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

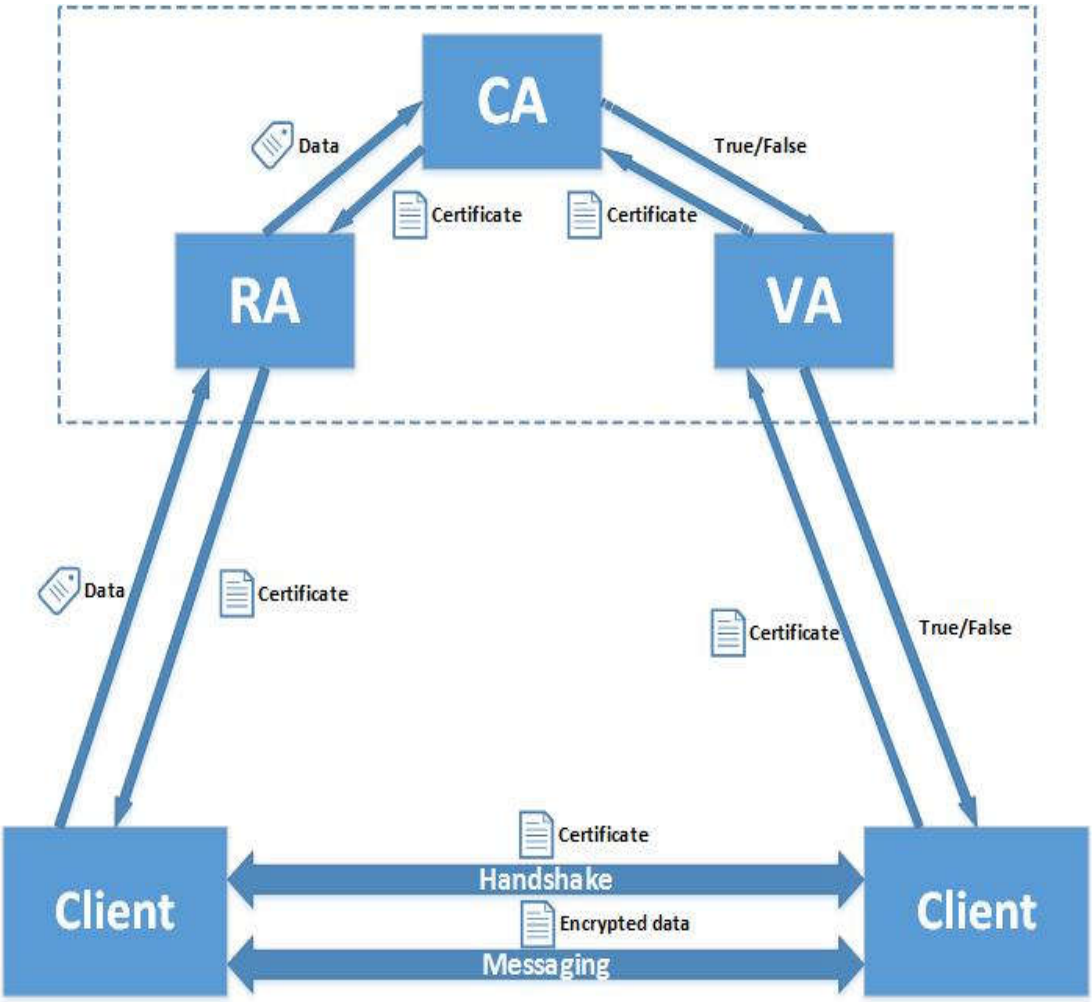
Asistent: Stefan Dejanović

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. Vrši 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.

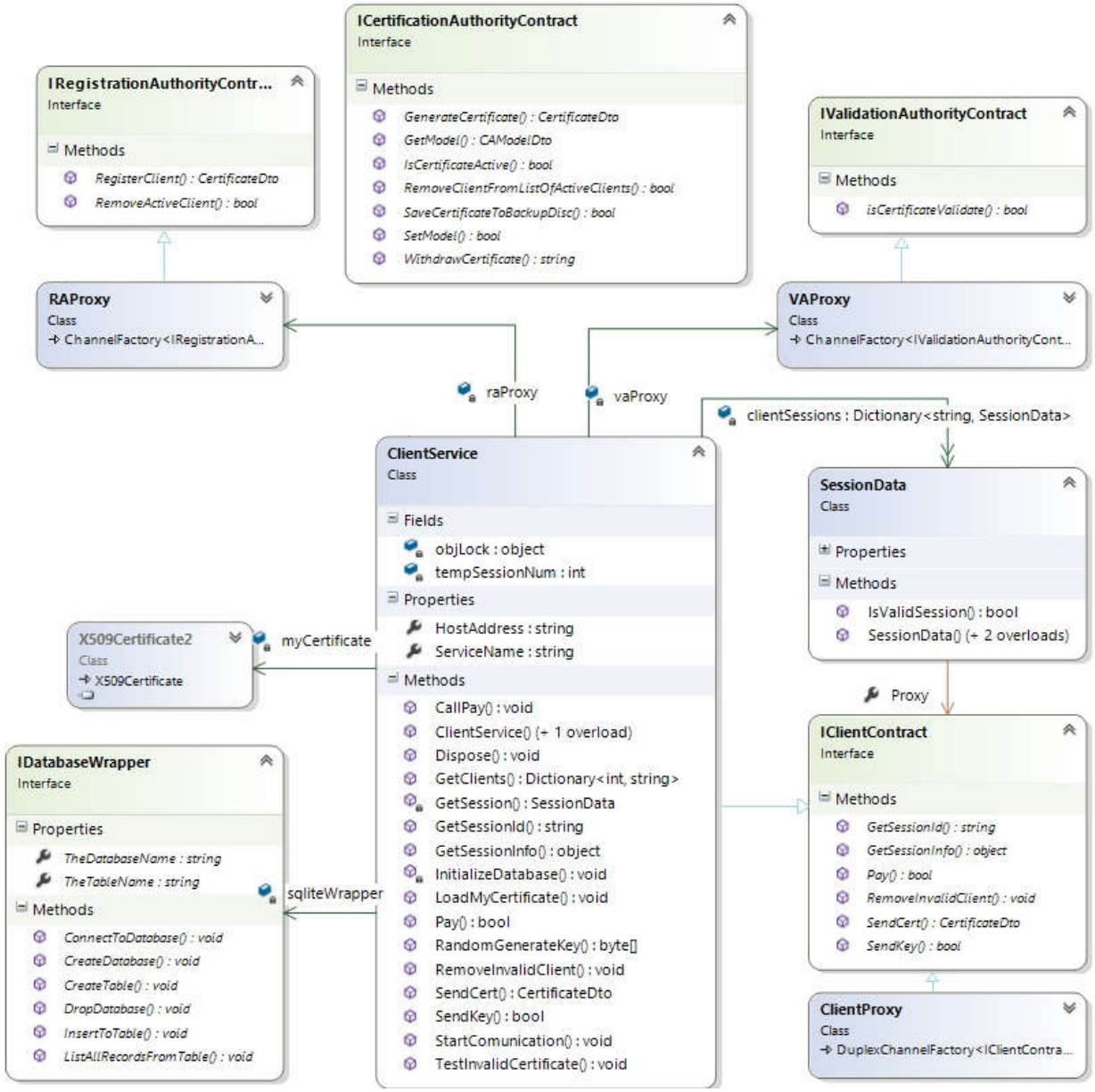
Da bi klijenti uspešno komunicirali potrebno je da prvo budu registrovani. Za registraciju se obraćaju RA komponenti. Prilikom uspostave komunikacije vrše obostranu validaciju sertifikata, tako što se obraćaju VA komponenti sa zahtevom validiranja sertifikata druge strane. Nakon uspšne validacije, klijenti 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 raspologanje `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

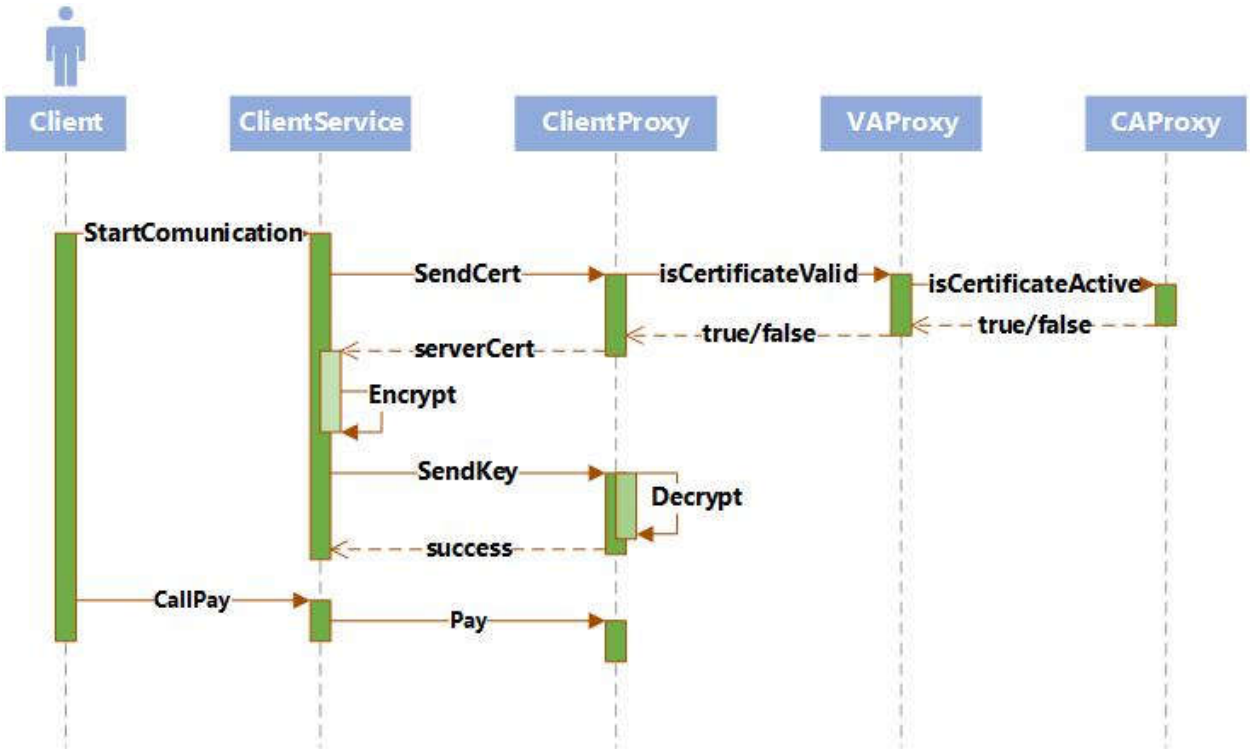
Svaka klijentska aplikacija može da se ponaša i kao klijent i kao servis u komunikaciji sa drugim klijentima. Za potrebe komunikacije klijent nudi `IClientContract` interfejs (Listing 4).

```
[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

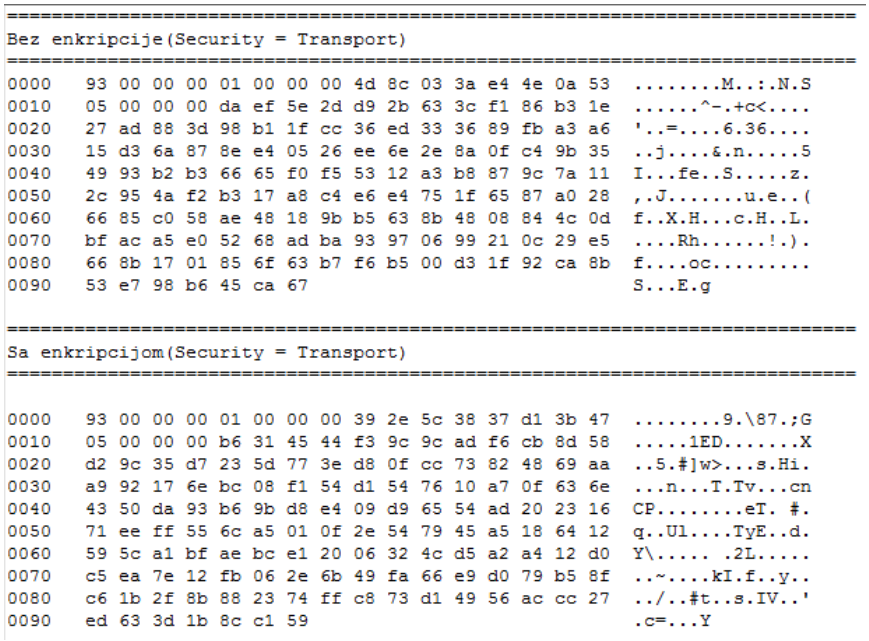
Rezultati testiranja su prikazani u tabeli 1. Testovi su vršeni na procesoru Intel Pentium 2 x 2.20 GHz. Iz ovih nekoliko testova se zaključuje da veličina poruke direktno utiče na brzinu izvršavanja algoritma, kada je paralelna obrada u pitanju. Kada je poruka prilično mala, u ovom slučaju samo 1 KB,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

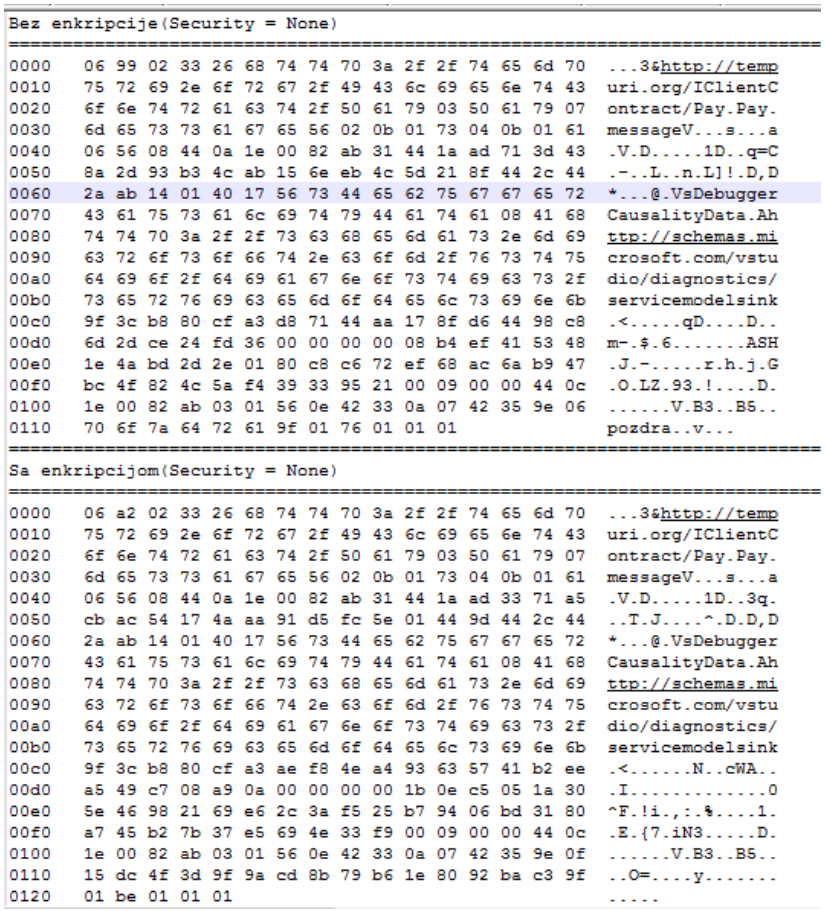
2.1 Analiza enkriptovanja uz pomoć Wireshark programa

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.



Slika 4. Razmenjena poruka za Transport mod

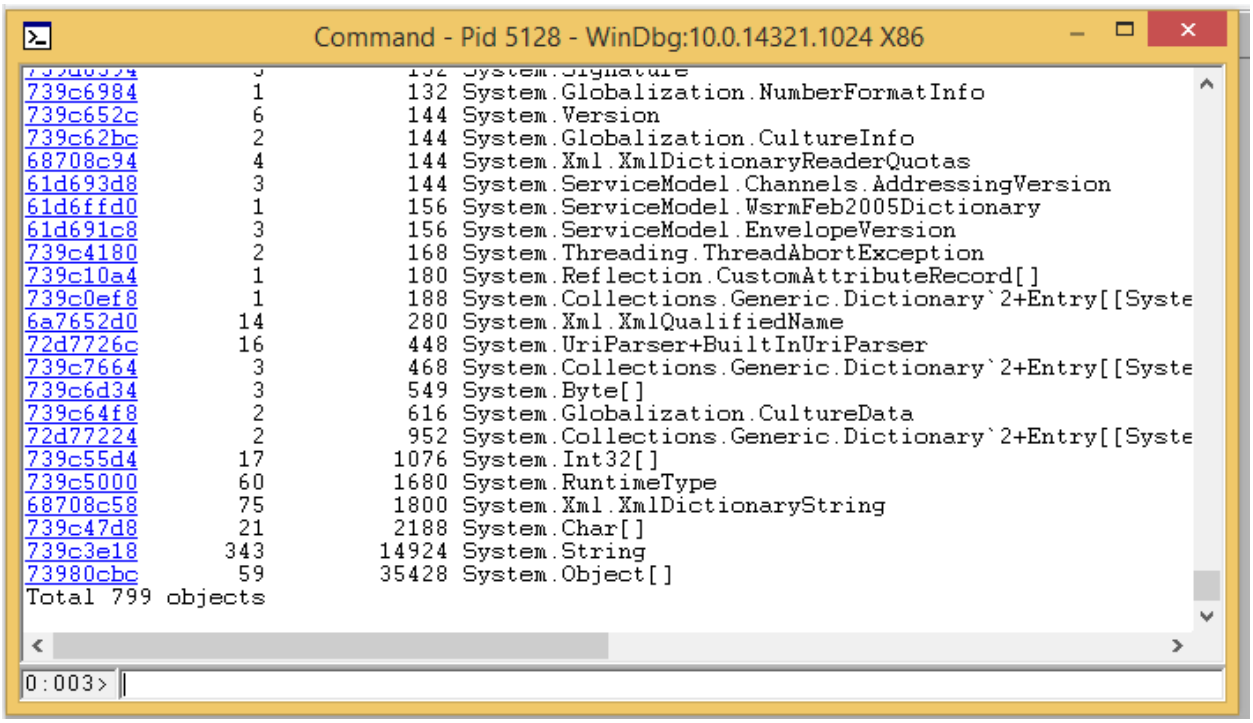
Na slici 5. Binding.Security.Mode je podešen na None. Razlika u odnosu na Transport je uočljivija jer se određeni podaci ipak mogu primetiti. Ukoliko nemamo enkripciju i poruka se razmenjuje bez ikakve zaštite možemo doći do otvorenog teksta i saznati šta su to klijenti razmenjivali. Ukoliko se poruka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



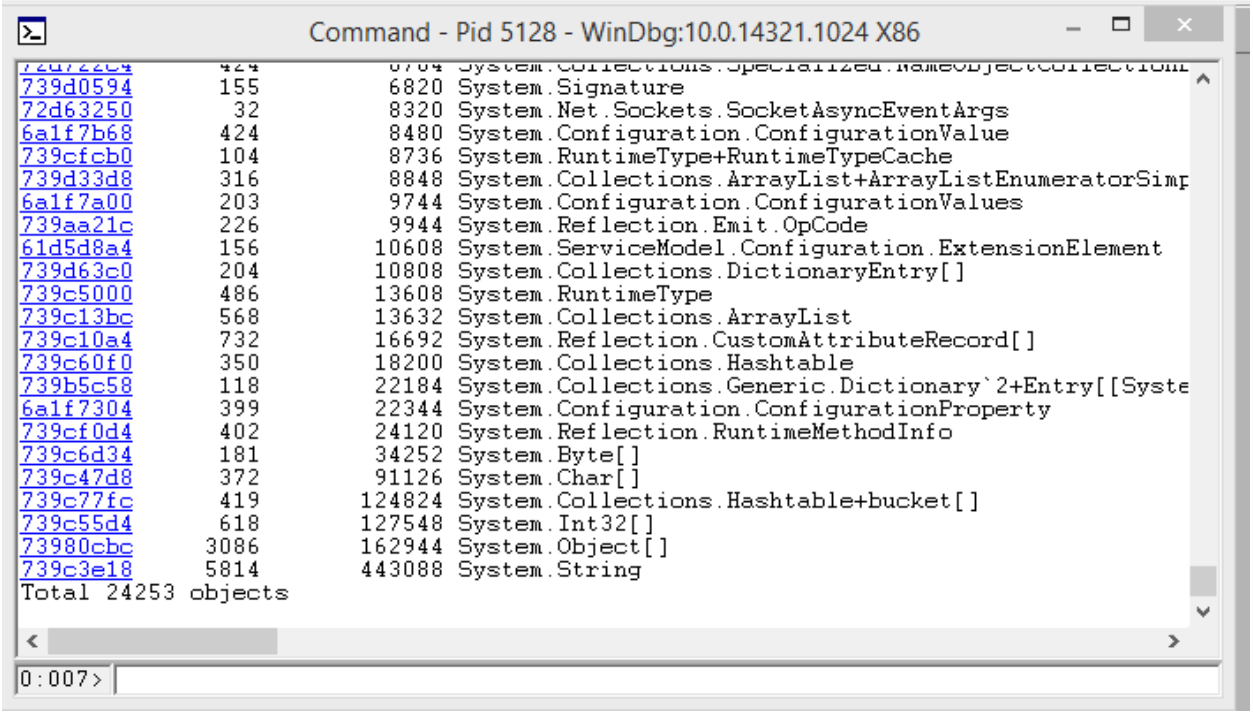
Slika 5. Razmenjena poruka za None mod

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

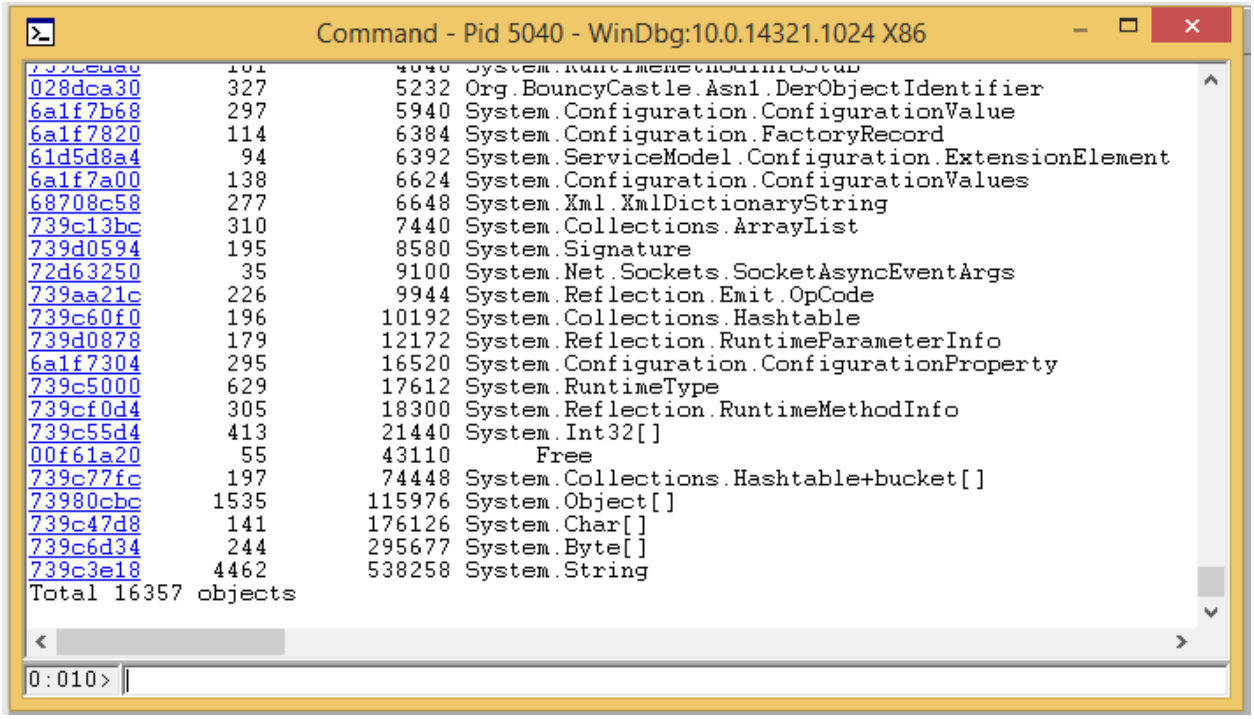
Uz pomoć alata WinDbg analiziran je scenario pokretanja CA servera i registrovanja novog klijenta korišćenjem RA komponente. Na slikama 6, 7 i 8 prikazano je stanje heap-a u ključnim trenucima.



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



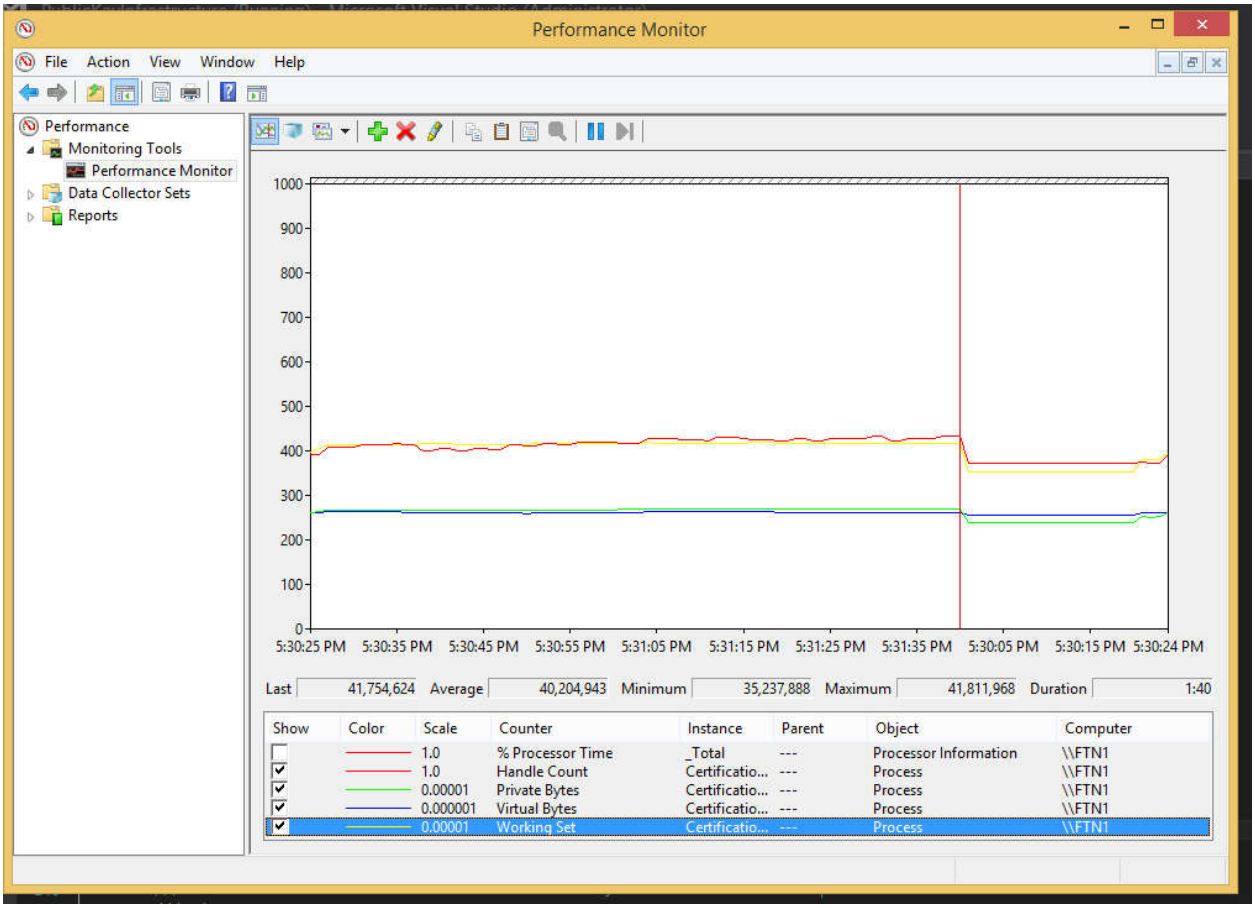
Slika 7. Stanje heap-a neposredno nakon hostovanja CA servisa



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 primetnog 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