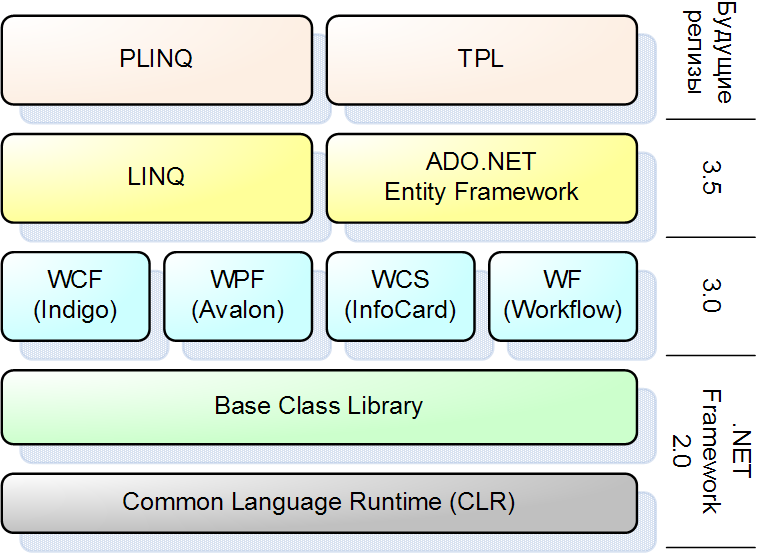
**Գլուխ 2**

**Օգտագործվող գործիքամիջոցների և միջավայրի նկարագրությունը**

**1.1 WPF միջավայրի նկարագրությունը**

Windows Presentation Foundation (WPF) համակարգ է, որը նախատեսված է Windows-ի հաճախորդային ծրագրերի կառուցման համար, որոնք ունեն օգտատիրոջ հետ փոխհարաբերության վիզուալ գրավիչ հնարավորություն և հանդիսանում է ենթահամակարգ .Net Framework կազմում (սկսած 3.0 տարբերակից), որը օգտագործում է XAML լեզուն: WPF ներդրված է Windows Vista-ում (.Net Framework 3.0), Windows 7-ում (.Net Framework 3.5), Windows 8-ում (.Net Framework 4.0 և 4.5): WPF-ի միջոցով հնարավոր է ստեղծել ինչպես ավտոնոմ, այնպես էլ բրաուզերով մեկնարկող բազում ծրագրեր:



*WPF-ը .NET-ի կազմում*

WPF-ի հիմքում ընկած է վիզուալիզացիայի վեկտորային համակարգը, որը ստեղծված է ժամանակակից գրաֆիկական սարքավորումների հնարավորությունները հաշվի առնելով: WPF-ը հնարավորություն է տալիս ստեղծելու վիզուալ ինտերֆեյս` ներառելով XAML լեզու, կառավարման էլեմենտներ, երկչափ և եռաչափ գրաֆիկա, անիմացիա, ոճեր, շաբլոններ, փաստաթղթեր, տեկստ, մուլտիմեդիա և ձևավորում:

WPF-ի հիմքում ընկած գրաֆիկական տեխնոլոգիան DirectX-ն է, ի տարբերություն Windows Forms-ից, որտեղ օգտագործվում է GDI/GDI+: WPF-ի արտադրողականությունը բարձր է, քան GDI+ -ինը` DirectX-ի միջոցով ապարատային արագացումը օգտագործելու հաշվին:

WPF-ի հետ աշխատելու համար պահանջվում է ցանկացած լեզու, որը համատեղելի է .Net-ի հետ: Այդ լեզուների շարքին են պատկանում C#, VB, C++, Ruby, Python, Dephi (Prism), Lua և ծատ այլ լեզուներ: Լիարժեք աշխատանքի համար հնարավոր էօգտագործել և' Visual Studio և Expression Blend: Առաջինը կողմնորոշված է ծրագրավորման վրա, իսկ երկրորդը` դիզայնի, և հնարավորություն է տալիս անել շատ բաներ` առանց XAML-ի խմբագրման: Ինչպես օրինակ անիմացիաներ, ոճավորում, կառավարման էլեմենտների ստեղծում և այլն:

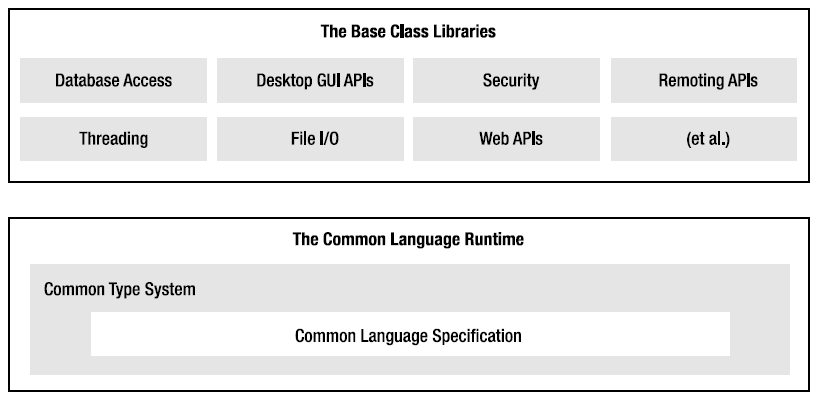
Ինչպես արդեն նշեցինք, WPF ծրագրերում, սովորաբար որպես ինտերֆեյսի նկարագրման միջոց, օգտագործվում է XAML (Extendable Application Markup Language) նշիչնեչի լեզուն: XAML – ը իրենից ներկայացնում է XML, որում փաստացի իրականացված են .NET հենքի տիպեր: Ինչպես նաև իրականացված է կոդի և դիզայնի բաժանման մոդելը: Բացի այդ կա տարրերի ոճի ներդրված աջակցում, իսկ հենց տարրերը շատ հեշտ բաժանվում են երկրորդ մակարդակի կառավարման տարրերի, որոնք իրենց հերթին էլ են բաժանվում մինչև վեկտորային մարմինների և հատկություն/գործողության: Դա թույլ է տալիս հեշտությամբ որոշել ոճ ցանկացած էլեմենտի համար:

**1.2 .Net Framework – ի նկարագրությունը**

.NET հենքը ստեղծված է աջակցելու բարդ և խիստ բաշխված կոմպոնենտային ծրագրերի նախագծման և աշխատեցման պրոցեսներին: Նա հնարավորություն է տալիս համատեղ օգտագործել տարբեր ծրագրային լեզուներ, ինչպես նաև ապահովում է ծրագրերի անվտանգությունը, տեղափոխելիությունը և Windows ՕՀ-ի ծրագրավորման ընդհանուր մոդելը: Ծրագրավորողի տեսանկյունից .NET-ը իրենից ներկայացնում է կատարող միջավայր և բազային տիպերի ահռելի գրադարան: Կատարման միջավայրի մակարդակը կոչվում է լեզուների համընդհանուր կատարման միջավայր (Common Language Runtime-CLR): Այս համակարգը կառավարում է ծրագրերի կատարումը: CLR-ը հանդիսանալով .NET-ի բաղադրիչ մաս, ապահովում է բազմալեզու ծրագրավորում, ինչպես նաև ծրագրերի տեղափոխելիությունը և անվտանգ կատարումը: CLR-ի հիմնական խնդիրը դա .NET-ի տիպերի ավտոմատ հայտնաբերման, բեռնման և կառավարման գործընթացն է(ծրագրավորողի փոխարեն): CLR-ը նաև զբաղվում է մի շարք ցածր մակարդակի խնդիրներով, ինչպիսիք են` հիշողության կառավարումը, ծրագրի սպասարկումը, հոսքերի վերամշակմամբ և անվտանգության հետ կապված զանազան ստուգներով: .NET-ի մյուս բաղադրիչը դա ընդհանուր տիպերի համակարգն է(Common Type System) կամ կրճատ` CTS համակարգ: CTS-ի հատկորոշման մեջ ներկայացված է բոլոր հնարավոր տիպերի և ծրագրային կառուցվածքների ամբողջ նկարագարությունը, որոնց աջակցում է կատարող միջավայրը, այն, թե ինչպես են այդ էությունները միմյանց հետ համագործակցում, և այն, թե ինչպես կարող են նրանք հանդես գալ որպես .NET-ի մետատվյալ:

Կարևոր է հասկանալ , որ CTS- ում ցանկացած որոշված ֆունկցիա կարող է չաջակցվել .NET-ի որոշակի լեզուների հետ: Դրա համար գոյություն ունի համընդհանուր լեզուների հատկորոշում (Common Language Specifiaction) կամ կրճատ` CLS հատկորոշում, որում նկարագրված է միայն այն ընդհանուր տիպերի ենթաբազմությունը և ծրագրային կառուցվածքները, որոնց կարող են աջակցել .NET –ի ծրագրավորման բոլոր լեզուները:

Ուստի .NET –ի միայն ֆունկցիոնալ հատկություններով տիպեր կառուցելու դեպքում, որոնք դիտարկված են CLS-ում, կարելի է լիովին վստահ լինել նրանում , որ .NET–ի հետ համատեղելի բոլոր լեզուները կարող են դրանք օգտագործել:Եվ հակառակը` CLS-ում չգտնվող որևէ տիպի տվյալների կամ ծրագրավորման կառուցվածքների օգտագործման դեպքում չի կարելի կարծել, որ .NET–ի հետ համատեղելի բոլոր լեզուները կարող են դրանք օգտագործել:

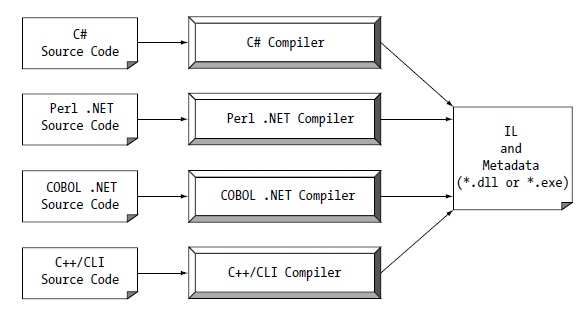


*CLR-ի, CTS-ի, CLS-ի և հենքի բազային դասերի գրադարանների միջև հարաբերությունը*

.NET-ի ծրագրավորման լեզուներից յուր­­­աքանչյուրն ընտրելիս, կարևոր է հիշել, որ .NET-ի երկուական կոդերի ֆայլերը ունեն նույն ընդլայնումները ինչ որ Win32-ի չկառավարվող ծրագրերի ֆայլերը(\* . dll կամ \* . ехе), բայց նրանց ներքին

կառուցվածքը տարբեր է: .NET-ի երկուական կոդերի ֆայլերը պարունակում են ոչ թե հատկորոշիչ տվյալներ այլ ընդհակառակը` հենքից անկախ տվյալներ, միջանկյալ

լեզվով(Intermediat Language -IL), ինչպես նաև տիպերի մետատվյալներ:



*Բոլոր .NET կոմպիլիատորները գեներացնում են IL կոդ և մետատվյալներ*

.NET-ի կոմպիլյատորի միջոցով \*.dll կամ \*.exe ֆայլ ստեղծելիս, ստացվում է մեծ երկուական օբյեկտ, որը կոչվում է հավաքածու (assembley): Այդ հավաքածուի մեջ է գտնվում CIL կոդը, որը հիմնովին նման է Java-ի բայթ կոդին նրանով, որ չի թարգմանվում հենքին հատուկ կոդի, մինչև չի լինում դրա անհրաժեշտությունը: Սովորաբար անհրաժեշտությունը լինում է, երբ CIL-կանոնների մի մասը դիմում է ֆունկցիայի, որը գտնվում է .NET-ի կատարող միջավայրում:

**1.3 C# լեզվի նկարագրությունը**

Քանի որ .NET-ը այդքան տարբերվում է իրեն նախորդող տեխնոլոգիաներից, Microsoft-ում ստեղծեցին հատուկ հենքի տակ աշխատող նոր ծրագրավորման լեզու` C#: C#-ը հանդիսանում է C ծրագրավորման լեզվի ընտանիքի անդամ դրա համար է, որ ունեն նման սինտակսիս:

Սինտակսիսի տեսանկյունից C#-ը հանդիսանում է մի քանի լեզվից ստեղծված գործնականում հզոր միջավայր, ինչպես C++-ը: Թվարկենք C# լեզվի մի քանի հատկություն՝

* Ցուցիչների կիրառման կարիք չկա: C# լեզվով գրված ծրագրերում սովորաբար ցուցիչներ կիրառելու պահանջ չի լինում (չնայած լեզուն հնարավորություն տալիս է օգտագործել ցուցիչներ):
* Հիշողության կառավարումը իրականացվում է ավտոմատ, այսպես կոչված` աղբ հավաքողի կողմից, դրա համար C#-ում չի կիրառվում delete հանգույցային բառը:
* Տրամադրվում են ֆորմալ սինտակսիսային կառուցվածքային տիպերի, ինտերֆեյսների, ստրուկտուրաների, թվարկությունների և դելեգատների համար:

.NET-ի տվյալ թողարկման մեջ C# լեզուն թարմացվել է, և օժտված է մի շարք նորություններով, որոնցից կարևորները թվարկված են ստորև`

* Ոչ անհրաժեշտ պարամետրերի հնարավորություն, ինչպես նաև անունով արգումենտներ:
* Անդամների դինամիկ որոնման հնարավորություն կատարման ընթացքում, dynamic բանալիային բառի միջոցով:
* Ընդհանրացված տիպերի հետ աշխատանքը բավականին հեշտացել է, ի շնորհիվ տվյալների հեշտ արտապատկերմա:

C#-ով ծրագրավորման ժամանակ կարելի է ստեղծել միայն այնպիսի կոդ, որը կթարգմանվի միայն .NET-ի միջավայրում:

**1.4 Microsoft SQL Server**

Microsoft SQL Server, ռելացիոն տվյալների բազաների կառավարման համակարգ ([DBMS](http://hy.wikipedia.org/w/index.php?title=DBMS&action=edit&redlink=1))` մշակված [Microsoft](http://hy.wikipedia.org/wiki/Microsoft) կորպորացիայի կողմից: Հարցումների օգտագործվող հիմնական լեզուն [Transact-SQL](http://hy.wikipedia.org/w/index.php?title=Transact-SQL&action=edit&redlink=1) է, որը ստեղծվել է համատեղ` Microsoft-ի և Sybase-ի կողմից։ Transact-SQL-ը հանդիսանում է ANSI/ISO ստանդարտի իրագործումը՝ ըստ հարցումների ենթակառուցվածքային լեզվի (SQL) ընդլայնումների։ Օգտագործվում է անհատականից մինչև հսկա կազմակերպությունների տվյալների բազաների հետ աշխատանքների համար, այն մրցակցում է շուկայի տվյալ սեգմենտում այլ ռելացիոն տվյալների բազաների կառավարման համակարգերի հետ։

**Պատմությունը**

MS SQL Server-ի ելակետային կոդը (մինչ 7.0 տարբերակը) հիմնվում էր [Sybase SQL Server](http://hy.wikipedia.org/w/index.php?title=Sybase_SQL_Server&action=edit&redlink=1) կոդի վրա, և դա թույլ տվեց Microsoft-ին՝ դուրս գալ կազմակերպությունների տվյալների բազայի շուկա, որտեղ այդ ժամանակ մրցակցում էին [Oracle](http://hy.wikipedia.org/w/index.php?title=Oracle&action=edit&redlink=1), [IBM](http://hy.wikipedia.org/wiki/IBM), իսկ ավելի ուշ ինքը՝ [Sybase](http://hy.wikipedia.org/w/index.php?title=Sybase&action=edit&redlink=1)-ը։ Microsoft-ը, [Sybase](http://hy.wikipedia.org/w/index.php?title=Sybase&action=edit&redlink=1)-ը և [Ashton-Tate](http://hy.wikipedia.org/w/index.php?title=Ashton-Tate&action=edit&redlink=1)-ը ի սկզբանե միավորվեցին՝ մի ծրագրի ստեղծման և շուկա բացթողնման համար, որը ստացավ [SQL Server 1.0](http://hy.wikipedia.org/w/index.php?title=SQL_Server_1.0&action=edit&redlink=1) OS/2 -ի համար անվանումը ([1989](http://hy.wikipedia.org/wiki/1989) թ.), որը փաստացի համարժեքն էր Sybase SQL Server 3.0 Unix, VMS-ի և այլնի համար։ Microsoft SQL Server 4.2-ը թողարկվել է 1992թվականին և գտնվում էր Microsoft OS/2 1.3 տարբերակի օպերացիոն համակարգի կազմում։ Microsoft SQL Server Windows NT օպերացիոն համակարգի համար 4.21 տարբերակի պաշտոնական հրապարակումը տեղի ունեցավ հենց [Windows NT](http://hy.wikipedia.org/w/index.php?title=Windows_NT&action=edit&redlink=1)-ի (տարբերակ 3.1) պաշտոնական հրապարակման (հանրայնացման) հետ միաժամանակ։

[Microsoft SQL Server 6.0](http://hy.wikipedia.org/w/index.php?title=Microsoft_SQL_Server_6.0&action=edit&redlink=1)-ը առաջին [SQL Server](http://hy.wikipedia.org/w/index.php?title=SQL_Server&action=edit&redlink=1)-ի տարբերակն էր, որը ստեղծված էր բացառապես Windows NT-ի ճարտարապետության համար և առանց մշակման գործընթացում Sybase-ի մասնակցության։

Այն ժամանակ, երբ շուկա դուրս եկավ Windows NT օպերացիոն համակարգը, [Sybase](http://hy.wikipedia.org/w/index.php?title=Sybase&action=edit&redlink=1)-ն ու Microsoft-ը բաժանվեցել էին և հետևում էին ծրագրային արտադրանքի սեփական մոդելներին և մարքեթինգային սխեմաներին։ Microsoft-ը ձգտում էր Windows-ի համար SQL Server-ների բոլոր տարբերակների գծով բացառիկ իրավունքի ձեռքբերմանը։ Ավելի ուշ, Sybase-ը փոխեց իր արտադրանքի անվանումը և անվանեց այն Adaptive Server Enterprise, խուսափելու համար Microsoft SQL Server-ի հետ շփոթելի լինելուց։ Մինչև 1994 թվականը Microsoft-ն արդեն ստացել էր երեք ծանուցում Sybase-ից՝ որպես ակնարկ Microsoft SQL Server-ի ծագման մասին։

Բաժանվելուց հետո ընկերությունները թողարկեցին մի քանի ինքնուրույն ծրագրեր։ SQL Server 7.0-ը դարձավ տվյալների բազայի առաջին սերվերը, որն օժտված էր օգտագործողի համար կառավարման իսկական գրաֆիկական ինտերֆեյսով։ Sybase-ի կողմից հեղինակային իրավունքի պահանջներից խուսափելու նպատակով 7-րդ տարբերակի ողջ ժառանգվող կոդը վերագրվեց։  
SQL Server 2005 տարբերակը ներկայացվել է 2005 թվականի նոյեմբերին։ Տարբերակի թողարկումն իրականացվում էր Visual Studio 2005-ի թողարկման հետ զուգահեռ։ Գոյություն ունի նաև Microsoft SQL Server-ի կրճատված տարբերակը՝ Microsoft SQL Server Express։ Այն հասանելի է բեռնման համար և կարող է անվճար տարածվել այն օգտագործող ծրագրային ապահովման հետ մեկտեղ։

SQL Server (SQL Server 2000)-ի նախորդ տարբերակի թողարկման պահից իրականացվում էին աշխատանքներ՝ մշակման ինտեգրացված ոլորտի և SQL Server 2005-ի կազմ մտնող մի շարք լրացուցիչ ենթահամակարգերի զարգացման ուղղությամբ։ Փոփոխություններից անմասն չմնացին նաև ETL (Տվյալների դուրսբերում, ձևափոխում և բեռնում) տեխնոլոգիայի իրացումը, որը մտնում էր SQL Server Integration Services (SSIS)-ի կազմի մեջ, ծանուցման սերվերը, տվյալների բազմաչափ մոդելների մշակման վերլուծական միջոցները (OLAP)-ը և անհրաժեշտ տեղեկատվության հավաքագրումը (երկու ծառայություններն էլ մտնում են Microsoft Analysis Services կազմի մեջ), ինչպես նաև մի քանի հաղորդագրության ծառայություններ, մասնավորապես՝ Service Broker և Notification Services։

Բացի այդ կատարվում էին աշխատանքներ նաև արտադրողականության բարելավման ուղղությամբ։

**1.4.1 Ֆունկցիոնալություն**

Տվյալների բազաների սերվերը Microsoft SQL Server որպես լեզվի հարցումներ օգտագործում է SQL լեզվի տարբերակը, որը ստացել է Transact-SQL (կրճատ T-SQL)։ T-SQL լեզուն [SQL-92](http://hy.wikipedia.org/w/index.php?title=SQL-92&action=edit&redlink=1)-ի (ISO ստանդարտ SQL լեզվի համար) իրականացումն է մի քանի ընդլայնումներով։ Պահվող պրոցեդուրաների համար T-SQL-ը թույլ է տալիս օգտագործել լրացուցիչ սինտաքսիս և ապահովում է գործարքի աջակցություն (կառավարման ծրագրով տվյալների բազաների փոխգործակցում)։

Microsoft SQL Server և Sybase ASE ցանցերի փոխգործակցության ժամանակ օգտագործվում է Tabular Data Stream (TDS, աղյուսակային տվյալների փոխանցման արձանագրություն) ծրագրային մակարդակի արձանագրություն։

TDS արձանագրությունը նույնպես իրականացվել է FreeTDS նախագծում, որպիսի տարբեր ծրագրերին տալ հնարավորություն փոխգործակցել Microsoft SQL Server և Sybase տվյալների բազաների հետ։

Տվյալներին մուտք ապահովելու համար Microsoft SQL Server-ը ապահովում է Open Database Connectivity (ODBC) – ծրագրերի փոխգործակվման ինտերֆեյս ՌՏԲԿՀ-ի հետ (Ռելացիոն տվյալների բազաների կառավարման համակարգ)։

SQL Server 2005 տարբերակը ապահովում է օգտագործողների վեբ-ծառայությունների միջոցով միացում՝ օգտագործելով SOAP արձանագրություն։ Այն թույլ է տալիս օգտագործողների ծրագրերին քրոսհարթակային կապով միանալ SQL Server-ին, նույնիսկ եթե ծրագրերը նախատեսված չեն Windows օպերացիոն համակարգի վրա։ Microsoft նույնպես թողարկել է հավաստագրված JDBC դրայվեր, որը թույլ է տալիս JAVA պլատֆորմի վրա աշխատող ծրագրերին (ինչպիսիք BEA и IBM WebSphere) կապվել Microsoft SQL Server-ին։

SQL Server-ը ապահովում է տվյալների բազաների հայլեացում և կլաստերզացիա։ SQL սերվերի կլաստերը նույն կոնֆիգուրացիա ունեցող սերվերների հավաքածու է. այդպիսի սխեման օգնում է աշխատանքի ծանրաբեռնվածությունը բաշխել մի քանի սերվերների միջև։ Բոլոր սերվերները ունեն միևնույն վիրտուալ անունը և տվյալները բաշխվում են կլաստերի IP հասցեների օգնությամբ, աշխատանքի փուլի ընթացքում։ Բացի այդ, մի սերվերի կլաստերի խափանման կամ անսարքության դեպքում հնարավոր է ծանրաբեռնման ավտոմատ փոխանցում ուրիշ սերվերին։

SQL սերվերը ապահովում է հավելյալ տվյալների կրկնապատկումը 3 սցենարներով.

Նկար. Ստեղծվում է տվյալների բազաների «նկարը», որը սերվերը ուղարկում է ստացողներին

Փոփոխումների պատմություն. Տվյալների բազաների բոլոր փոփոխությունները անընդհատ փոխանցվում են օգտագործողներին

Ուրիշ սերվերների հետ համաժամացում(սինխրոնիզացիա). Մի քանի սերվերների տվյալների բազաները համաժամացվում են միմյանց միջև։ Բոլոր տվյալների բազաների փոփոխությունները յուրաքանչյուր սերվերում կատարվում են միմյանցից անկախ, իսկ համաժամացման ժամանակ բոլոր տվյալները համեմատվում են։ Այդպիսի կրկնապատկման տեսակը ապահովում է տվյալների բազաների միջև տարաձայնությունների լուծման հնարավորությունը։

[SQL Server 2005](http://hy.wikipedia.org/w/index.php?title=SQL_Server_2005&action=edit&redlink=1)-ում ներդրված է [.NET Framework](http://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1)-ի ապահովում։ Դրա շնորհիվ տվյալների բազաների պահպանելի պրոցեդուրաները կարող են գրվել .NET պլատֆորմի ցանկացած լեզվով՝ օգտագործելով գրադարանների ամբողջ հավաքածուն, որը առկա է .NET Framework-ի համար, գումարած Common Type System-ը (Microsoft .NET Framework-ում տվյալների տեսակների դիմելու համակարգ)։ Սակայն, ուրիշ պրոցեսների հետ համեմատած, .NET Framework-ը, լինելով բազիսային համակարգ SQL Server 2005-ի համար, հատկացնում է հավելյալ հիշողություն և օգտագործում է SQL Server-ի կառավարման միջոցները, Windows-ի միջոցները օգտագործելու փոխարեն։ Դա բարձրացնում արտադրողականությունը՝ համեմատած Windows-ի ընդհանուր ալգորիթմների հետ, քանի որ ռեսուրսների բաշխման ալգորիթմները հատուկ կարգավորված են SQL Server-ի կառուցվածքներում օգտագործվելու համար։

**1.4.2 Ծրագրերի մշակում**

Microsoft-ը և մյուս ընկերությունները ստեղծում են մեծ քանակությամբ մշակման միջոցներ, որը թույլ է տալիս, օգտվելով Microsoft SQL Server-ի տվյալների բազաներից, ստեղծել բիզնես-ծրագրեր։ Microsoft SQL Server 2005-ը նույնպես իր մեջ ներառում է Common Language Runtime (CLR) [Microsoft .NET](http://hy.wikipedia.org/w/index.php?title=Microsoft_.NET&action=edit&redlink=1), որը թույլ է տալիս իրականացնել պահպանվող պրոցեդուրաներ և տարբեր ծրագրային ֆունկցիաներ՝ ստեղծված .NET (օրինակ, [VB.NET](http://hy.wikipedia.org/w/index.php?title=VB.NET&action=edit&redlink=1) կամ [C#](http://hy.wikipedia.org/wiki/C)) լեզուների պլատֆորմի վրա։ Microsoft-ի նախորդ ստեղծված միջոցների տարբերակները օգտագործել են միայն API, որպիսի ստանան Microsoft SQL Server-ին ֆունկցիոնալ մուտք։

**SQL Server Express Edition**

[Microsoft SQL Server Express](http://hy.wikipedia.org/w/index.php?title=Microsoft_SQL_Server_Express&action=edit&redlink=1)-ը SQL Server տարբերակի անվճար տարածվող տեսակն է, [MSDE](http://hy.wikipedia.org/w/index.php?title=MSDE&action=edit&redlink=1) համակարգի զարգացմամբ։ Տվյալ տարբերակը ունի տեխնիկական որոշ սահմանափակումներ։ Այդպիսի սահմանափակումները դարձնում են նրան մեծ տվյալներ բազաների ստեղծման համար ոչ պիտանի, բայց այն միանգամայն պիտանի է փոքր ընկերությունների սահմաններում ծրագրային կոմպլեքսների ներմուծման համար։ Լիարժեք ապահովում է տվյալների նոր տիպերի օգտագործումը, այդ թվում [XML](http://hy.wikipedia.org/wiki/XML)-հատկորոշումներ (սպեցիֆիկացիա)։ Փաստորեն, այն լիարժեք MS SQL Server է, որը ներառում է իր ծրագրավորման բոլոր բաղադրիչները, ապահովում է ազգային և[Unicode](http://hy.wikipedia.org/wiki/Unicode) այբուբեններ։ Այդ պատճառով օգտագործվում է ծրագրերում, նախագծման կամ ինքնուրույն հետազոտման համար։ Չկա ոչ մի խոչընդոտ՝ հետագայում կուտակած MS SQL Server ոչ էքսպրես տարբերակի տվյալների բազաների տեղակայման համար։ [2007](http://hy.wikipedia.org/wiki/2007) թ. Microsoft-ը թողարկեց գրաֆիկական ինտերֆեյսով առանձին գործիք այդ տարբերակի կառավարման համար, որը նույնպես կարելի է անվճար բեռնել ընկերության պաշտոնական կայքից։

**Սահմանափակումներ**

1 պրոցեսոր(այն կարող է տեղադրված լինել ցանկացած սերվերում)

1 գբ հասցեային հիշողություն

4 գբ բազայի առավելագույն չափս (10 գբ SQL Server 2008 R2-ի համար)

SQL Server Management Studio 2005-ի ինտերֆեյսից անհնար է տվյալների մուտքագրում/արտածում (2008 տարբերակի մեջ այդ հնարավորությունը կա)

[Windows](http://hy.wikipedia.org/wiki/Windows)-ի համար նախատեսված տարբերակներ

1992 - SQL Server 4.2

1993 - SQL Server 4.21 Windows NT-ի համար

[SQL Server 2008](http://hy.wikipedia.org/w/index.php?title=SQL_Server_2008&action=edit&redlink=1)-ը ուղղված է տվյալների ղեկավարումը ինքնակարգավորվող, ինքնակառավարվող և ինքնասպասարկող մեխանիզմ դարձնելուն. այդպիսի հնարավորությունների իրականացման համար ստեղծվել են SQL Server Always On տեխնոլոգիաները: Այն հնարավորություն է ընձեռում զրոյացնել սերվերի չաշխատելու վիճակի ժամանակը:

SQL Server 2008-ում ավելացվել է կառուցվածքային և կիսակառուցվածքային տվյալների ապահովումը, ներառելով` նկարների, ձայնային, վիդեո և այլ թվային մուլտիմեդիա տիպի ֆորմատներ։ Մուլտիմեդիա-ֆորմատների ապահովումը բազաներում թույլ տվեց հատուկ ֆունկցիաների փոխգործակցել այդ տիպի տվյալների հետ։

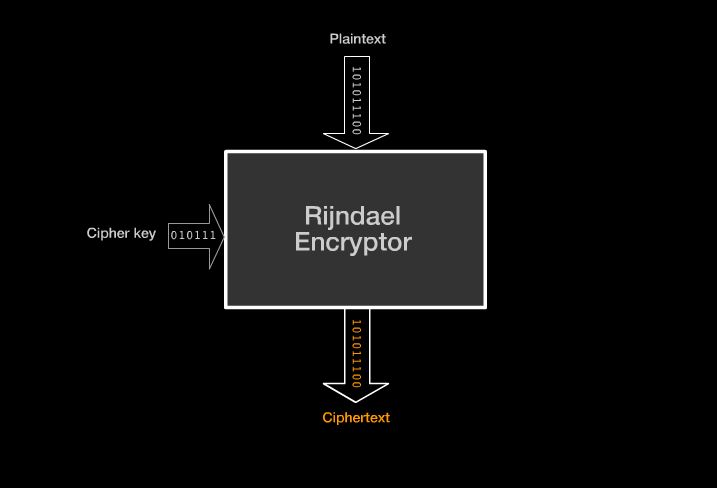
Բացի այդ, ներառվեցին նաև հատուկ ֆորմատներ՝ ամսաթիվ և ժամանակ և տարածային (անգլ. Spatial) տիպ հատուկ տարածակախյալ տիպի տվյալների համար։ Ոչ կառուցվածքային տվյալների համար ստեղծվեց հատուկ տիպ. օրինակ՝ Պատկեր։

SQL Server-ում կառավարման էֆեկտիվությունը բարձրացնելու համար ներառվեցին [Declarative Management Framework](http://hy.wikipedia.org/w/index.php?title=Declarative_Management_Framework&action=edit&redlink=1) գրադարանները, որի շնորհիվ հնարավոր է բաշխել լիազորություններ տվյալների բազաների կամ առանձին աղյուսակների համար։ Նորացվել են տվյալների խտացման եղանակները։ [. SQL Server Katmai](http://hy.wikipedia.org/w/index.php?title=._SQL_Server_Katmai&action=edit&redlink=1)-ն[[1]](http://hy.wikipedia.org/wiki/Microsoft_SQL_Server#cite_note-2k8-1)[[2]](http://hy.wikipedia.org/wiki/Microsoft_SQL_Server#cite_note-PaulFlessner-2)ապահովում է [ADO.NET Entity Framework](http://hy.wikipedia.org/w/index.php?title=ADO.NET_Entity_Framework&action=edit&redlink=1) գրադարանների հավաքածուն և կրկնման ծանուցումների միջոցներ ու տվյալների տիպի որոշում։

**1.5 AES: ալգորիթմի նկարագրությունը**

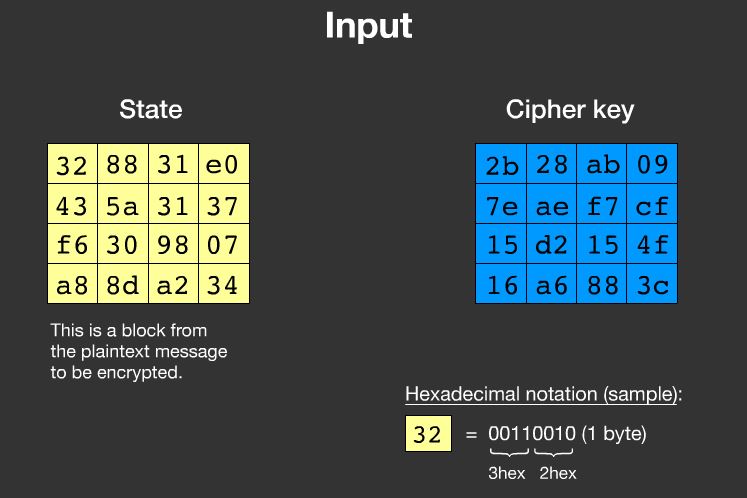
AES գաղտնագրման ալգորիթմը ստեղծվել է DES-ին փոխարինելու համար, որ գաղտնակայունությունն այդ ժամանակ արդեն խոցելի էր: Ի տարբերություն DES- ի AES-ում մտցվել է գաղտնիության մակարդակներ: Գաղտնագրումը կարելի է կատարել 128,192,256 բիթ երկարությամբ բանալիներով (կախված նրանից ինչ կարևորություն ունի գաղտնագրվող ինֆորմացիան): AES – ը այսօր աշխարհում ամենատարածված համաչափ գաղտնագրման ալգորիթմն է և մինչ այսօր չի կոտրվել:

AES – ը դա Rijndael ալգորիթմն է (չնայած կա որոշակի տարբերություն` Rijndael – ում գաղտնագրվող բլոկի չափը կարող է լինել 128,192 կամ 256 բիթ, իսկ AES – ում ֆիքսված է և հավասար է 128 - բիթի): Ալգորիթմում օգտագործվում են հետևյալ ընդունված նշանակումները .

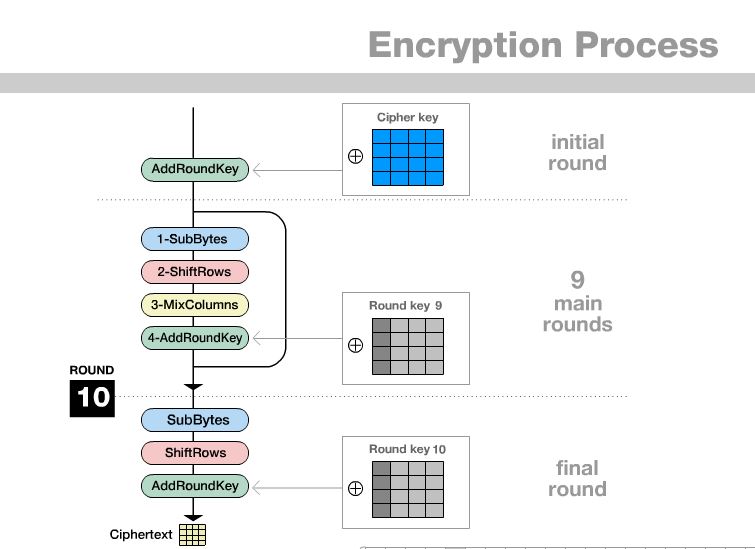
**

*Նկ.1*

Գաղտնագրման պրոցեսսը բաղկացած է երկու մասից` բաց հատվածի գաղտնագրում և բանալիների ստացում: Նախ դիտարկենք բաց հատվածի գաղտնագրման պրոցեսսը:



Նկ.2

State-ը համապատասխանում է բաց տեքստի հատվածին, Ciper Key-ն բանալին է:

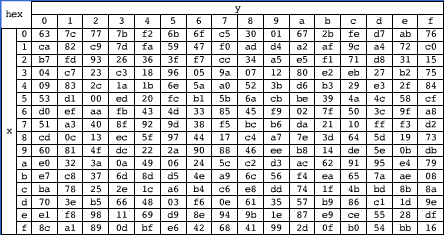
Նկ.3

Վերը ցուցադրված նկարից երևում է, որ գաղտնագրման պրոցեսը բաղկացած 9-ը հիմնական ր 1 վեևջին 10-րդ փուլերից:Բոլոր փուլերին համապատասխանում է տվյալ փուլի բանալին, որոնց ստացումը կդիտարկենք ավելի ուշ: Դիտարկենք ալգորիթմի բլոկերն առանձին-առանձին:

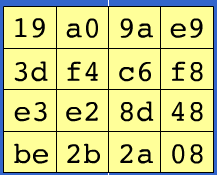
**1.5.1 Գաղտնագրում**

**SubBytes/Բայթերի փոխարինում**

Բայթերի փոխարինման պրոցեսսում օգտվում ենք նախապես հաշվարկված և բոլորին հայտնի Ս-բոքսից, որն իրենից ներկայացնում է 16\*16 երկչափ զանգված, որի ամեն մի անդամ իրենից ներկայացնում է 0 – 255 միջակայքի որևէ թիվ նկ. 3:

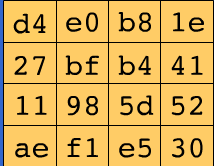


Նկ.4 Ս-բոքս



Նկ. 5 Բաց տեքստի բլոկ

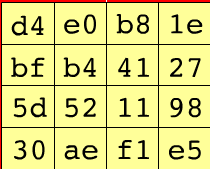
Հերթականությամբ բաց տեքստի ամեն բլոոկի ամեն բայթի համար դիտարկում ենք ավագ բայթը որպես տողի համար, իսկ կրտսերը` սյան: Այդ համարներին համապատասխան գտնում ենք նշված տողի և սյան հատման անդամը Ս-բոքսում և փոխարինում բաց տեքստի բայթը վերջինովս:



Նկ.6 Բայթերի փոխաինման արդյունք

**Shift Rows/Տողերի տեղաշարժ**

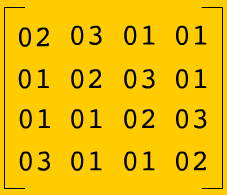
Նախորդ փուլի արդյունքում ստացված արդյունքի տողերը ցիկլիկ տեղաշարժում ենք դեպի ձախ համապատասխանաբար առաջին տողը` 0, երկրորդը` 1, երրորդը` 2, չորրորդը` 3 անգամ:



Նկ.7 Տողերի տեղաշարժի արդյունք

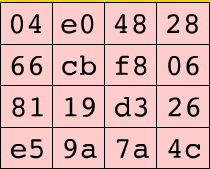
**Mix Columns/Սյուների խառնում**

Նախորդ փուլի արդյունքի յուրաքանչյուր սյուն բազմապատկվում է հետևյալ հաշվարկված և հայտնի մատրիցի հետ:



Նկ.8 Սյուների փոխանակման մատրից

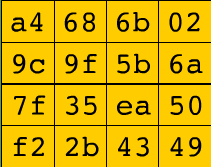
Արդյունքում ստանում ենք՝



Նկ.9 Սյուների խառնման արդյունք

**Add Round Key/Փուլի բանալիի գումարում**

Նախորդ փուլում ստացված արդյունքի յուրաքանչյուր սյուն ըստ մոդուլ 2-ի գումարվաում է տվյալ փուլի բանալիի հետ և արդյունքը գրվում է առաջին սյան տեղը:

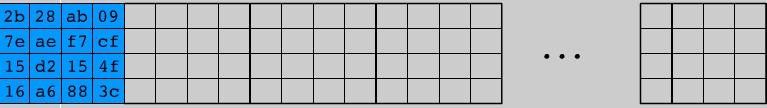


Նկ.10 Փուլի բանալու գումարման արդյունք

**1.5.2 Բանալիների ստացում**

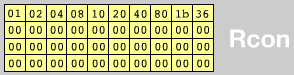
Այժմ դիտարկենք փուլերի բանալիների ստացման գործընթացը:

Մենք ունենք 128 բիթանոց բանալի, որը կօգտավործվի նախնական փուլում: Այժմ պետք է ստանալ մյուս 10 փուլերի բանալիները:



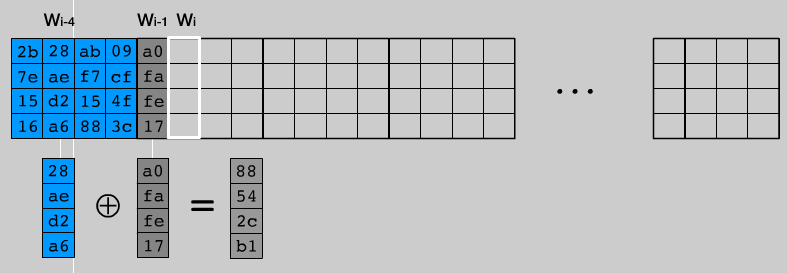
Պետք է ստանալ 10 փուլերի բանալիները

Բանալիների ստացման գործընթացում կկիրառենք նախապես հաշվարկված և բոլորին հայտնի RCon երկչափ զանգվածը:



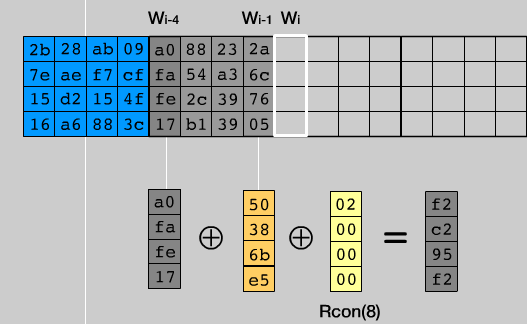
Նկ.11 RCon երկչափ զանգված

Նախ վերցվում է նախնական բանալու վերջին սյունը և ցիկլիկ տեղաշարժ կատարվում մեկ քայլ դեպի վերև: Ապա սյան հետ կատարվում է բայթերի փոխարինման գործողությունը: Ստացված սյանը ըստ մոդուլ 2-ի գումարվում է նրանից 3-ով ետ գտնվող սյունը և RCon ի հերթական սյունը, որը այնուհետև դուրս է հանվում: Արդյունքը գրվում է առաջին փուլի բանալու առաջին սյան դիրքում: Այնուհետև ստացված սյունը ըստ մոդուլ 2-ի գումարվում է նախորդ բանալու երկրորդ սյանը և ստացվածը գրվում է նոր բանալու երկրորդ սյան դիրքում:



Նկ.12 Առաջին փուլի բանալու ստացումը

Նույն պրոցեսսով ստացվում են նոր բանալու երրորդ և չորրորդ սյուները: Այնուհետև երկրորդ փուլի բանալին ստացվում է նույն գարծողությունները կատարելով, այս դեպքում արդեն առաջին փուլի բանալիի հետ:

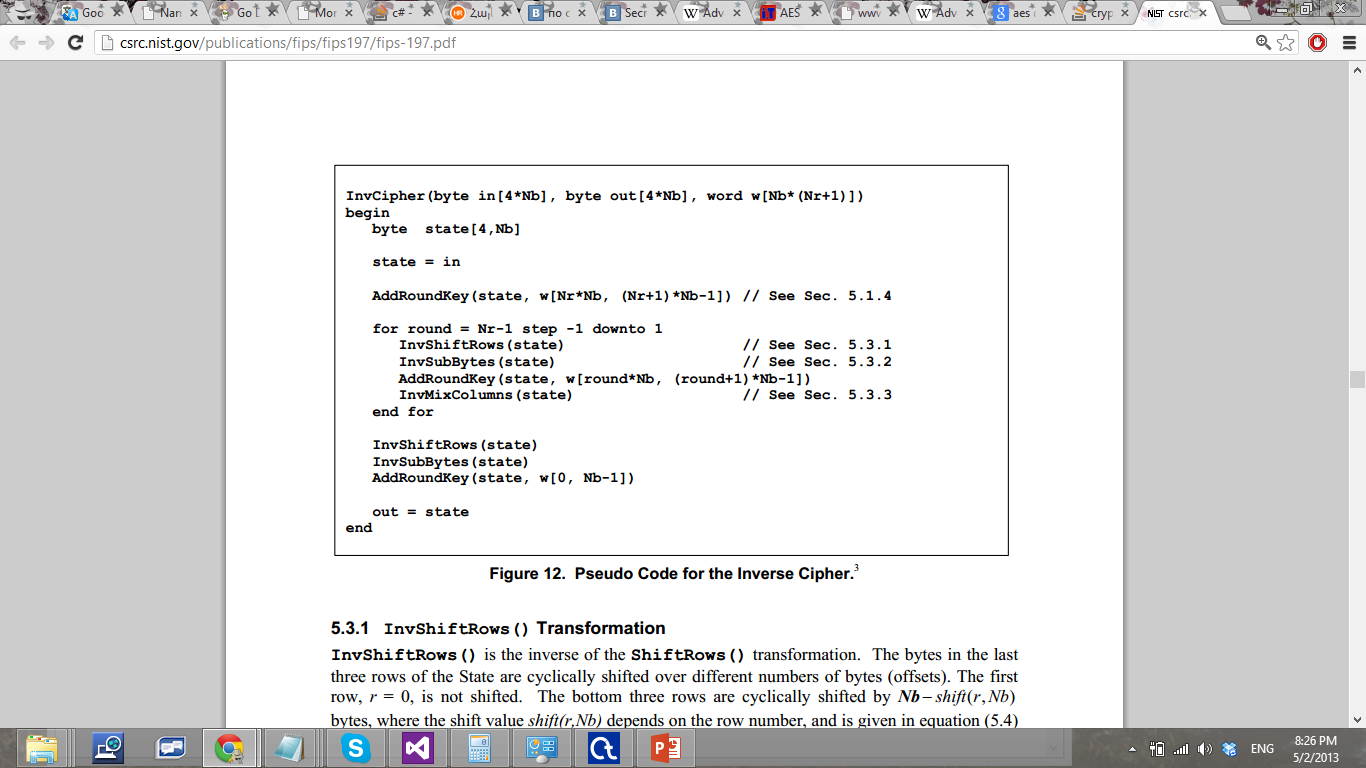


Նկ.13 Երկրորդ փուլի բանալու ստացումը

Արդյունքում ստանում ենք բոլոր 10 փուլերի բանալիները:

**1.5.3 Գաղտնավերծանում**

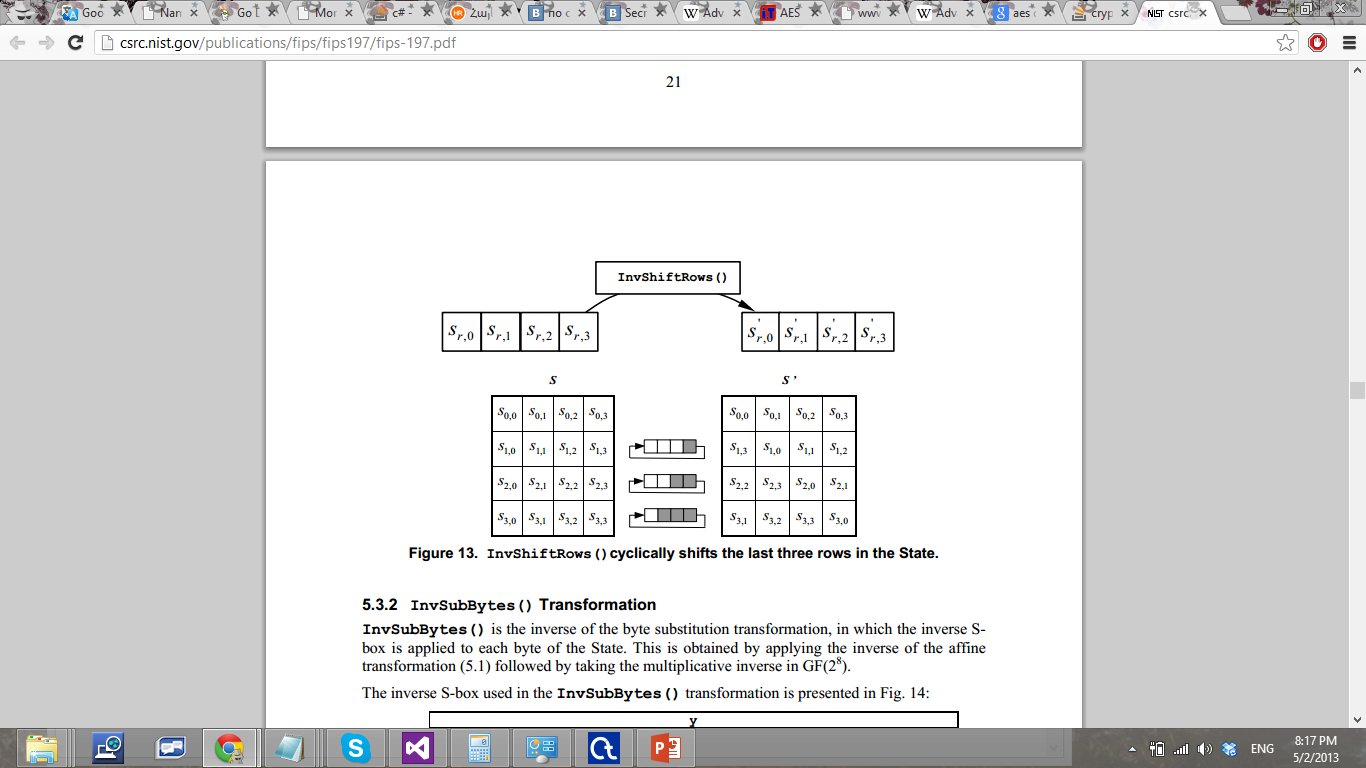
Գաղտնավերծանման պրոցեսսը կատարվում է կրկին նախնական փուլով , 9 միջանկյալ փուլերով և վերջնական փուլով և օգտագործվում են նույն փուլերի բանալիներն ինչ գաղտնագրման դեպքում էր: Սակայն փուլերի գործողություններն և հերթականությունն փոքր ինչ ուրիշ է:



Նկ.14 Գաղտնավերծանման ալգորիթմը

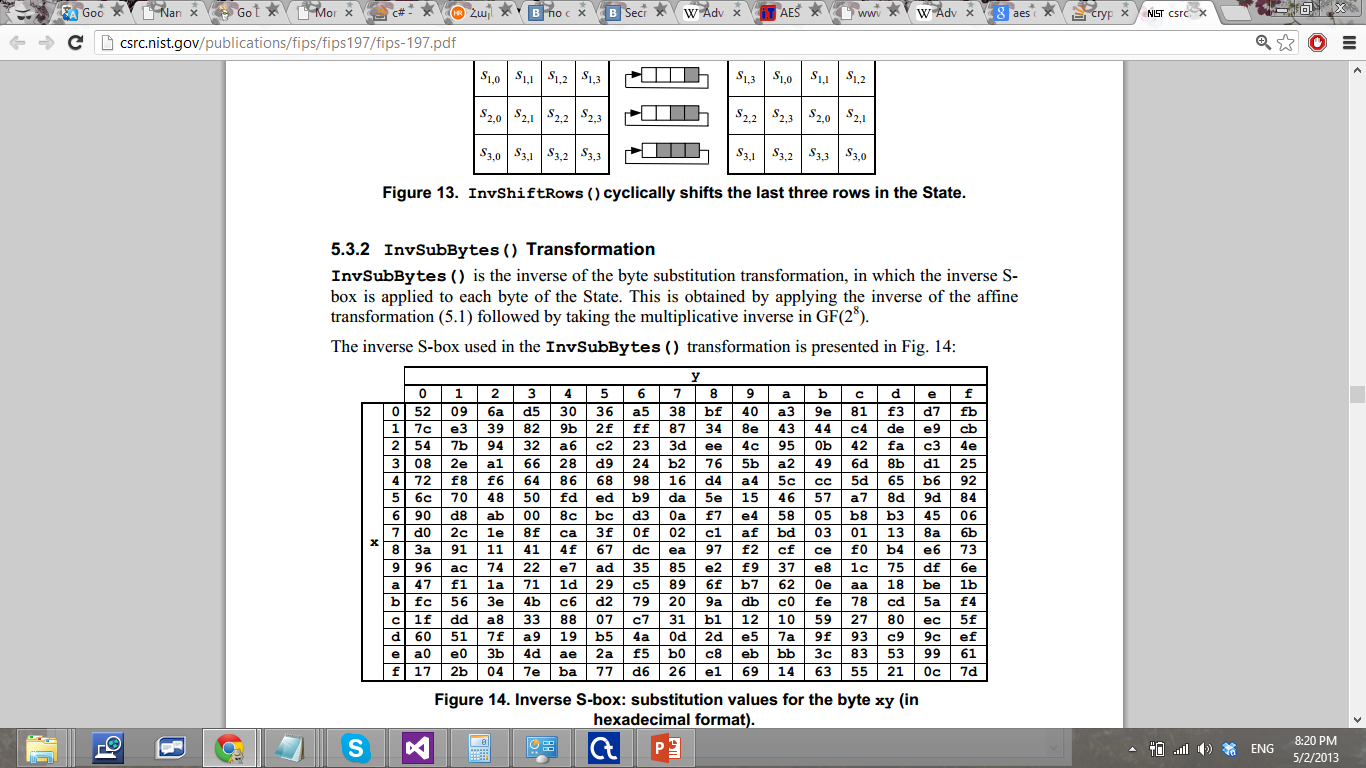
Սկզբում կատարվում է փուլի բանալու գումարում, որը ոչինչով չի տարբերվում նման փուլից գաղտնագրման դեպքում: Այնուհետև 9 փուլ կատարվում է տողերի ինվերս տեղաշարժ, ապա բայթերի ինվերս փոխարինում, ապա փուլի բանալու գումարում և սյուների ինվերս խառնում: Վերջին փուլում չի կատարվում սյուների ինվերս խառնում:

**Տողերի ինվերս տեղաշարժ**

Կատարվում է տողերի տեղաշարժ գործողությունը, բայց այս դեպքում դեպի աջ:

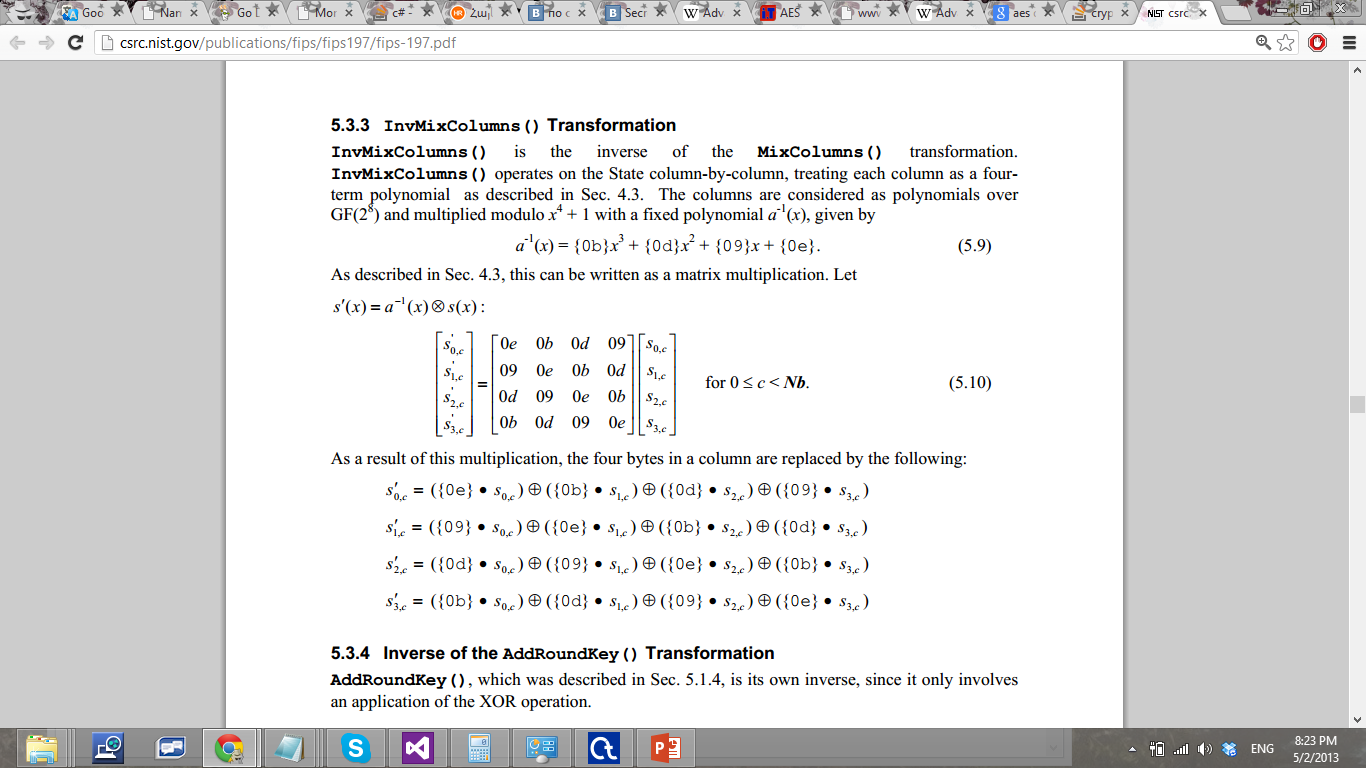
Նկ.15 Տողերի ինվերս տեղաշարժ

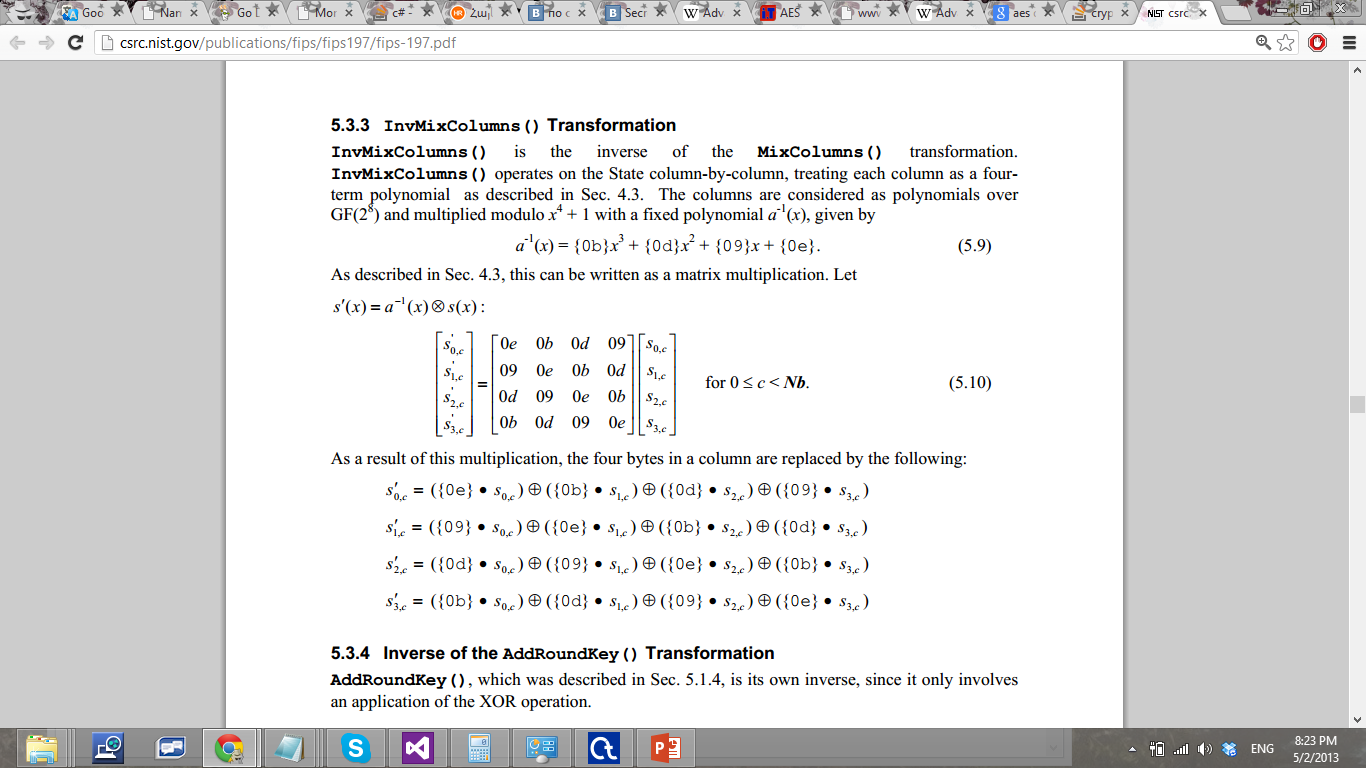
**Բայթերի ինվես փոխարինում**

Կատարվում է բայթերի փոխարինման գործողությունը օգտագործելով ինվերս Ս-բոքս: Նկ.16 Ինվերս Ս-բոքս

**Սյուների ինվերս խառնում**

Կատարվում է սյուների խառնման գործողությունը հաշվարկված ինվերս մատրիցի հետ:





Նկ.17 Սյուների ինվերս խառնում

**1.6 RC4 ալգորիթմի նկարագրությունը**

RC4 հոսքային գաղտնագիրը մշակվել է 1987թ.-ին Ռոն Ռիվեստի կողմից:

RC4 հոսքային գաղտնագիրը բաղկացած է 3 մասից.

1. Բանալու ստեղծում
2. Գաղտնագրման ալգորիթմ
3. Վերծանման ալգորիթմ

RC4 գաղտնագրային համակարգում բանալին իրենից ներկայացնում է կամայական երկարությամբ բայթերի հաջորդականություն, որի միջոցով ստեղծվում է S գաղտնագրի նախնական տեսքը- բոլոր 256 բայթերի վերադասավորումը:

RC4 ալգորիթմը, ինչպես և ցանկացած հոսքային գաղտնագիր, կառուցվում է պարամետրավորված բանալու գեներատորի հավասարաչափ բաշխված հաջորդական բիթերի հիման վրա: Բանալու երկարությունը կարող է լինել 40-2048 բիթ(8 քայլով):

Հոսքային գաղտնագրերի ալգորիթմի միջուկը բաղկացած է գամմա գեներատորից, որից ստացվում է հիմնական հոսքը(գամմա): Ֆունկցիան գեներացնում է բիթերի հաջորդականություն () ըստ մոդուլ 2-ի գումարման արդյունքում միավորվում է բաց տեքստի հետ (): Այսպիսով ստացվում է գաղտնագիրը(шифрограмма).

Վերծանումն իրականացվում է հիմնական հոսքի () և ըստ մոդուլ 2-ի լրացման գաղտնագրի() ռեգեներացիայով: Արդյունքում ստանում ենք սկբնական բաց տեքստը():

RC4-ը ալգորիթմների կլասս է, որը որոշվում է բլոկի չափով(S բլոկ):

n-ը դա բառի երկարությունն է, որը որոշում է S բլոկ–ի երկարությունը: Սովորաբար n=8, սակայն անվտանգության բարձրացման համար պետք է մեծացնել այն: n-ի մեծացման դեպքում(օրինակ մինչև 16 բիթ) S բլոկում էլեմենտների քանակը դառնում է 65536, և համապատասխանաբար սկզբնական ցիկլի ժամանակահատվածը մեծանում է: Սակայն մեծանում է գաղտնագրման արագությունը:

RC4-ի ներքին վիճակը ներկայացվում է տեսքի զանգվածի(S բլոկ) և 2 հաշվիչների տեսքով: S բլոկ-ը պարունակում է բառի հնարավոր արժեքների վերադասավորում:

RC4-ի իրականացումը բաղկացած է 2 մասից.

1. S բլոկի իրականացում
2. K պատահական բառի գեներացիա

**S բլոկի իրականացում**

Ալգորիթմը հայտնի է նաև ինչպես Key-Scheduling Algorithm(KSA):

Ի սկզբանե S-ը լրացվում է 0…255 հաջորդական արժեքներով: Որից հետո S-ի յուրաքանչյուր հաջորդ էլեմենտը տեղերով փոխվում է այն էլեմենտի հետ, որի համարը որոշվում է K բանալու էլեմենտով, այդ էլեմենտով և էլեմենտների համարների գումարով, որոնք վերադասավորվել են նախորդ ցիկլերում:

Քանի որ միայն մեկ գործողություն է կատարվում S-ի հետ, ապա պետք է հաստատվի, որ S-ը պարունակում է միևնույն արժեքների խումբը, որը տրվել էր սկզբնական իրականացման ժամանակ (S[i] := i)

**for** i **from** 0 **to** 255

S[i] := i

**endfor**

j := 0

**for** i **from** 0 **to** 255

j := (j + S[i] + Key[i mod L]) mod 256 // *n* = 8 ; 2*8* = 256

**endfor**