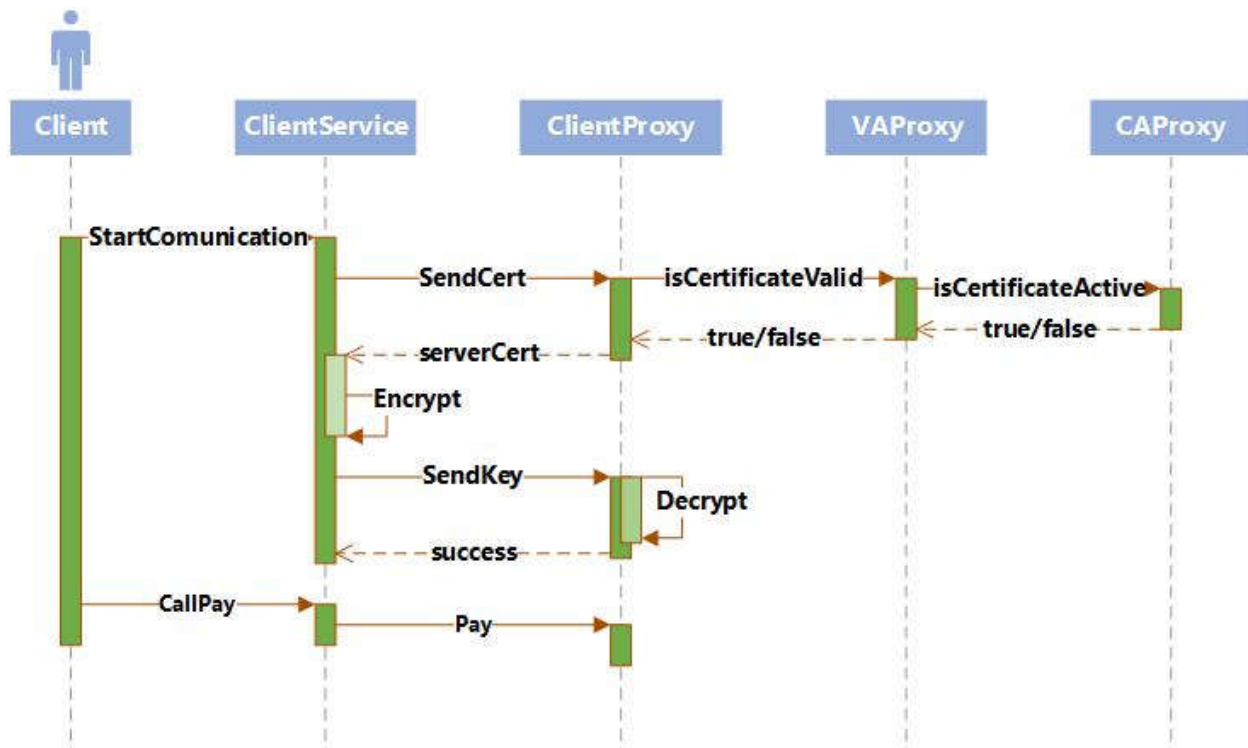
[OperationContract]
bool SendKey(byte[] key);

.....
}

```

Listing 4. Interfejs klijentskog servisa

Tok uspostavljanja komunikacije između klijenata prikazuje dijagram na slici 3.



Slika 3. Uspostava komunikacionog kanala

Klijent inicira komunikaciju ka drugom klijentu tako što mu pošalje svoj sertifikat sa javnim ključem. Drugi klijent potom izvrši validaciju sertifikata tako što se obrati VA servisu. Ukoliko je sertifikat validan, klijent će vratiti svoj javni ključ, koji će takođe biti validiran kod VA servisa. Tim javnim ključem će prvi klijent kriptovati ključ sesije, a drugi klijent će isti dekriptovati svojim privatnim ključem. Nakon toga je uspostavljena komunikacija, pa će se sve ostale poruke kriptovati AES algoritmom.

AES algoritam je simetrični algoritam, koji kriptuje podatke podelom na blokove koji mogu biti različite dužine. U ovom ovom radu je implementiran u ECB obliku, što znači da pored ključa ne koristi dodatne podatke za kriptovanje. Ključ je dužine 128 bita, a i svaki blok je isto dužine 128 bita. Zbog boljih performansi, algoritam je paralelizovan tako da paralelno vrši kriptovanje/dekriptovanje blokova.

Veličina poruke (B)	Broj iteracija	Vreme - paralelno	Vreme - sekvencijalno
1000 B	100	2837 ms	1727 ms
5000 B	100	5800 ms	7331 ms
10000 B	100	9926 ms	14421 ms
30000 B	100	26795 ms	46863 ms

Tabela 1. Poređenje paralelne i sekvencijalne varijante AES algoritma

Rezultati testiranja su prikazani u tabeli 1. Testovi su vršeni na procesoru Intel Pentium 2 x 2.20 GHz. Iz ovih nekoliko testova se zaključuje da veličina poruke direktno utiče na brzinu izvršavanja algoritma, kada je paralelna obrada u pitanju. Kada je poruka prilično mala, u ovom slučaju samo 1 KB, paralelna obrada biva čak sporija u odnosu na sekvencijalnu. Kako veličina poruke raste, tako paralelna obrada postaje sve brža.

Za slučaj da se paralelizuju još neke operacije kao što je na primer dodavanje ključa runde, tada performanse značajno padaju. Na primer kod primene paralelizacije kod dodavanja ključa runde i poruka veličine 1 KB i 100 iteracija, paralelna obrada traje 15385 ms, a sekvencijalna slično tome 15308. To je velika razlika u odnosu na prvi test primer gde je paralelna obrada trajala 2837 ms a sekvencijalna 1727 ms.

2. Testiranje

2.1 Analiza enkriptovanja uz pomoć Wireshark programa

U okviru aplikacije enkripcija podataka se radila prilikom međusobne komunikacije klijenata. Kako bi utvrdili da su poruke koje se između njih razmenjuju zaista zaštićene upotrebili smo Wireshark program. U prvom delu testiranja Binding.Security.Mode je podešen na Transport, i na slici 4 se može videti kako izgleda razmenjena poruka(sa i bez enkripcije). U svim test slučajevima je sa jednog na drugi klijent slata poruka „pozdrav“. Primećuje se da je u oba slučaja za Transport mod gotovo nemoguće izvući neku smisleniju poruku.

```

=====
Bez enkripcije(Security = Transport)
=====
0000  93 00 00 00 01 00 00 00 4d 8c 03 3a e4 4e 0a 53  .....M...N.S
0010  05 00 00 00 da ef 5e 2d d9 2b 63 3c f1 86 b3 1e  .....^-.+c<....
0020  27 ad 88 3d 98 b1 1f cc 36 ed 33 36 89 fb a3 a6  '..=....6.36....
0030  15 d3 6a 87 8e e4 05 26 ee 6e 2e 8a 0f c4 9b 35  ..j....&.n....5
0040  49 93 b2 b3 66 65 f0 f5 53 12 a3 b8 87 9c 7a 11  I...fe..S.....z.
0050  2c 95 4a f2 b3 17 a8 c4 e6 e4 75 1f 65 87 a0 28  ,.J.....u.e..(
0060  66 85 c0 58 ae 48 18 9b b5 63 8b 48 08 84 4c 0d  f..X.H...C.H..L.
0070  bf ac a5 e0 52 68 ad ba 93 97 06 99 21 0c 29 e5  ....Rh.....!).
0080  66 8b 17 01 85 6f 63 b7 f6 b5 00 d3 1f 92 ca 8b  f....oc.....
0090  53 e7 98 b6 45 ca 67                               S...E.g

=====
Sa enkripcijom(Security = Transport)
=====
0000  93 00 00 00 01 00 00 00 39 2e 5c 38 37 d1 3b 47  .....9.\87.;G
0010  05 00 00 00 b6 31 45 44 f3 9c 9c ad f6 cb 8d 58  ....1ED.....X
0020  d2 9c 35 d7 23 5d 77 3e d8 0f cc 73 82 48 69 aa  ..5.#]w>...s.Hi.
0030  a9 92 17 6e bc 08 f1 54 d1 54 76 10 a7 0f 63 6e  ...n...T.Tv...cn
0040  43 50 da 93 b6 9b d8 e4 09 d9 65 54 ad 20 23 16  CP.....eT. #.
0050  71 ee ff 55 6c a5 01 0f 2e 54 79 45 a5 18 64 12  q..Ul....TyE..d.
0060  59 5c a1 bf ae bc e1 20 06 32 4c d5 a2 a4 12 d0  Y\.....2L....
0070  c5 ea 7e 12 fb 06 2e 6b 49 fa 66 e9 d0 79 b5 8f  ..~....kI.f..y..
0080  c6 1b 2f 8b 88 23 74 ff c8 73 d1 49 56 ac cc 27  ../.#t...s.IV..'
0090  ed 63 3d 1b 8c c1 59                               .c=...Y

```

Slika 4. Razmenjena poruka za Transport mod

Na slici 5. Binding.Security.Mode je podešen na None. Razlika u odnosu na Transport je uočljivija jer se određeni podaci ipak mogu primetiti. Ukoliko nemamo enkripciju i poruka se razmenjuje bez ikakve zaštite možemo doći do otvorenog teksta i saznati šta su to klijenti razmenjivali. Ukoliko se poruka koja se razmenjuje kriptuje njen sadržaj se ne može pročitati, bez obzira i na None mod koji je postavljen.


```

Bez enkripcije(Security = None)
=====
0000 06 99 02 33 26 68 74 74 70 3a 2f 2f 74 65 6d 70 ...3$http://temp
0010 75 72 69 2e 6f 72 67 2f 49 43 6c 69 65 6e 74 43 uri.org/IClientC
0020 6f 6e 74 72 61 63 74 2f 50 61 79 03 50 61 79 07 ontract/Pay.Pay.
0030 6d 65 73 73 61 67 65 56 02 0b 01 73 04 0b 01 61 messageV...s...a
0040 06 56 08 44 0a 1e 00 82 ab 31 44 1a ad 71 3d 43 .V.D.....1D..q=C
0050 8a 2d 93 b3 4c ab 15 6e eb 4c 5d 21 8f 44 2c 44 .-..L..n.L]!.D,D
0060 2a ab 14 01 40 17 56 73 44 65 62 75 67 67 65 72 *...@.VsDebugger
0070 43 61 75 73 61 6c 69 74 79 44 61 74 61 08 41 68 CausalityData.Ah
0080 74 74 70 3a 2f 2f 73 63 68 65 6d 61 73 2e 6d 69 ttp://schemas.mi
0090 63 72 6f 73 6f 66 74 2e 63 6f 6d 2f 76 73 74 75 crosoft.com/vstu
00a0 64 69 6f 2f 64 69 61 67 6e 6f 73 74 69 63 73 2f dio/diagnostics/
00b0 73 65 72 76 69 63 65 6d 6f 64 65 6c 73 69 6e 6b servicemodelsink
00c0 9f 3c b8 80 cf a3 d8 71 44 aa 17 8f d6 44 98 c8 .<.....qD....D..
00d0 6d 2d ce 24 fd 36 00 00 00 00 08 b4 ef 41 53 48 m-$.6.....ASH
00e0 1e 4a bd 2d 2e 01 80 c8 c6 72 ef 68 ac 6a b9 47 .J-.....r.h.j.G
00f0 bc 4f 82 4c 5a f4 39 33 95 21 00 09 00 00 44 0c .O.LZ.93.!...D.
0100 1e 00 82 ab 03 01 56 0e 42 33 0a 07 42 35 9e 06 .....V.B3..B5..
0110 70 6f 7a 64 72 61 9f 01 76 01 01 01 pozdra..v...

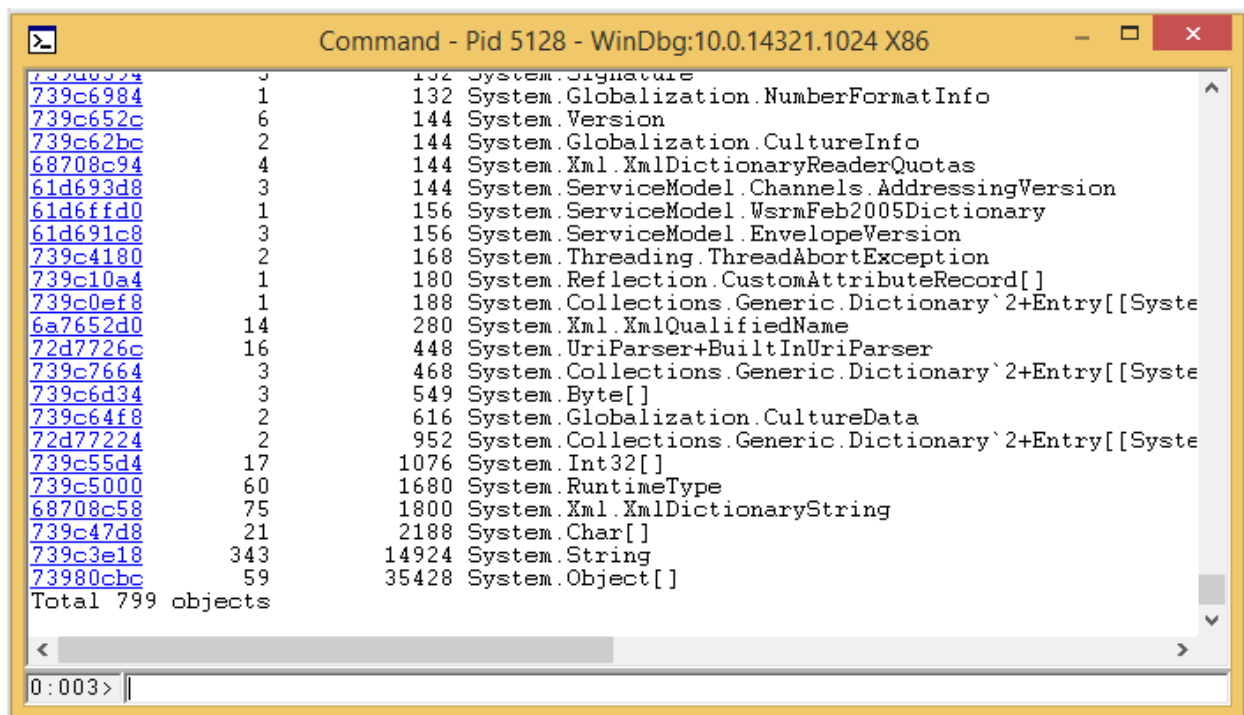
Sa enkripcijom(Security = None)
=====
0000 06 a2 02 33 26 68 74 74 70 3a 2f 2f 74 65 6d 70 ...3$http://temp
0010 75 72 69 2e 6f 72 67 2f 49 43 6c 69 65 6e 74 43 uri.org/IClientC
0020 6f 6e 74 72 61 63 74 2f 50 61 79 03 50 61 79 07 ontract/Pay.Pay.
0030 6d 65 73 73 61 67 65 56 02 0b 01 73 04 0b 01 61 messageV...s...a
0040 06 56 08 44 0a 1e 00 82 ab 31 44 1a ad 33 71 a5 .V.D.....1D..3q.
0050 cb ac 54 17 4a aa 91 d5 fc 5e 01 44 9d 44 2c 44 ..T.J....^.D.D,D
0060 2a ab 14 01 40 17 56 73 44 65 62 75 67 67 65 72 *...@.VsDebugger
0070 43 61 75 73 61 6c 69 74 79 44 61 74 61 08 41 68 CausalityData.Ah
0080 74 74 70 3a 2f 2f 73 63 68 65 6d 61 73 2e 6d 69 ttp://schemas.mi
0090 63 72 6f 73 6f 66 74 2e 63 6f 6d 2f 76 73 74 75 crosoft.com/vstu
00a0 64 69 6f 2f 64 69 61 67 6e 6f 73 74 69 63 73 2f dio/diagnostics/
00b0 73 65 72 76 69 63 65 6d 6f 64 65 6c 73 69 6e 6b servicemodelsink
00c0 9f 3c b8 80 cf a3 ae f8 4e a4 93 63 57 41 b2 ee .<.....N..cWA..
00d0 a5 49 c7 08 a9 0a 00 00 00 00 1b 0e c5 05 1a 30 .I.....0
00e0 5e 46 98 21 69 e6 2c 3a f5 25 b7 94 06 bd 31 80 ^F.!i.,.%.1.
00f0 a7 45 b2 7b 37 e5 69 4e 33 f9 00 09 00 00 44 0c .E.{7.iN3.....D.
0100 1e 00 82 ab 03 01 56 0e 42 33 0a 07 42 35 9e 0f .....V.B3..B5..
0110 15 dc 4f 3d 9f 9a cd 8b 79 b6 1e 80 92 ba c3 9f ..O=....y.....
0120 01 be 01 01 01 .....

```

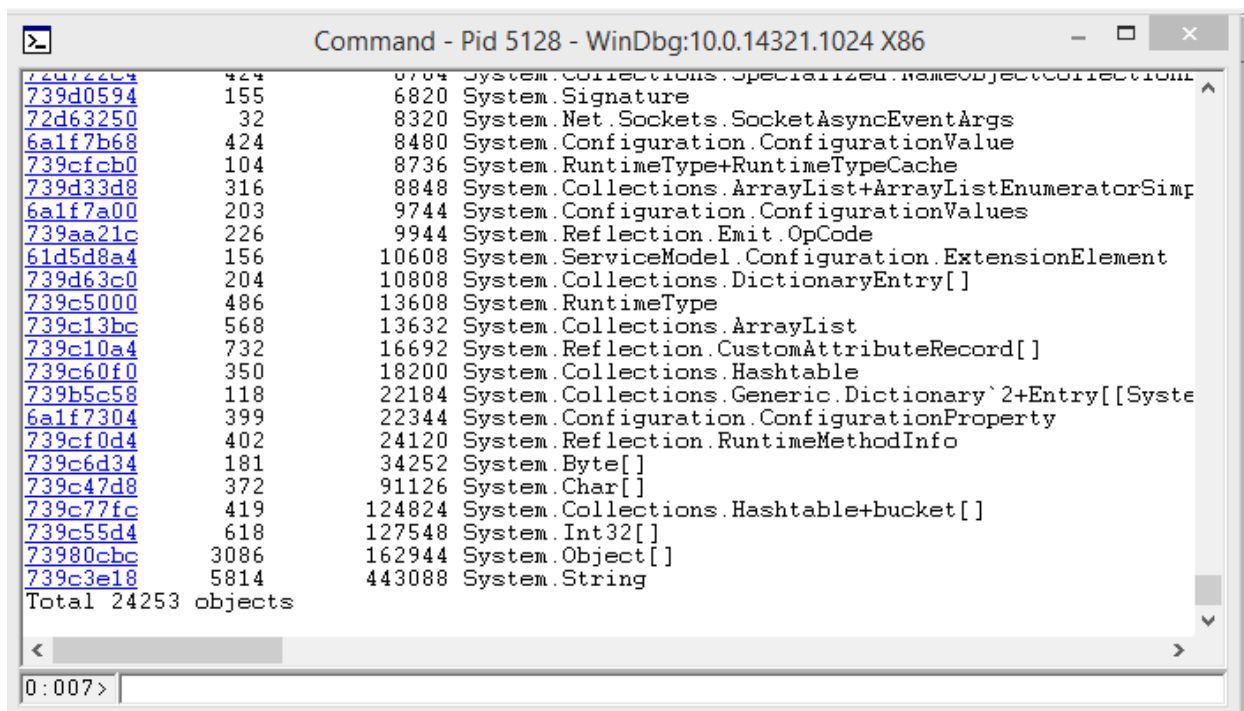
Slika 5. Razmenjena poruka za None mod

3.2. Analiza performansi uz pomoć WinDbg-a i Performance Monitor-a

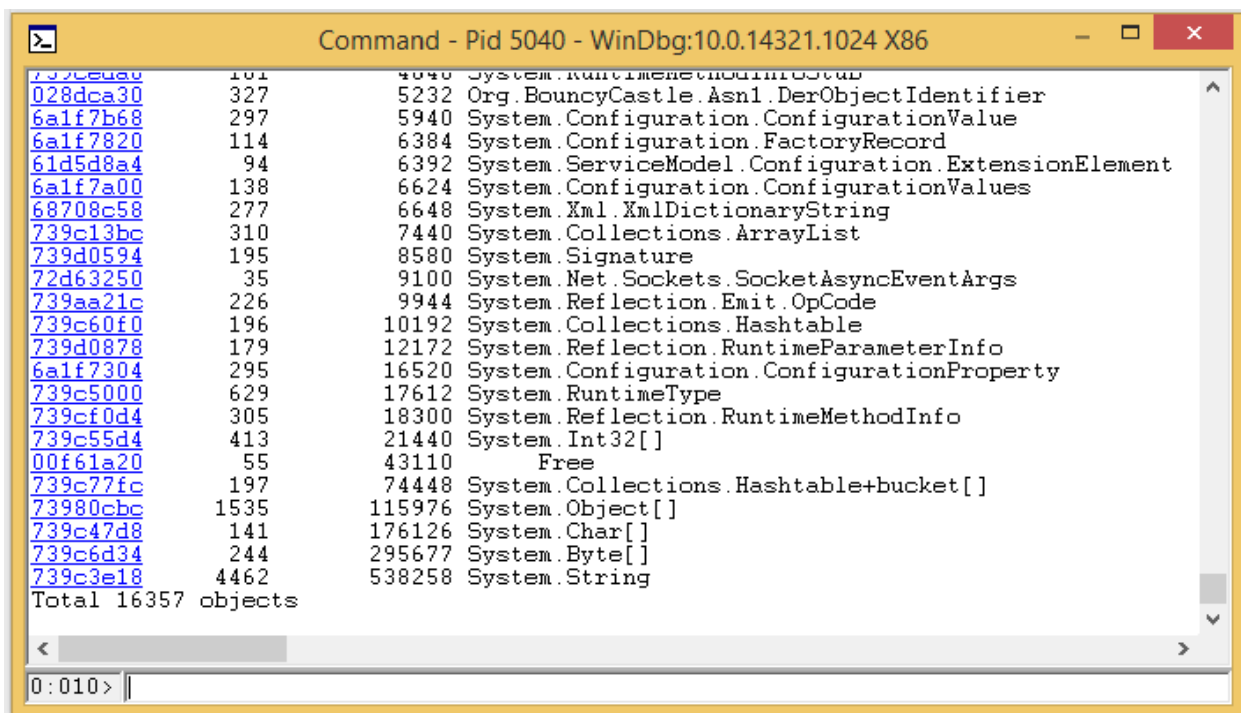
Uz pomoć alata WinDbg analiziran je scenario pokretanja CA servera i registrovanja novog klijenta korišćenjem RA komponente. Na slikama 6, 7 i 8 prikazano je stanje heap-a u ključnim trenucima.



Slika 6. Stanje heap-a neposredno pre hostovanja CA servisa



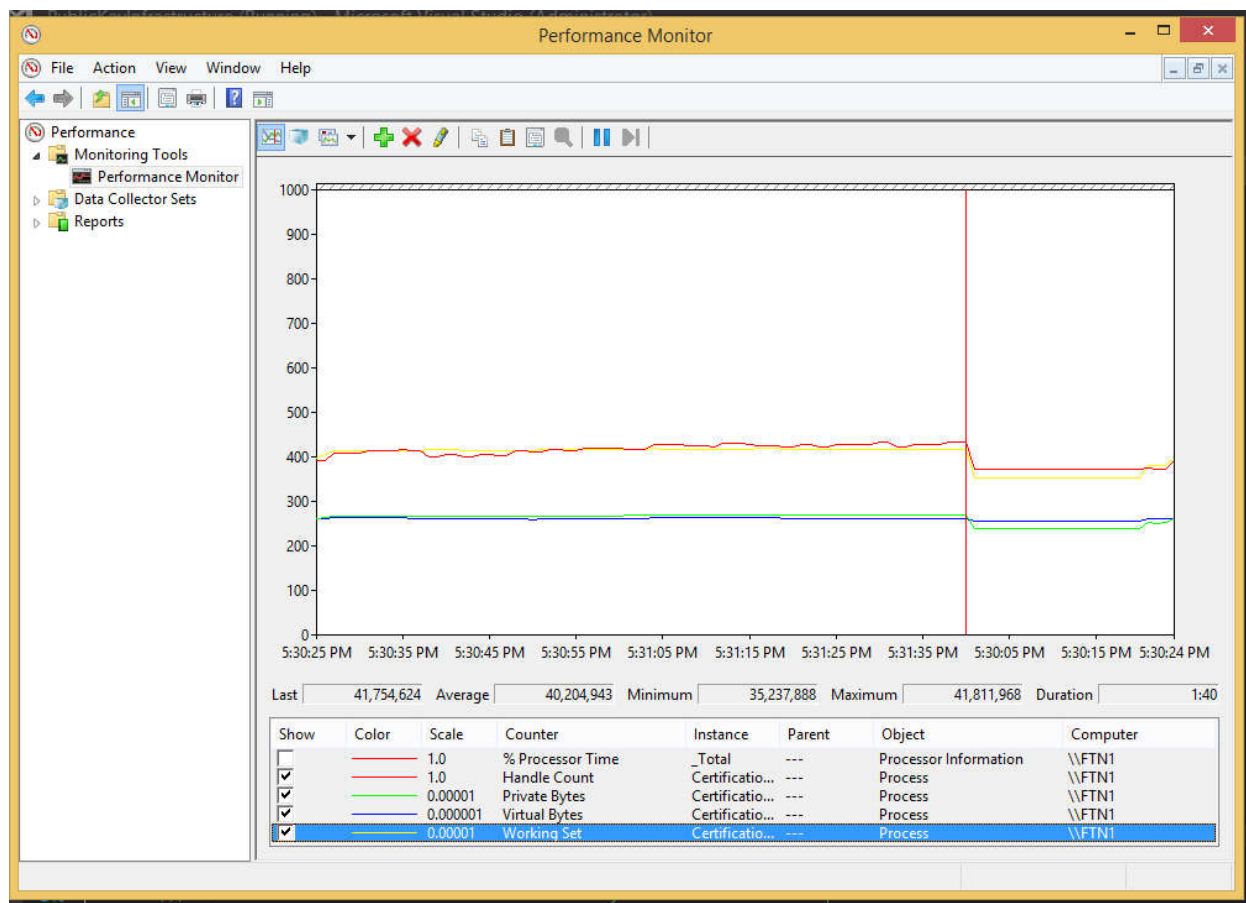
Slika 7. Stanje heap-a neposredno nakon hostovanja CA servisa



Slika 8. Stanje heap-a nakon uspešnog registrovanja novog klijenta

Na osnovu prikazanih stanja heap-a može se uočiti porast broja zauzetih objekata prilikom inicijalnog hostovanja CA servera. Porast broja zauzetih objekata se javlja kao posledica instanciranja klasa CAProxy i CertificationAuthorization. Međutim, nakon uspešnog registrovanja novog klijenta (slika 8) dolazi do приметnog smanjenja broja zauzetih objekata na heap-u iz razloga što su u ovom trenutku dealocirani objekti koji su prethodno služili za inicijalizaciju CA servera.

Sledeći test u okviru kojeg su analizirane performanse CA servera podrazumeva nekoliko uzastopnih registracija i validacija klijenata u određenom vremenskom intervalu (pogledati sliku 9). Pomoću alata Performance Monitor analizirane su promene sledećih vrednosti: handle count, private bytes, virtual bytes i working set. Kao što se može primetiti vrednosti private bytes i virtual bytes uz manje oscilacije ostaje manje-više nepromenjene dok se handle count i working set vraćaju na početne vrednosti prilikom gašenja CA servera.



Slika 9. Stanje na Performance Monitor-u