Understanding the Data Protection API and the Secret Manager

Roland Guijt
CONSULTANT | TRAINER | AUTHOR

@rolandguijt www.rmgsolutions.nl

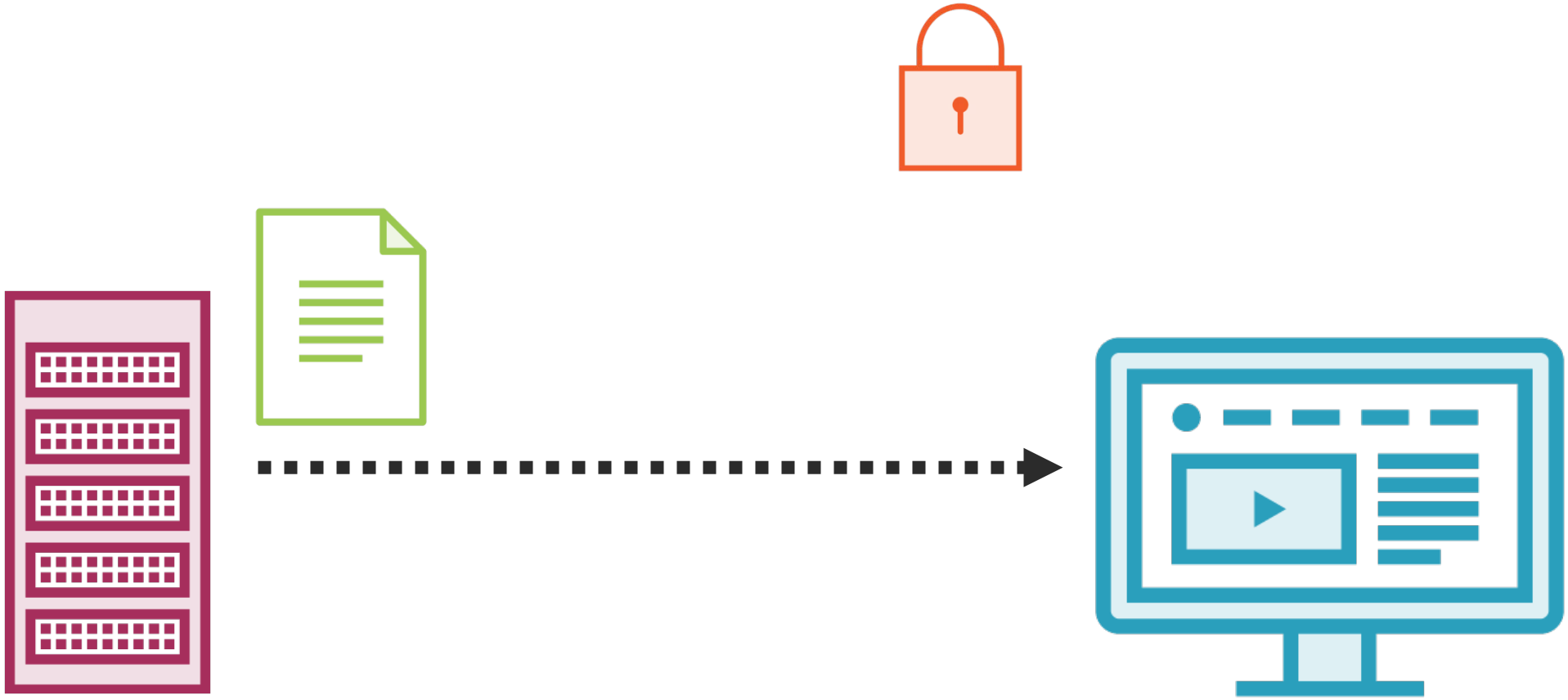
Overview

Machine.config encryption

Data protection API encryption

Protecting secrets at development time

Cookie Encryption



Symmetrical Encryption

Makes data unreadable using algorithm

Encrypt: key needed

Decrypt: same key needed

Hashing

Anti tampering

Encrypted data is encrypted using one-way algorithm

Upon return hashed data must match previously hashed data

Machine Key

```
<configuration>
  <system.web>
    <machineKey decryptionKey="Decryption key goes here"
                validationKey="Validation key goes here" />
  </system.web>
</configuration>
```

Machine Key Architecture: Disadvantages

- Hard to sync in web farm scenarios
- No key rotation
- No key protection
- Attacker can do disastrous things when stolen

What Is the Data Protection API?

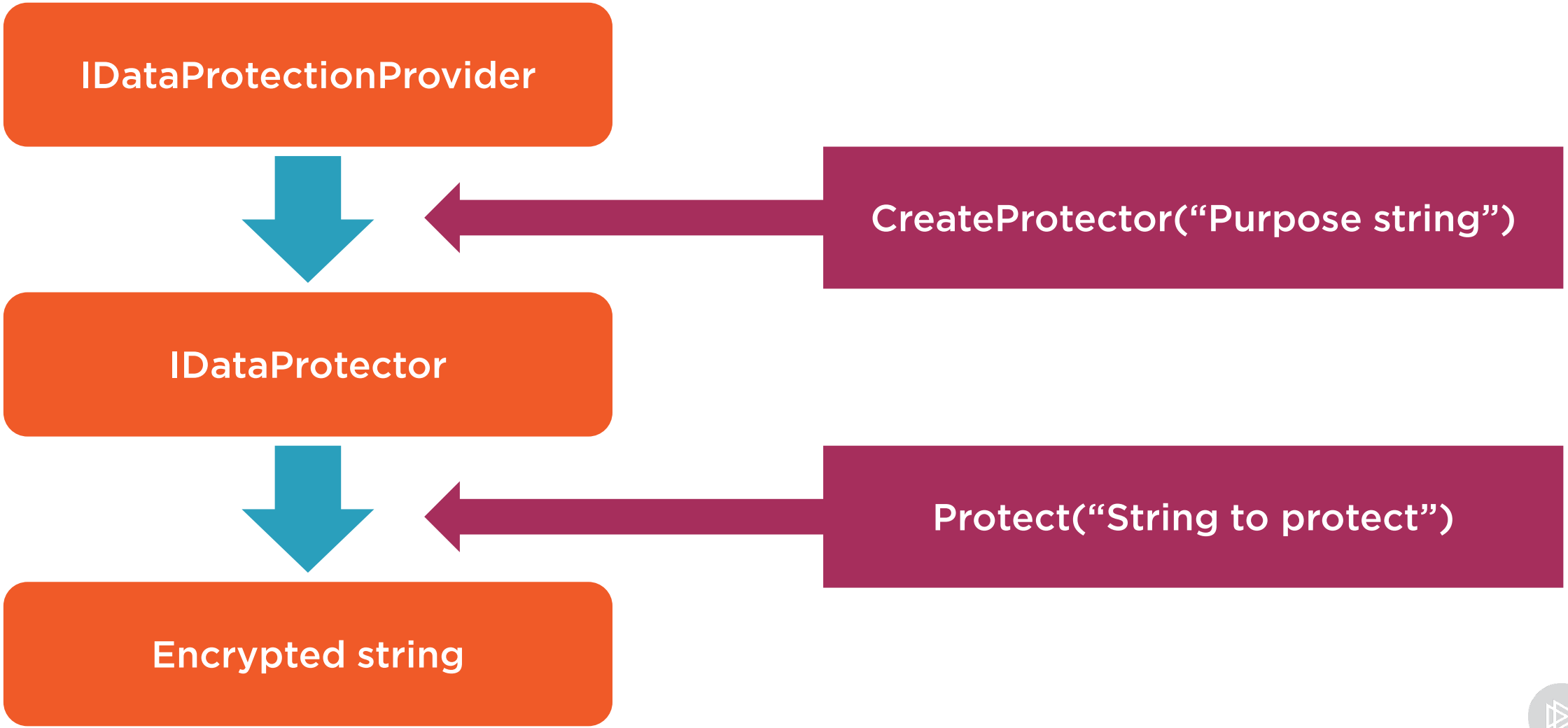
- Replacement for machine key
- Keys are protected
- Keys per application, per purpose
- Key rotation
- Relies on default settings

Data Protection API: Scenarios

Cookie storage

Anti forgery token

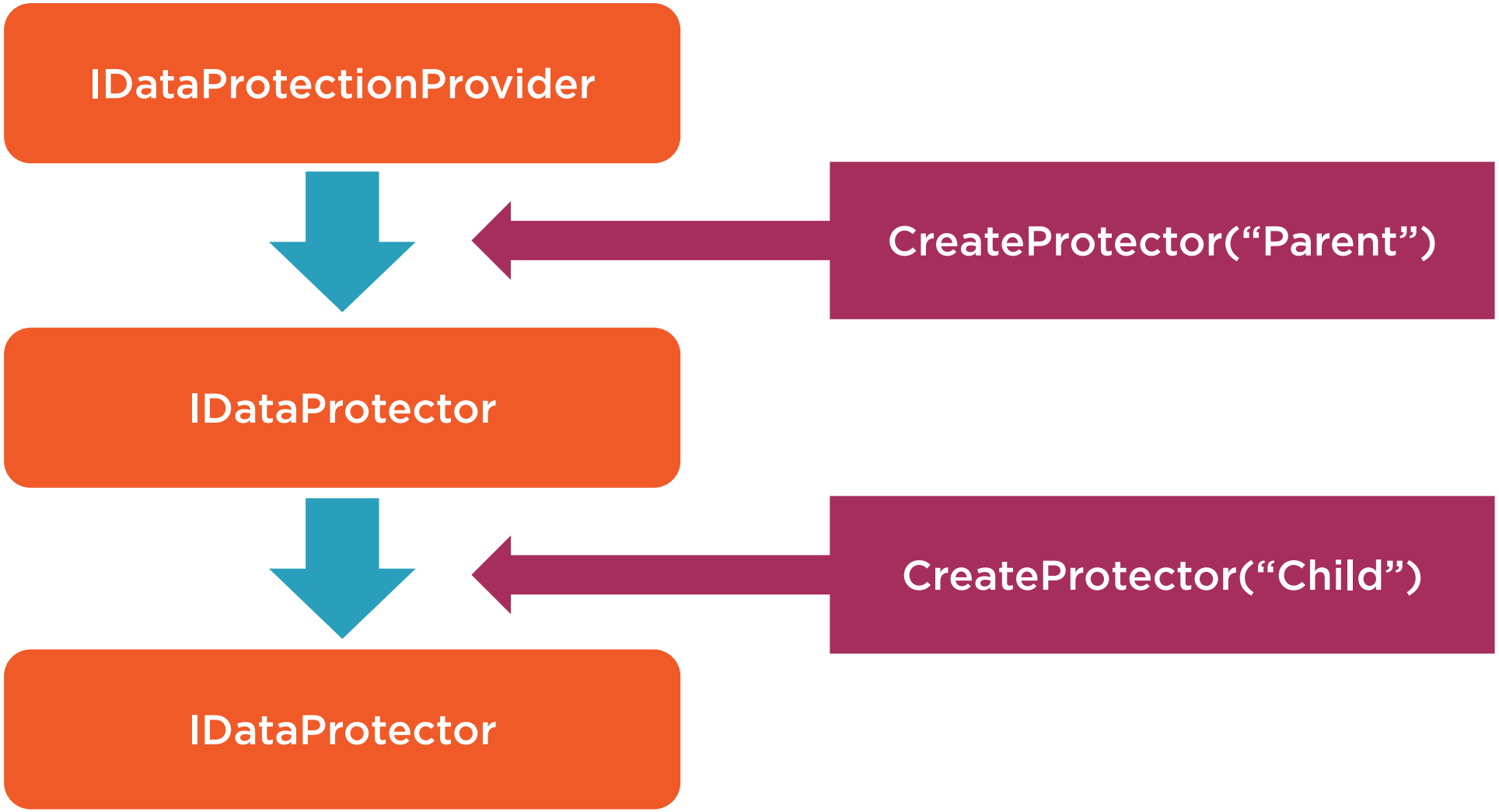
Data Protection API Basics



Master Keys and Purpose Strings

Master key + Purpose string = Used key

Purpose Hierarchy



Chaining Data Protectors

CreateProtector("Parent", "Child")

Versioning

```
CreateProtector("Demo.WebApp", "v1")
```

Versioning

```
CreateProtector("Demo.WebApp", "v2")
```

Multi-tenancy

IDataProtectorProvider
(application purpose string)

CreateProtector(”Tenant1”)

CreateProtector(”QueryString”)

CreateProtector(”Cookie”)

CreateProtector(”Tenant2”)

CreateProtector(”QueryString”)

CreateProtector(”Cookie”)

Master Keys

User profile available

- Local app data folder + DPAPI

IIS

- Registry + DPAPI

Azure

- Data-Protection-Keys folder

Other

- No key persistance

Time Limiting Protected Data

Protected data expires

Microsoft.AspNetCore.DataProtection.Extensions

protector.ToTimeLimitedDataProtector()

Protect(string, DateTime)

Protect(string, TimeSpan)

You Don’t Want

Secrets in source code

Secrets in Source control

Use production secrets in development/test

Options

Use environment variables

Use the secret manager

The Configuration Builder

```
var builder = new ConfigurationBuilder()
    .SetBasePath(env.ContentRootPath)
    .AddJsonFile("appsettings.json")
    .AddJsonFile($"appsettings.{env.EnvironmentName}.json",
       optional: true);
var configuration = builder.Build();
var conString = configuration([“DefaultConnection”]);
```

Adding Environment Variables

```
var builder = new ConfigurationBuilder()
    .SetBasePath(env.ContentRootPath)
    .AddJsonFile("appsettings.json")
    .AddJsonFile($"appsettings.{env.EnvironmentName}.json",
       optional: true);
    .AddEnvironmentVariables();
```

Secret Manager

A DotNet CLI extension verb

Adds secrets to json file in user profile folder (not encrypted!)

App needs UserSecretsId

Development purposes only

Add secrets to ConfigurationBuilder by adding AddUserSecrets()

Summary

Machine.config based encryption is gone

Data protection API very sophisticated but easy to use

Use secret manager to use secrets in development scenarios