PARI MATCH

ASP. Net MVC

Back 2 school

TRADING TOOLS

AGENDA

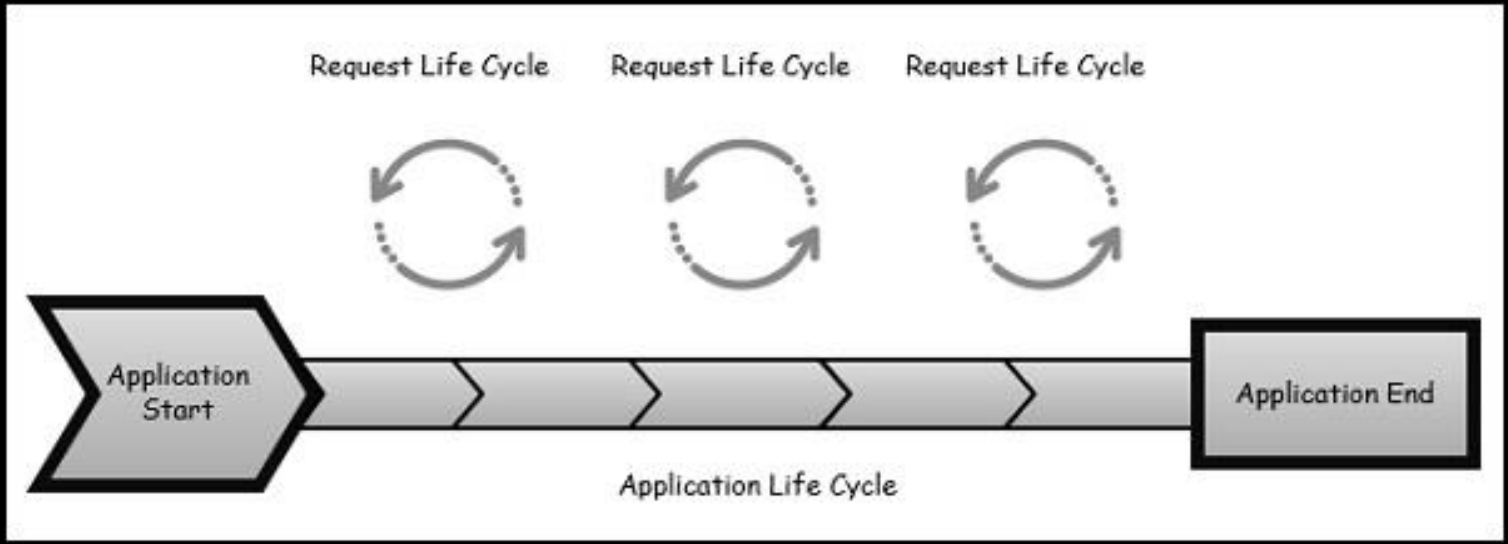
- Application Life Cycle
- Http Module
- Http Handler
- Adaptive Layouts
- Input/Output Formatters
- TypeConverter
- Tracing
- Output cache
- Security
  - Hashing
  - Symmetric/Assymetric encryption
  - User Secrets/Azure Key Vault

PARI MATCH

APPLICATION LIFE CYCLE

TRADING TOOLS

APPLICATION LIFE CYCLE



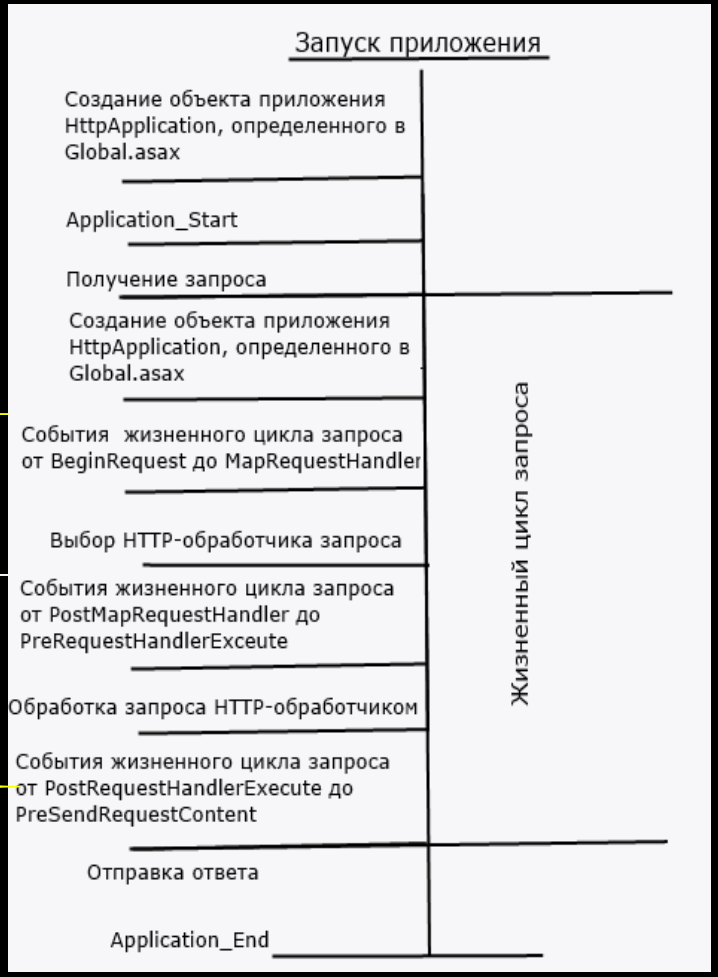
TRADING TOOLS

PARI MATCH

APPLICATION LIFE CYCLE

1. BeginRequest: когда приложение получает новый запрос
2. AuthenticateRequest/PostAuthenticateRequest: идентификация (аутентификация) пользователя
3. AuthorizeRequest/PostAuthorizeRequest: авторизация
4. ResolveRequestCache/PostResolveRequestCache: при получении данных из кэша
5. MapRequestHandler/PostMapRequestHandler: определении обработчика запроса
6. AquireRequestState/PostAquireRequestState: при получении данных состояния, связанных с запросом (например, данные сессии)
7. PreRequestHandlerExecute/PostRequestHandlerExecute: перед/после работы обработчика запроса
8. ReleaseRequestState/PostReleaseRequestState: перед/после освобождения данных, ассоциированные с запросом
9. UpdateRequestCache: обновлении данных в кэше
10. LogRequest/PostLogRequest: перед/после каждым логгированием
11. EndRequest: когда данные для ответа уже готовы к отправке клиенту
12. PreSendRequestHeaders: перед отправкой HTTP-заголовков браузеру клиента
13. PreSendRequestContent: после отправки заголовков, но перед отправкой основного содержимого ответа

Error

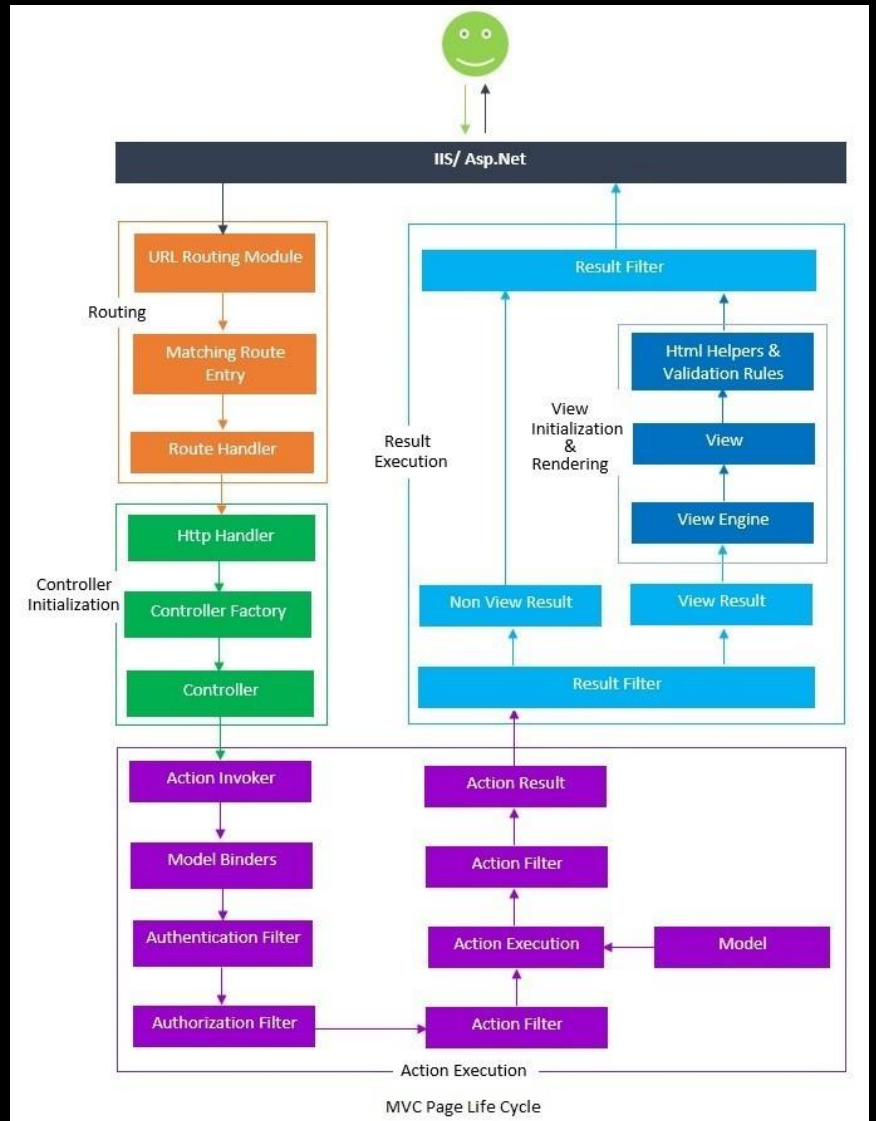


PARI MATCH

APP LIFECYCLE DEMO

TRADING TOOLS

Request Life cycle



FILTERS

Тип фильтров	Реализуемый интерфейс	Стандартная реализация	Описание
Фильтры аутентификации	IAuthenticationFilter	-	Фильтр, определяющий, аутентифицирован ли клиент. Данный фильтр запускается до выполнения любого другого фильтра или метода действий
Фильтры авторизации	IAuthorizationFilter	AuthorizeAttribute	Фильтр, определяющий, имеет ли пользователь доступ к данному ресурсу. Данный фильтр запускается после фильтра аутентификации, но до любого другого фильтра или метода действия
Фильтры действий	IActionFilter	ActionFilterAttribute	Фильтр, применяемый к действиям. Может запускаеться как до, так и после выполнения метода действий
Фильтры результатов действий	IResultFilter	ActionFilterAttribute	Фильтр, применяемый к результатам действий. Может запускаться как до, так и после выполнения результата действия
Фильтры исключений	IExceptionFilter	HandleErrorAttribute	Атрибут для обработки исключений, выбрасываемых методом действий и результатом действий

HTTP Module

HTTP модули выполняются до и после обработки и обеспечивают метод для взаимодействия с запросом. Пользовательские модули должны быть унаследованы от System.Web.IHttpModule интерфейса.

```
void Init(HttpApplication context);
void Dispose();
```

Модули, как правило, синхронизированы с событиями System.Web.IHttpModule класса в рамках Global.asax.cs

Зарегать модуль нужно к web.config

```
<system.webServer>
 <modules>
  <add name="Timer" type="LifeCycleApp.Modules.TimerModule"/>
 </modules>
</system.webServer>
```

1. BeginRequest
2. AuthenticateRequest
3. AuthorizeRequest
4. ResolveRequestCache
5. AcquireRequestState
6. PreRequestHandlerExecute
7. PostRequestHandlerExecute
8. ReleaseRequestState
9. UpdateRequestCache
10. EndRequest
11. PreSendRequestHeaders *
12. PreSendRequestContent *

Error *

* могут произойти в любое время в течение запроса, все остальные перечислены в порядке их вызова.

PARI MATCH

HTTP MODULE DEMO

TRADING TOOLS

HTTP Handler

Используются для генерации ответа на HTTP-запрос. При обработке одного запроса мы можем задействовать несколько различных модулей, но только один HTTP Handler. Пользовательские хендлеры должны быть унаследованы от System.Web.IHttpHandler интерфейса.

```
void ProcessRequest(HttpContext context);
bool IsReusable { get; }
```

HTTP Handler, как правило, регистрируются при формировании RouteConfig в качестве обработчика к маршруту. Провайдером для хендлера является IRoteHandler

Зарегать хендлер нужно к web.config

```
<system.webServer>
 <handlers>
  <add name="MyHttpHandler" path="/handler/" verb="GET"
type="LifeCycleApp.Handlers.MyHttpHandler"/>
 </handlers>
</system.webServer>
```

1. BeginRequest
2. AuthenticateRequest
3. AuthorizeRequest
4. ResolveRequestCache

--- MapRequestHandler (selecting handler) ---

5. AcquireRequestState
6. PreRequestHandlerExecute

--- executes HTTP Handler ---

5. PostRequestHandlerExecute
6. ReleaseRequestState
7. UpdateRequestCache
8. EndRequest
9. PreSendRequestHeaders *
10. PreSendRequestContent *

Error *

Мы также можем поставить любой тип/глагол/verb запроса - в этом случае используется звездочка *

PARI MATCH

HTTP HANDLER DEMO

TRADING TOOLS

Handler vs Module

Modules

1 запрос может быть обработан несколькими модулями
Создан для изменения или передачи запроса остальным модулям
Реализует IHttpModule интерфейс
Создан что бы быть интегрированными с любимы событием из жизненного цикла

Handlers

Только 1 хендлер может обработать запрос
Отвечает за формирует ответ и возвращает
Реализует IHttpHandler интерфейс
Связан с событиями выбора хенделра и обработки запроса

Регистрируется в коде или в web.config файле

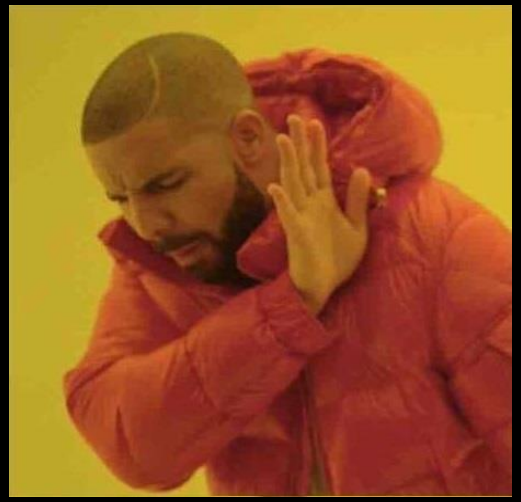
PARI MATCH

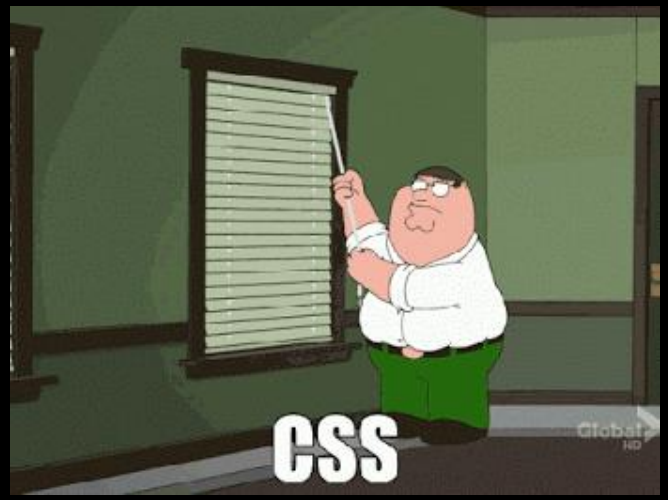
ADAPTIVE LAYOUTS

TRADING TOOLS

Как быть если менеджер хочет красивый вид на мобильных девайсах?

Adaptive Layouts





1. Ничего не делать

2. ~~Трахаться с~~ Настроить CSS

3. Сделать отдельный лейаут и оптимизировать передачу данных*

* есть компании в которых каждый вторник скорость сети ограничена до уровня 2G

ADAPTIVE LAYOUTS DEMO

TRADING TOOLS

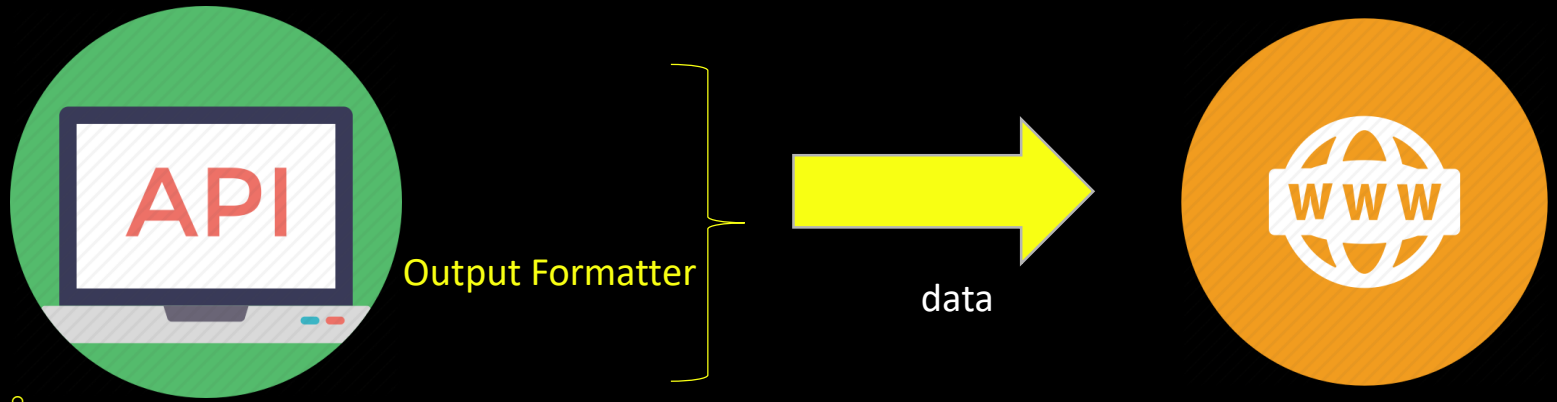
INPUT/OUTPUT FORMATTERS

TRADING TOOLS

Formatters

1. Когда нам шлют ~~говно~~ сообщения в устарелом формате

2. Когда мы шлем ~~говно~~ сообщения в устарелом формате



INPUT/OUTPUT FORMATTERS DEMO

TRADING TOOLS

TRACING

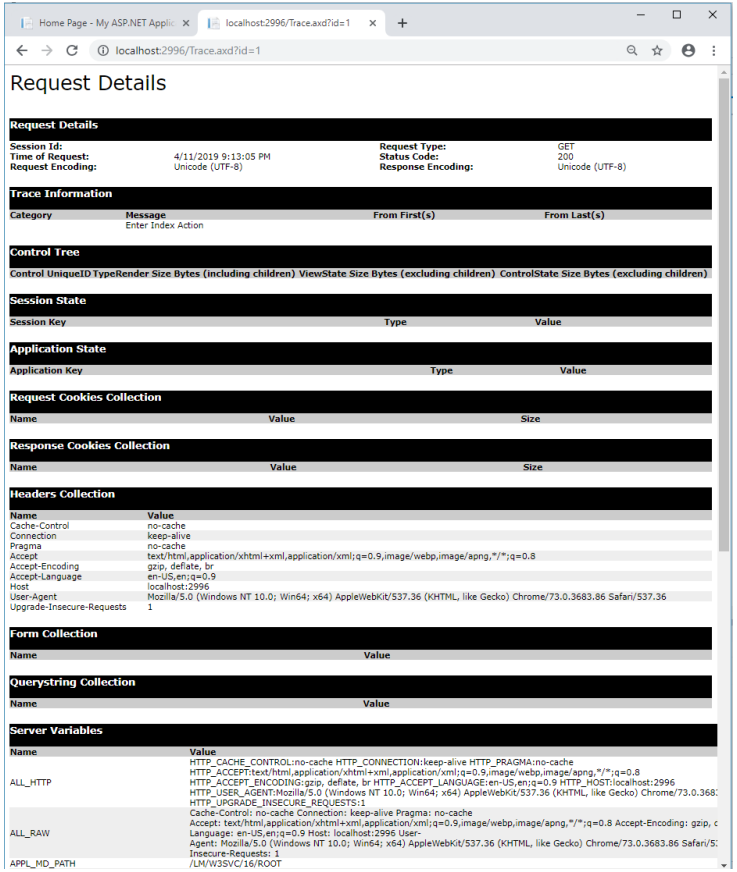
TRADING TOOLS

Step 1. Add Trace section at system.web

```
<system.web>
<trace enabled="true" requestLimit="50“
mostRecent="true" localOnly="true" writeToDiagnosticsTrace="true“
pageOutput="true" />
    <compilation debug="true" targetFramework="4.7.1"/>
    <httpRuntime targetFramework="4.7.1"/>
  </system.web>
```

2. Add Listener

```
<system.diagnostics>
    <trace>
      <listeners>
        <add name="WebPageTraceListener“
type="System.Web.WebPageTraceListener, System.Web,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"/>
      </listeners>
    </trace>
  </system.diagnostics>
```



Step 3. Usage

```
this.HttpContext.Trace.Write("Enter Index Action");
```

PARI MATCH

TRACING DEMO

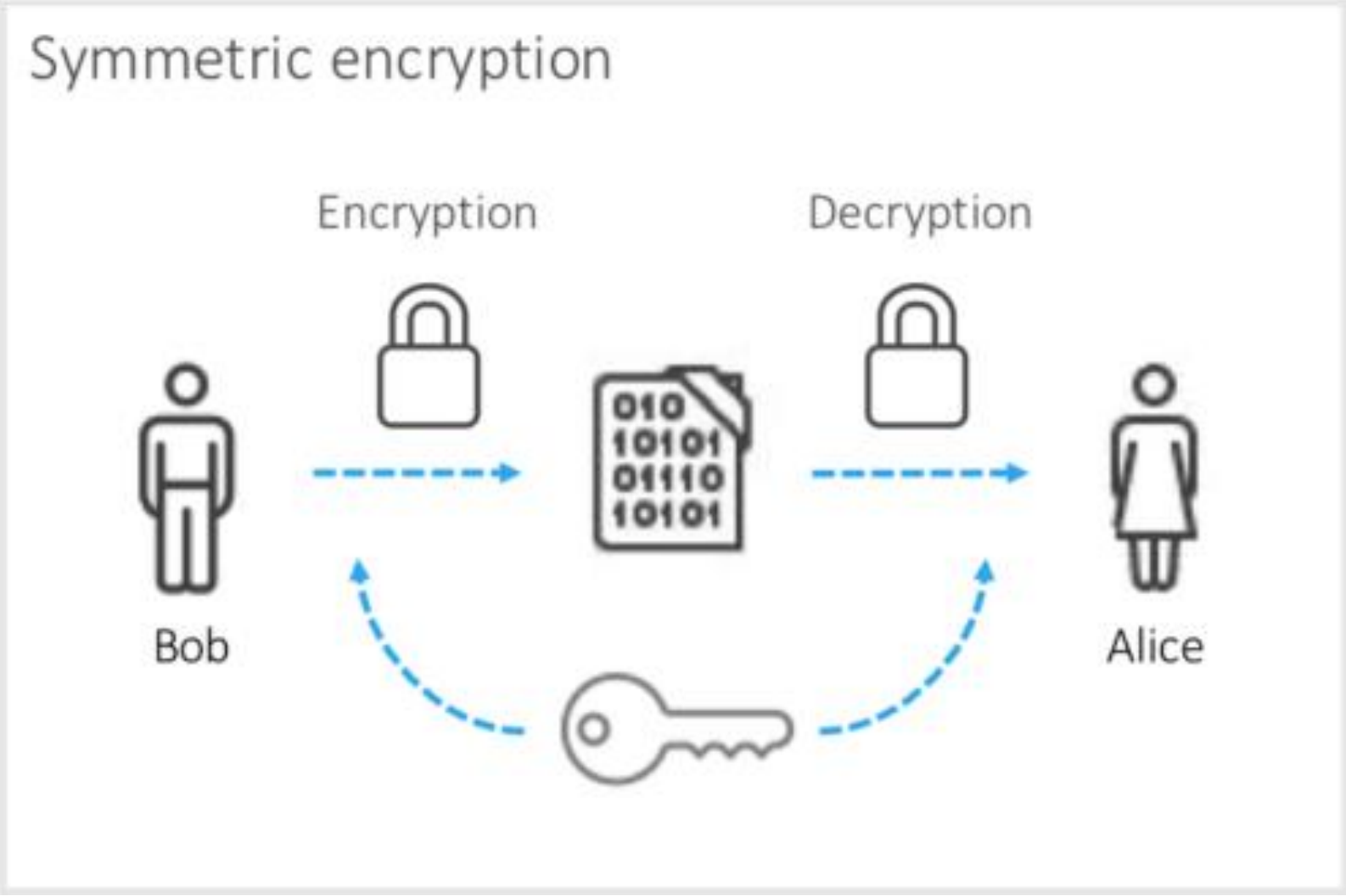
TRADING TOOLS

SECURITY

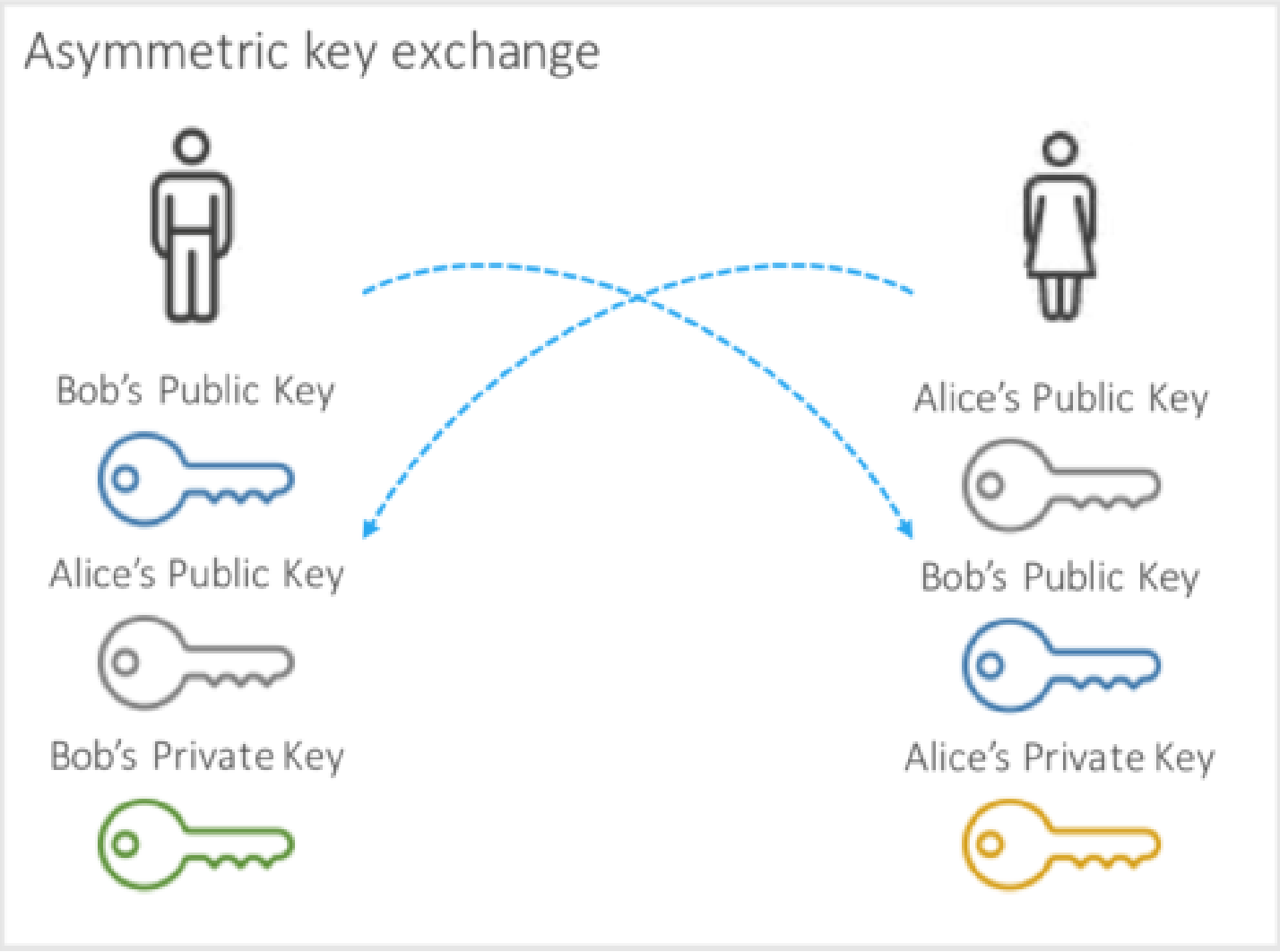
TRADING TOOLS

Encryption

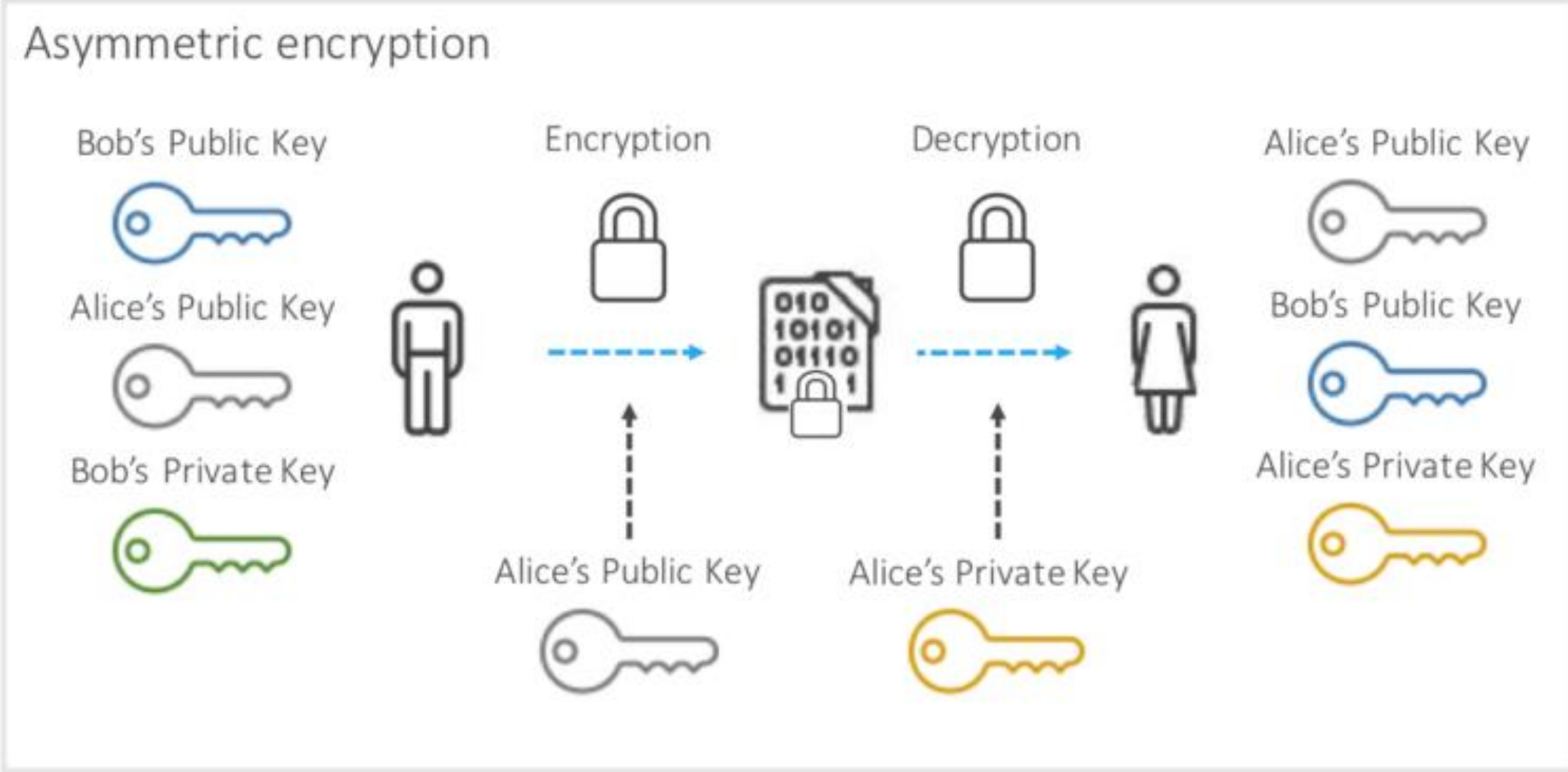
Algorithm namespace	Type of algorithm
System.Security.Cryptography.Aes	Symmetric
System.Security.Cryptography.DES	
System.Security.Cryptography.RC2	
System.Security.Cryptography.Rijndael	
System.Security.Cryptography.TripleDES	
System.Security.Cryptography.DSA	Asymmetric
System.Security.Cryptography.ECDiffieHellman	
System.Security.Cryptography.ECDsa	
System.Security.Cryptography.RSA	


Symmetric encryption
Encryption
Decryption
010
10101
01110
10101
Bob
Alice

PARI MATCH


Asymmetric key exchange
Bob's Public Key
Alice's Public Key
Bob's Private Key
Alice's Public Key
Bob's Public Key
Alice's Private Key

PARI MATCH


Asymmetric encryption
Bob's Public Key
Alice's Public Key
Bob's Private Key
Encryption
Alice's Public Key
010
10101
01110
Decryption
Alice's Private Key
Alice's Public Key
Bob's Public Key
Alice's Private Key

PARI MATCH

Encryption

Symmetric Example

```
using (RijndaelManaged rjdM = new RijndaelManaged()) {
        CryptoStream crypoStream = new CryptoStream(myManagedStream,
                                        rjdM.CreateEncryptor(), CryptoStreamMode.Write);
};

using (RijndaelManaged rjdM = new RijndaelManaged()) {
        CryptoStream crypoStream = new CryptoStream(myManagedStream,
                                        rjdM.CreateEncryptor(),CryptoStreamMode.Read);
};
```

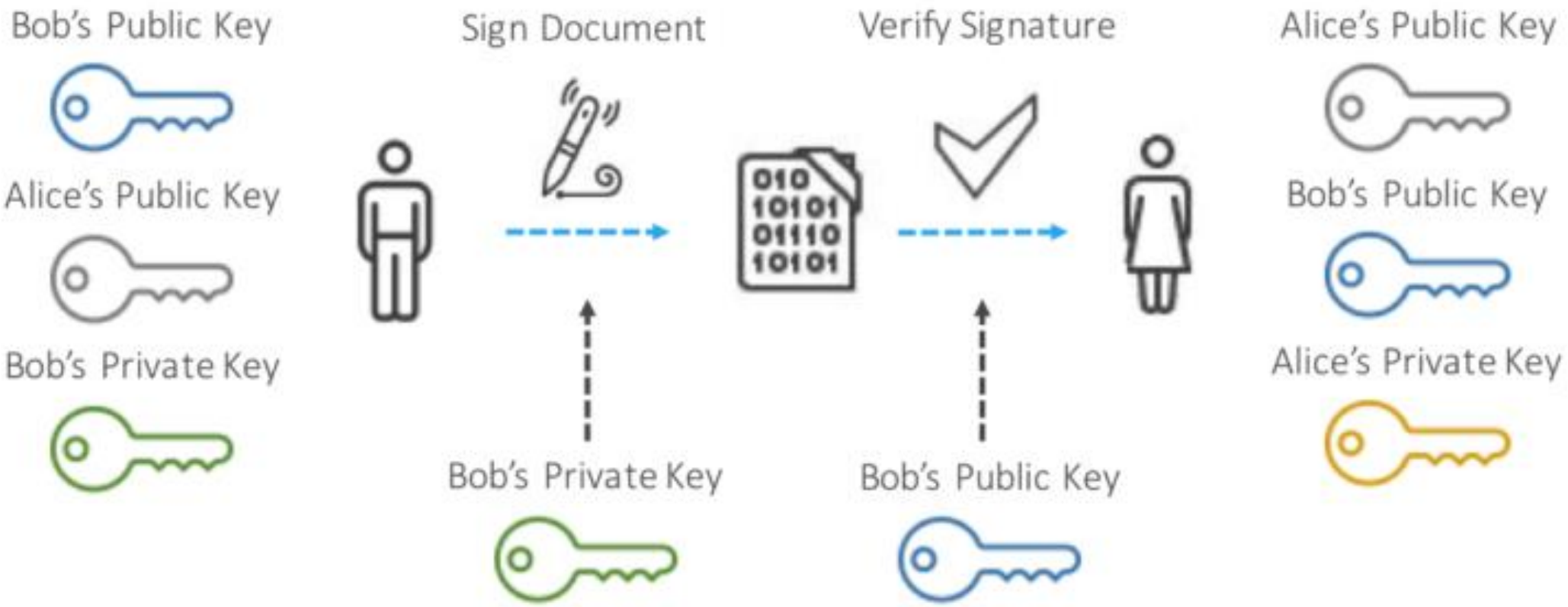
Asymmetric Example

```
using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider()) {
        RSA.ImportParameters(RSAKeyInfo);
        encryptedData = RSA.Encrypt(DataToEncrypt, DoOAEPPadding);
        decryptedData = RSA.Decrypt(encryptedData, DoOAEPPadding);
}
```

RSAKeyInfo is of type RSAParameters and contains the public key
DoOAEPPadding is a Boolean value that should be set to true to perform direct RSA encryption using Optimal Asymmetric Encryption Padding (OAEP).

PARI MATCH

Signing with asymmetric encryption



PARI MATCH

Signing

Bob's Example

```
// create the hash code of the text to sign
SHA1 sha = SHA1.Create();
byte[] hashcode = sha.ComputeHash(TextToConvert);

// use the CreateSignature method to sign the data
DSA dsa = DSA.Create();
byte[] signature = dsa.CreateSignature(hashcode);
```

Alice's Example

```
// create the hash code of the text to verify
SHA1 sha = SHA1.Create();
byte[] hashcode = sha.ComputeHash(TextToVerify);

// use the VerifySignature method to verify the DSA signature
DSA dsa = DSA.Create();
bool isSignatureValid = dsa.VerifySignature(hashcode, signature);
```

There are some additional rules that should be followed for both symmetric and asymmetric encryption:

Use unique keys Rather than using a single key for everything being encrypted in your application, choose different keys for different business functions. Doing so will complicate the effort of anyone trying to decrypt the information.

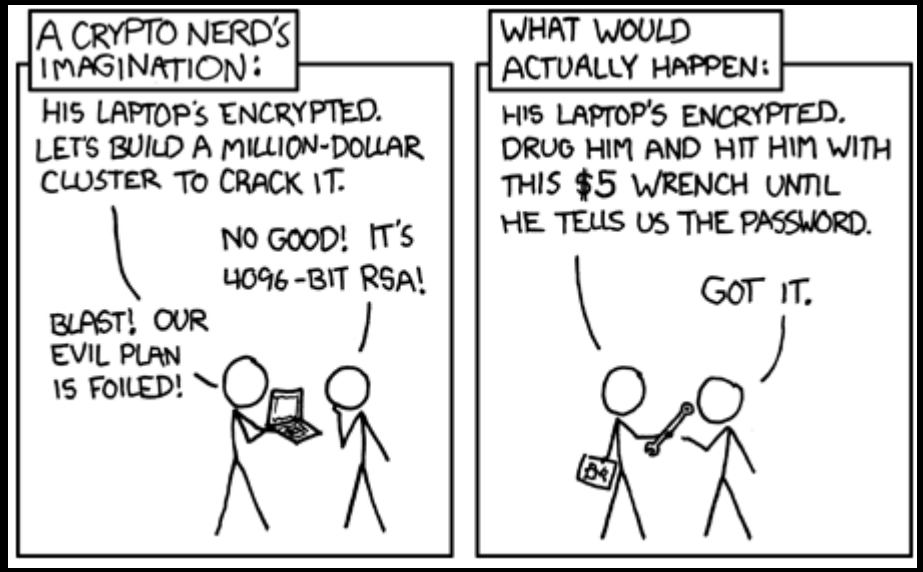
Protect your keys All secure data will lose its protection if the key is released to the public.

Ensure that your keys are not with your data Store keys in a separate location from your data. This makes it more difficult for hackers to locate both keys and data and to crack your system.

Configure keys to expire You should have rules and processes in place to generate, store, replace, use, distribute, update, revoke, and expire your tokens. When performing any of these operations, consider the effort of decrypting and re-encrypting information with the new key during the transition period.

PARI MATCH

Encrypt Web.config



Step 1. Go 2 place where magic lives

cd c:\Windows\Microsoft.NET\Framework64\v4.0.30319\

Step 2. Ask CryptoProvider 2 do all the magic

Encrypting

aspnet_regiis
-pef "appSettings“
"C:\ %some_path_to_your_website_dir%”
-prov
"RsaProtectedConfigurationProvider“

Decrypting

aspnet_regiis
-pef "appSettings“
"C:\ %some_path_to_your_website_dir% “
-prov
"RsaProtectedConfigurationProvider"

DPAPIProtectedConfigurationProvider This provider uses the Windows Data Protection API (DPAPI)
RsaProtectedConfigurationProvider This provider uses the RSA encryption algorithm

UserSecrets

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp2.1</TargetFramework>
    <UserSecretsId>6f0103f9-c6a5-4e71-bc4a-
213c7cb60bf7</UserSecretsId>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.App" />
  </ItemGroup>
</Project>
```

Сама инфа лежит здесь:
%APPDATA%\Microsoft\UserSecrets\<user_secrets_id>\secrets.json

От PROD’а у нас нету секретов, но лучше хранить все в AzureKeyVault или AWS Secrets Manager

SECURITY DEMO

TRADING TOOLS

Q&A

TRADING TOOLS