VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÉ APLIKACE PRO AUTENTIZACI UŽIVATELŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

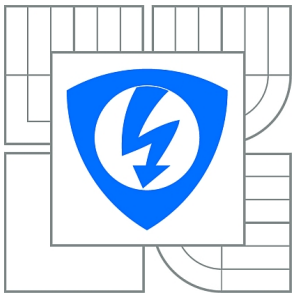
AUTOR PRÁCE
AUTHOR

Bc. PETR VYBÍRAL

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÉ APLIKACE PRO AUTENTIZACI UŽIVATELŮ

WEB APPLICATIONS FOR USER AUTHENTICATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

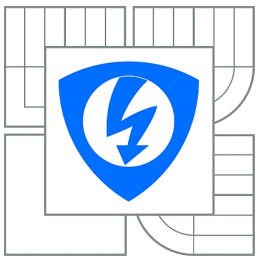
AUTOR PRÁCE — Bc. PETR VYBÍRAL

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN HAJNÝ, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Petr Vybíral	***ID:***	125341
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Webové aplikace pro autentizaci uživatelů

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte jednoduchý webový portál, který bude obsahovat jak volně dostupné, tak uzavřené sekce. Pro přístup k uzavřeným sekcím implementujte jednoduché řízení přístupu pomocí jména a hesla. Využijte hashovací funkce. Dále implementujte ověření pomocí atributové autentizace. Pro atributovou autentizaci využijte tokeny uložené na klientském PC. Analyzujte možnosti využití cookies pro ověření klienta. Rozšiřte možnost ověření o autentizaci pomocí QR kódu a/nebo čipové karty.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and network security: principles and practice. Seventh edition. xix, 731 pages. ISBN 01-333-5469-5.

[2] HAJNÝ, J.; MALINA, L. Unlinkable Attribute-Based Credentials with Practical Revocation on Smart-Cards. In Smart Card Research and Advanced Applications. Lecture Notes in Computer Science. LNCS. Berlin: Springer- Verlag, 2013. s. 62-76. ISBN: 978-3-642-37287- 2. ISSN: 0302- 9743.

Termín zadání:	10.2.2014	***Termín odevzdání:***	28.5.2014

Vedoucí práce: Ing. Jan Hajný, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce se věnuje problematice autentizace uživatelů. V první kapitole je rozebrán problém autentizace, její metody a využití. Druhá kapitola představuje jednotlivé možnosti zabezpečení komunikace. Popisuje zabezpečovací, komunikační a autentizační protokoly. Na závěr kapitoly je popsán čárový 2D QR kód. Třetí kapitola se věnuje technologii ASP.NET, jejímu vývoji a možnosti využití. Pozornost je zaměřena na webové formuláře a serverové ovládací prvky. Poté jsou analyzovány prvky cookies a jejich možnost využití. Poslední kapitolu tvoří praktická část, ve které je popsán vývoj webové aplikace. Následuje popis částí aplikace, jako je např. databáze, navigace po webu, vzorové stránky a další. Stěžejní část kapitoly tvoří rozbor a implementace formulářové autentizace, atributové autentizace a autentizace pomocí QR kódu. Následně je popsán způsob zabezpečení komunikace pomocí certifikátu.

KLÍČOVÁ SLOVA

autentizace, ASP.NET, cookies, Diffie-Hellman, kryptografie, SHA, socket, HTTPS, HM12, QR kód, webkamera, ZXING

ABSTRACT

The thesis deals with the problems of user authentication. The first chapter analyzes the problem of authentication, its methods and its utilization. The second chapter presents the different security options for communication. The chapter describes security, communication and authentication protocols. There is the 2D barcode QR Code described at the end of the chapter. The third chapter is devoted to ASP.NET technology, its development and possibilities of utilization. Attention is focused on web form and server controls. There is an analysis of elements of cookies and possibilities of their use. The last chapter consists of a practical part, which describes the development of a web application. There is a description of the parts of application, such as the database, the Web navigation, master pages and etc. in the following chapter. The cardinal part of the chapter consists of an analysis and implementation of forms authentication, the attribute authentication and authentication with QR code. Finally, there is a description of way how to secure the communication by using a certificate.

KEYWORDS

authentication, ASP.NET, cookies, Diffie-Hellman, cryptography, SHA, socket, HTTPS, HM12, QR code, webcam, ZXING

VYBÍRAL, Petr *Webové aplikace pro autentizaci uživatelů*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 61 s. Vedoucí práce byl Ing. Jan Hajný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Webové aplikace pro autentizaci uživatelů“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Janu Hajnému, Ph.D. za odborné vedení, ochotnou pomoc a rady při zpracování práce.

Brno ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Cílem diplomové práce je návrh webového portálu, který obsahuje uzavřené i volně dostupné sekce. Pro přístup k uzavřeným sekcím je implementováno formulářové ověřování s dokazovacím faktorem jméno a heslo. Další forma ověření je pomocí atributové autentizace, která využívá uložené tokeny na klientském počítači. Jako poslední je implementováno ověření s využitím QR kódu, který je snímán v klientské aplikaci pomocí webkamery.

Pro vývoj webové aplikace pro autentizaci uživatelů je zvolena technologie ASP.NET, která je velmi rozšířenou technologií. Kód je napsán v programovacím jazyce `C#`, což je objektově orientovaný programovací jazyk, vyvinutý zároveň s platformou .NET.

První kapitola prezentuje možnosti ověřování. Obsahuje jednotlivé autentizační metody a jejich následný rozbor.

Druhá kapitola představuje jednotlivé možnosti zabezpečení komunikace. Popisuje kryptografický systém a následně způsoby zabezpečeného přenosu informace. Více do hloubky je v kapitole rozebrána problematika asymetrického šifrování, a to sestrojení klíče pomocí protokolu Diffie-Hellman. Představen je hašovací algoritmus SHA-1. Dále je obsažen rozbor potřebných zabezpečovacích protokolů, autentizačního protokolu HM12 a čárového 2D QR kódu.

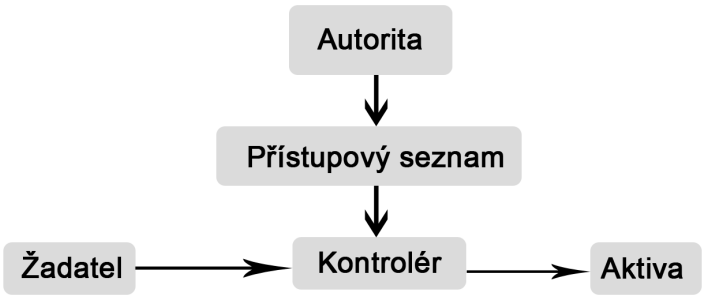
Třetí kapitola se zabývá technologií ASP.NET. Popisuje historický vývoj verzí ASP.NET a současně představuje její oblasti pro programování webových aplikací. Dále jsou v práci podrobně rozebrány webové formuláře a serverové ovládací prvky. Poté jsou analyzovány prvky cookies a jejich možnosti využití.

Poslední kapitola se věnuje tvorbě webové aplikace pro autentizaci uživatelů. Nejdříve je uveden rozbor struktury webu, popis potřebné databáze, způsob zpracování grafické stránky formulářů a navigace po webu. Následně je představena autentizace pomocí formulářů (jméno a heslo). Autentizace pomocí atributů a tokenů uložených na klienstkém počítači je rozebrána jak na straně serveru, tak i na straně klienta. Autentizace pomocí QR kódu představuje podstatnou část této kapitoly, popisuje sestrojení autentizačních dat, vytvoření komunikace a procesu ověření na straně serveru.

V příloze je vypsán obsah souborů z praktické části, jedná se o webovou aplikaci a dva klienty. Soubory jsou uloženy na přiloženém CD.

1 AUTENTIZACE

Autentizace je proces ověření identifikačních údajů žadatele, který vyžaduje přístup k aktivům (stránky, data, služby). Žadatel o přístup prokazuje kontroléru své ověřovací údaje, tzv. dokazovací faktor. Dokazovací faktor jsou buď soukromá data (heslo nebo klíč), unikátní charakteristické rysy autentizačního předmětu, nebo unikátní charakteristické rysy žadatele. Kontrolér má vždy určitý ověřovací faktor (například haš [1] hesla). Kontrolér vyhledá žadatele v přístupovém seznamu a zjistí jeho přístupová práva. Autorita na základě výsledku kontroléru rozhodujuje o přístupu uživatele – tento proces je nazýván autorizací, což je základní prvek ochrany sítí a systémů. Kromě povolení přístupu lze uživatele řadit do skupin a nastavovat jim různá oprávnění [2].



Obr. 1.1: Schéma řízení přístupu.

Přehled typů autentizací, kde je nosič autentizačního důkazu:

- Žadatel
  1. Autentizace **znalostí** – žadatel prokazuje svoji znalost (nejčastěji uživatelského jména a hesla).
  2. Autentizace **biometrikou** – unikátní charakteristické rysy žadatele (otisky prstů, sítnice, hlas).
- Předmět
  1. Autentizace **průkazem** – průkaz vydaný autoritou (například pas).
  2. Autentizace **úložištěm** – úložiště dat, která obsahují dokazovací faktor (například šifrovací klíč).

[1] haš – výstup hašovací funkce, více v části 2.4

1.1 Autentizace na základě znalosti

Žadatel je před povolením přístupu dotázán na znalost určité přístupové informace. Jedná se většinou o heslo (alfanumerický řetězec) nebo číselnou hodnotu (numerický řetězec). Heslo může být stálé, jednorázové nebo časově proměnné. Žadatel ukládá informace ve své paměti. Na straně kontroléru se hesla ukládají s využitím jednocestné hašovací funkce. Proces autentizace je poté závislý na porovnání příchozího a ověřovacího haše. Pro prolomení zabezpečení může útočník využít databázi hašů a slovníkových hesel. Obrana spočívá ve volbě těžko odhadnutelného hesla. Další ochranou je omezení počtu po sobě chybných pokusů zadání hesla kontroléru. Doplňková ochrana spočívá v přidání CAPTCHA [2], což je Turingův test pro rozlišení zadávání hesel mezi stroji a lidmi.

Mezi výhody autentizace znalostí patří, že jsou přístupová data uložena skrytě a dokazovací faktor má žadatel neustále u sebe. Nevýhodou metody je poměrně malý rozsah dokazovacích faktorů, protože si je žadatel potřebuje zapamatovat a následně si je vybavit při ověřování [2].

Při autentizaci znalostí typu *výzva-odpověď*, někdy také zvaná jako login-heslo, je vyžadována po žadateli znalost uživatelského jména a hesla. Žadatel vytvoří žádost u autority (zpravidla ověřovací server), autorita vytvoří odpověď obahující jednorázový náhodně vygenerovaný řetězec, tzv. nonce. Žadatel následně zasílá odpověď složenou z hesla a nonce za pomocí hašovací funkce. Kontrolér vyhledá v přístupovém seznamu (databázi) uživatelské jméno a jeho příslušné heslo. Vytvoří haš z hesla a odeslaného nonce a porovná ho s příchozí odpovědí žadatele. V případě shody je žadateli přístup povolen.

Dalším typem autentizace znalostí je *důkaz nulové znalosti* (Zero-Knowledge), kdy žadatel prokazuje autoritě znalost bez toho, aniž by prozradil svoji identitu. Využívá se zde asymetrické kryptografie pro sestrojení dokazovacího faktoru [4].

1.2 Autentizace předmětem

Žadatelé prokazují svoji identitu pomocí předmětu (tzv. tokenu) [16]. Přístupová informace je tedy uložena na určitém paměťovém zařízení (například čipová karta) nebo na povrchu předmětu (například identifikační karta). Z bezpečnostních důvodů se vyžaduje, aby útočník nemohl předmět padělat či odcizit.

Výhodou autentizace předmětem je, že si žadatel nemusí pamatovat přístupovou informaci, ale musí ji chránit proti odcizení, ztrátě a vhodnou technologií ji zajistit proti padělání.

[2] CAPTCHA je akronym pro „Completely Automated Public Turing test to tell Computers and Humans Apart“

Typy autentizačních předmětů lze rozdělit na:

1. **úložiště** (předmět, na který se ukládají autentizační informace):
   - se zabezpečenými daty,
   - s nezabezpečenými daty.
2. **autentizátory** (zařízení skládající se z dokazovacího procesoru a úložistě):
   - s přístupovou ochranou,
   - s kryptografickou ochranou.

1.3 Autentizace biometrikou

U autentizace biometrikou dochází k porovnávání biometrické charakteristiky žadatele (dokazovací faktor) s uloženými daty této charakteristiky (ověřovací faktor). Uložená charakteristická data se pořizují v rámci autorizace a jsou součástí přístupového seznamu nebo certifikátu podepsaného autoritou. Charakteristická data žadatele mají být jedinečná a neměnná. Ověřovaná data nebývají stejná, vždy nastane odchylka od vzoru, tudíž se hledá podobnost vzoru a předlohy [2].

Nejčastějším případem biometrické autentizace v informačních systémech je metoda založená na ověření otisku prstu. Využívají se i další fyziologické charakteristiky žadatele, jako jsou například oční sítnice, portrét tváře nebo geometrie ruky. Dalším dokazovacím faktorem mohou být charakteristické vlastnosti žadatele založené na rozdílnosti lidského chování, jako je například hlas. Zkoumá se rychlost řeči a frekvenční spektrum projevu žadatele.

1.4 Autentizace v ASP.NET

Autentizace v ASP.NET probíhá až po autentizaci v IIS (Internet Information Service). Technologie ASP.NET je popsaná v kapitole 3.

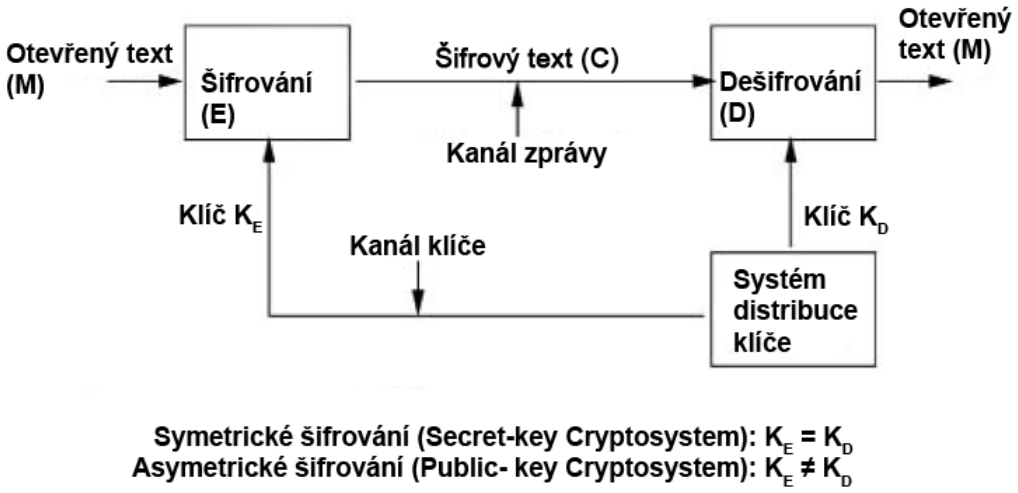
ASP.NET umožňuje tři módy autentizace:

- **Windows** – ASP.NET plně spoléhá na autentizaci v IIS. Po úspěšném ověření v IIS obdrží aplikace bezpečnostní token.
- **Formuláře** – proces autentizace probíhá plně v aplikaci. Uživatelé jsou ověřováni například pomocí autentizačních údajů v databázi. Postup používání formulářového ověřování je popsán v části 4.6.
- **Heslo** – .NET Password je centralizovaná autentizační služba poskytovaná společností Microsoft [11].

2 ZABEZPEČENÍ KOMUNIKACE

Pro bezpečný přenos informace mezi serverem a klientem se používají různé techniky zabezpečení přenášených dat. Šifrování je matematická transformace, která zabezpečuje původní zprávu (plaintext) na výstupní zpávu (ciphertext). Pro obnovení zaširovaných dat se zpětná transformace nazývá dešifrování. Šifrovací a dešifrovací algoritmy využívají kryptografické klíče. Popis šifrovacích a dešifrovacích algoritmů, definování formátu zprávy a klíče je nazýván jako kryptografický systém (kryptosystém) [10].

Průběh přenosu zprávy se zabezpečením zprávy je znázorněn na obr 2.1.



Obr. 2.1: Kryptografický systém.

Kryptografický systém se skládá z následujících částí [20]:

- **Otevřený text**(M – message, taktéž P – plaintext): původní zpráva.
- **Šifrový text** (C – ciphertext): šifrovaný text.
- **Klíč**(K):
  1. Pro případ:
  $$K_E = K_D, \tag{2.1}$$
  nástává symetrické šifrování (sekce 2.1), které pro šifrování i dešifrování využívá *tajný klíč* (secret key).
  2. Pro případ:
  $$K_E \neq K_D, \tag{2.2}$$

probíhá asymetrické šifrování (sekce 2.2) kde je:
K_E (public key): *veřejný klíč*, který slouží k šifrování.
K_D (private key): *soukromý klíč*, který slouží k dešifrování.

- **Šifrování** (E – encryption function): proces šifrování

$$C = E(K_E, M). \tag{2.3}$$

- **Dešifrování** (D – decryption function): proces dešifrování

$$M = D(K_D, C). \tag{2.4}$$

2.1 Symetrické šifrování

Symetrické kryptosystémy využívají pro šifrování i dešifrování stejný klíč. Zpráva se zašifruje pomocí tajného klíče a kdo ho zná, ten může zprávu dešifrovat. Klade se důraz na bezpečný přenos klíče pověřeným osobám. Symetrické šifry se dělí na *proudové* a *blokové* šifry.

Proudové šifry pracují na principu šifrování každého symbolu samostatně – případná chyba ovlivní pouze daný symbol. Výhodou je výpočetní rychlost a nenáročnost na výkon. Nevýhodou proudových šifer je náchylnost k úmyslným falzifikacím a modifikacím. Proudové šifry nezajišťují integritu a hrozí nebezpečí dvojího použití stejného hesla. Příklad proudové šifry je například RC4 [20], která se využívá u bezdrátové komunikace.

Blokové šifry šifrují skupinu symbolů otevřeného textu jako jeden blok. Velikost bloku zásadně ovlivňuje bezpečnost algoritmu. Blokové šifry poskytují vyšší bezpečnost než proudové šifry. V bloku šifry nemůže útočník změnit symbol, protože se projeví změna i při dešifrování. Nevýhodou blokových šifer je časová náročnost a šíření chyb. Při procesu šifrování je nutné přijmout celý blok a chyba v jedné části bloku ovlivní transformaci všech členů bloku. Příklad blokové šifry je například AES (Advanced Encryption Standard) [2].

2.2 Asymetrické šifrování

Asymetrické kryptosystémy využívají pro šifrování veřejný klíč a pro dešifrování soukromý klíč. Klíče nesmí být vzájemně odvoditelné ani se sobě nesmí rovnat. Asymetrické šifrování využívá výpočetně těžké problémy [10].

- Faktorizace velkých čísel.
- Diskrétní logaritmus.
- Eliptické křivky.

Bezpečnost algoritmů je zajištěna na základě obtížnosti matematické operace velkých čísel (například faktorizace).

Při komunikaci je nejprve zašifrována zpráva pomocí veřejného klíče. Následně se identifikuje protistrana. Protistrana vlastní soukromý klíč, kterým se následně dešifruje zpráva.

Obr. 2.2: Asymetrický kryptosystém.

Uvedené schéma na obrázku 2.2 zajišťuje autentičnost přenášených zpráv a neodmítnutelnost odpovědi podepisujícího (C může vytvořit jen majitel SK) [2].

2.2.1 Protokol Diffie-Hellman (DH)

Jedním z nejznámějších algoritmů asymetrického šifrování se stal protokol Diffie-Hellman, který jako první začal používat algoritmus, který poskytuje vytvoření a výměnu tajného klíče po nezabezpečeném přenosovém kanálu. Tento algoritmus vznikl v roce 1976 a autory jsou pánové Whitfield Diffie a Martin Hellman.

Protokol umožňuje bezpečné ustanovení klíče, který se využívá v symetrickém kryptosystému. Bezpečnost protokolu je založena na problému diskrétního logaritmu. Obě komunikující strany si vygenerují vlastní klíče, které si předají přes nezabezpečené prostředí a na základě příchozího klíče si odvodí tajný klíč pro vlastní přenos. Předností algoritmu je, že se nikdy nepřenáší tajný klíč v otevřené formě, takže ho útočník není schopen zachytit. Nevýhodou je útok tzv. *man in the middle*, kdy nedochází k ověřování identity komunikujících stran. Pro ověření identity se využívá zabezpečený přenos s použitím certifikátu.

Parametry kryptosystému [2]:

- Je dáno velké prvočíslo p a generátor g, jehož vlastností je, že pro: $x \in \{1, 2, 3, ..., p-1\}$ generuje funkce $y = (g^x)\ mod\ p$ všechna čísla z množiny $\{1, 2, 3, ..., p-1\}$.
- Odesílatel **A** zvolí náhodně velké číslo a a příjemce **B** zvolí náhodně velké číslo b. Tato čísla jsou tajná.

- Odesílatel **A** vypočítá číslo

$$X = (g^a)\ mod\ p, \tag{2.5}$$

příjemce **B** vypočítá číslo

$$Y = (g^b)\ mod\ p. \tag{2.6}$$

- Odesílatel **A** vypočítá tajný klíč

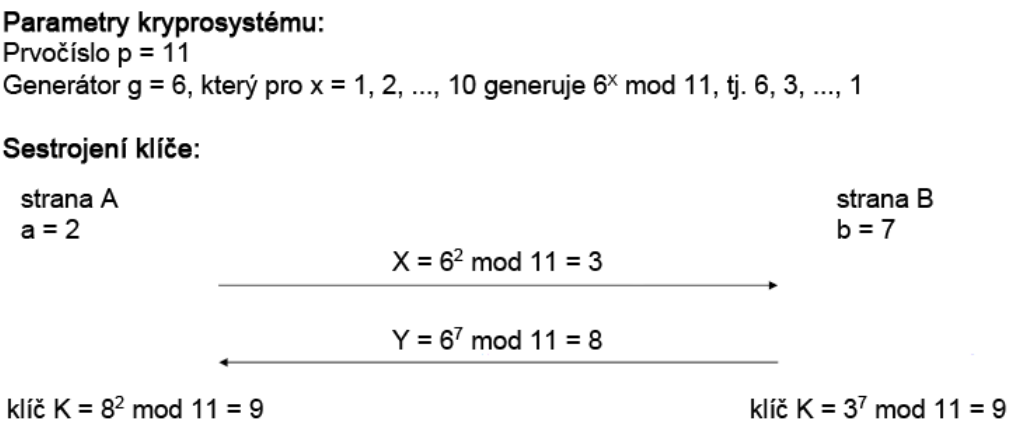
$$K = (Y^a)\ mod\ p, \tag{2.7}$$

příjemce **B** vypočítá tajný klíč

$$K = (X^b)\ mod\ p. \tag{2.8}$$

- Využívá se skutečnost, že:

$$K = (Y^a)\ mod\ p = (g^b)^a\ mod\ p = (g^a)^b\ mod\ p = (X^b)\ mod\ p. \tag{2.9}$$



Obr. 2.3: Příklad sjednání klíče protokolu Diffie-Hellman [3].

Útočník není schopen z odposlechnutého X nebo Y určit a nebo b. Zde nastává problém výpočtu diskrétního logaritmu (DL problém), tzn. že útočník není schopen vypočítat klíč K.

[3]Zdroj: Návrh, správa a bezpečnost počítačových sítí. Doc. Ing. Karel Burda, CSc. FEKT VUT v Brně (přednáška 1).

2.3 Kontrola integrity zpráv

Z důvodu zajištění autentičnosti (původnosti) přijatých dat se využívá asymetrických kryptosystémů. Ty však mají dva velké nedostatky – pomalé šifrování (dešifrování) a nutnost ověření veřejného klíče. První nedostatek se řeší pomocí tzv. hašů (hash) a druhý pomocí PKI (infrastruktura veřejných klíčů – Public Key Infrastucture). Místo šifrování celé zprávy se poté šifruje pouze její haš. Odesílatel ke každé zprávě připojuje autentizační kód zprávy, který slouží ke kontrole. Autentizační kódy se tvoří autentizační nebo hašovací funkcí. Hašovací funkce:

$$h = H(Z), \tag{2.10}$$

kde Z je původní zpráva a h je výsledný haš.

Kontrola integrity zprávy spočívá v tom, že autentizační kód h originálu zprávy Z je bezpečně přenesen v kryptogramu. Jeho hodnota se poté porovná s hodnotou $h´$ ověřované kopie zprávy Z. Vstupní řetězec libovolné délky je hašovací funkcí komprimován na konstantní délku (128, 160, 192, ...). Po hašovací funkci se vyžaduje, aby byla jednosměrná, odolná vůči modifikaci vzoru a odolná vůči kolizi [2].

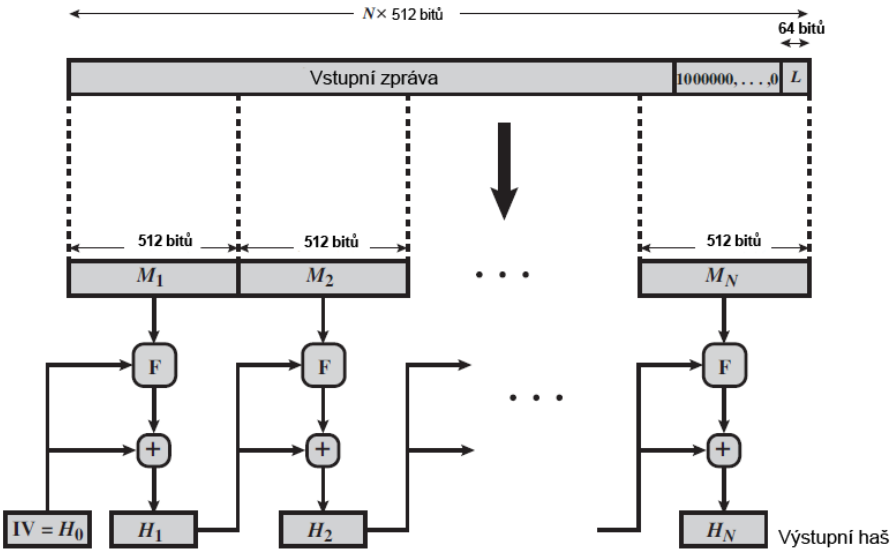
2.4 Hašovací algoritmus SHA-1

SHA (Secure Hash Algorithm) je kryptografický hašovací algoritmus vytvořený národním institutem standardů a technologie NIST (National Institute of Standards and Technology). SHA je založený na hašovací funkci MD4 [17]. V tabulce 2.1 je porovnání algoritmů SHA-1 a SHA-512.

Tab. 2.1: Srovnání parametrů hašovacích algoritmů SHA-1 a SHA-512

Algoritmus/Parametr	**SHA-1**	**SHA-512**
Výstupní zpráva [bit]	160	512
Vstupní zpráva [bit]	2^{64}	2^{128}
Délka bloku [bit]	512	1024
Délka slova [bit]	32	64
Počet kroků [-]	80	80

SHA-1 je specifikovaný v RFC 3174 [3]. Algoritmus přijímá vstupní zprávu o velikosti až 2^{64} a produkuje výstup o 160 bitech. Obrázek 2.4 znázorňuje celkové zpracování zprávy.



Obr. 2.4: Schéma zpracování zprávy při použití haše.

2.4.1 Předzpracování

- Přidání výplně (padding) do zprávy. První bit výplně je vždy „1“. Následně se doplní „0“ tak, aby její délka byla shodná s 448 (mod 512).
- Blok L je připojen na konec vyplněné zprávy. Tento blok je považován za nepodepsané 64bitové celé číslo (most significant byte). Délka zprávy je nyní násobkem 512 bitů.

2.4.2 Zpracování

- Rozdělení zprávy na bloky velké N x 512 bitů.
- Dělení bloků do M x 32bitových slov.
- Provedení požadovaných operací (F-nelineární funkce, rotace, sčítání mod 2^{32}) [19].
- Výstup tvoří 160 bitů dlouhý kontrolní haš.

2.5 Socketová komunikace

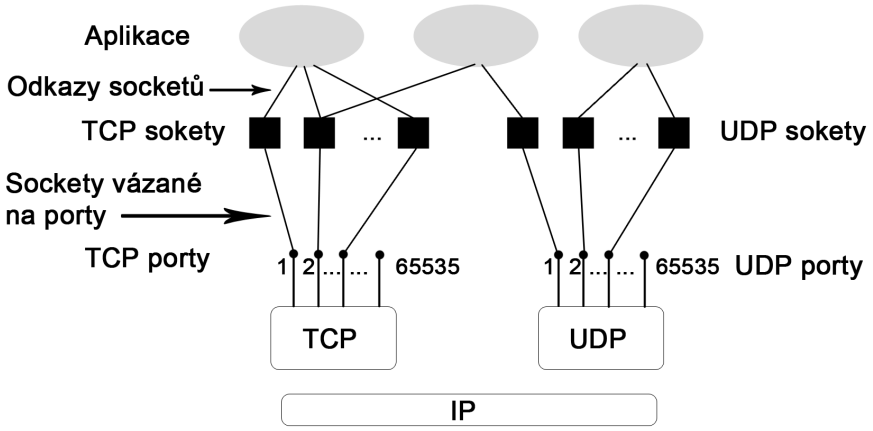
Socket představuje typ síťové komunikace mezi dvěma procesy. Tyto procesy reprezentují dva uživatelé (server, klient), kteří jsou připojeni k síti pomocí protokolu IP (Internet Protocol). Může také nastat situace, kdy server a klient komunikují ze stejného počítače pomocí speciální IP adresy „localhost“.

Socket je tedy datový objekt obsahující informace o stavu komunikace s druhou stranou. Pro vlastní komunikaci se využívají systémová volání, která na daný objekt

odkazují deskriptorem. Výhoda spočívá hlavně v transparentnosti vůči transportní vrstvě. Sockety mají využití tam, kde se počítá s propustností sítě.

Existují dva druhy přenosů socketové komunikace:

- **Spojový** (stream sockets) – nejdříve je sestaven komunikační kanál, kterým se následně zasílají data v podobě proudu bytů. Pro přenos se využívá protokol TCP (Transmission Control Protocol), který bude používán nadále v práci
- **Datagramový** (datagram sockets) – je sestaven balík dat, který je odeslán jako celek. Pracuje se protokolem UDP (User Datagram Protocol) [6].



Obr. 2.5: Socketová komunikace mezi aplikacemi.

Komunikace na straně serveru probíhá následujícím způsobem. Nejdříve se vytvoří obálka TCP socketu, poté se přiřadí její název, tedy server zaregistruje jeden port, na kterém bude pracovat. Nastaví se „naslouchací“ režim. Zde se čeká na připojení klienta. Když se klient připojí, server zahajuje komunikaci. Po dokončení relace se ukončí spojení a uzavře socket. Komunikace klienta je velmi podobná jako u serveru. Vytvoří se TCP socket, kterému klient přiřadí IP adresu a číslo portu, na kterém pracuje server. Poté probíhá vlastní komunikace. Po ukončení komunikace se klient odpojí od serveru a následně zruší socket. Prostá ukázka funkce je na obrázku 2.5.

2.6 Komunikační protokol HTTPS

Protokol HTTPS (HyperText Transfer Protocol Secure) je nadstavba síťového protokolu HTTP (HyperText Transfer Protocol), která slouží pro zabezpečení komunikace mezi serverem a klientem tím, že ověřuje identitu protistrany a šifruje přenášená data mechanismy SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security) [15].

SSL je zabezpečovací mechanismus, který se využívá na aplikační vrstvě. Zajišťuje autenticitu odesílatele, integritu a šifrování aplikačních dat při jejich přenosu přes veřejnou IP síť. SSL požaduje spolehlivý transportní protokol – TCP. SSL používá kombinaci šifrování veřejným a tajným klíčem. Asymetrické šifrování slouží v průběhu počáteční fáze k autentizaci a generování klíčů, tajný klíč se poté využívá pro šifrování zpráv.

SSL podporuje obousměrnou autentizaci, ale většinou se používá pouze jednosměrně. Autentizaci lze provést pomocí:

- jména a hesla (RADIUS),
- jména a tokenu (RSA SecureID),
- digitálního certifikátu.

Klient spoléhá na digitální certifikát serveru, k němuž se chce připojit. Útočník může ale certifikát zfalšovat a poté provést útok typu *man in the middle* [2]. Tomu lze zabránit dvoustupňovou autentizací pomocí hesla a tokenu [15].

SSL podporuje bezpečnostní algoritmus pro zajištění integrity zpráv SHA-1 a komunikuje většinou na portu 443.

2.7 Autentizační protokol HM12

Pro atributové ověřování pomocí tokenů uložených na klientských počítačích byl zvolen autentizační protokol nulové znalosti HM12. Jedná se o atributové ověřování za účelem zvýšení soukromí uživatelů. Jednou z výhod tohoto protokolu je, že přihlašovací data neobsahují důvěrné informace o uživateli. Vlastnictví těchto atributů lze anonymně prokázat ověřovateli. Bezpečnost protokolu je založená na problému nalezení diskrétního logaritmu.

Autentizační protokol HM12 byl vytvořen na VUT v Brně. Podporuje odvolání vyloučených uživatelů a taktéž odhalení identity. Žadatel prokazuje vlastnictví přístupového atributu díky vytvořeným autentizačním datům. Protokol definuje vytváření parametrů (registrační fáze), autentizaci (důkazová fáze) a revokaci. Funguje na principu rozlišování reprezentací diskrétních logaritmů (DL_{REPS}) [4].

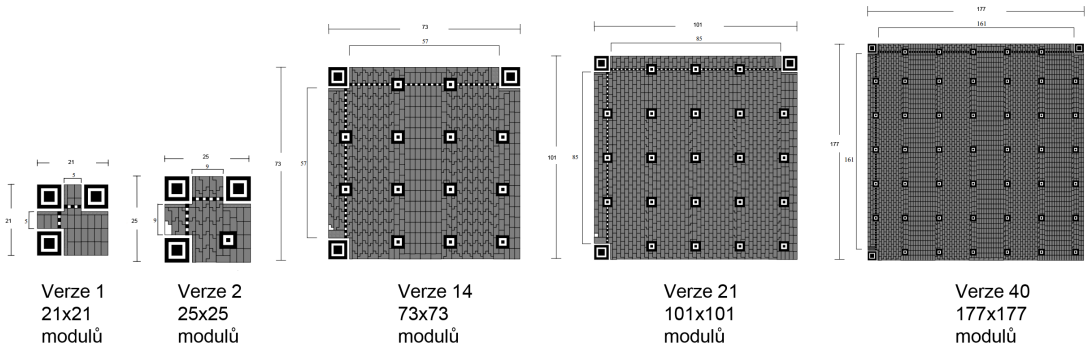
Žadatel využívá veřejné parametry, přidělený klíč a vlastní náhodně vygenerovaná čísla pro výpočet autentizačního tokenu. Žadatel vypočítá haš s využitím jednosměrné hašovací funkce a následně dopočítá odpověď. Odpověď zasílá serveru k ověření.

Ověřovatel z přijaté odpovědi vypočítá své parametry a s využitím jednosměrné hašovací funkce získá haš. Na základě porovnání přijatého a vypočítaného haše rozhoduje autentizační server o povolení přístupu. V případě shody je proces autentizace úspěšně dokončen a uživatel ověřen.

2.8 QR kód

QR kód (Quick Response – rychlá odezva) je maticový (2-D) čárový kód. Tento kód vychází z normy ISO/IEC 18004, ze které je čerpáno v této sekci [7]. QR kód využívá čtyři standardizované způsoby kódování, a to numerické, alfanumerické, binární/bytové a kanji. Data jsou přesně definována dvourozměrnými souřadnicemi černých modulů (čtvercové tečky), které jsou uspořádané v matici QR kódu. Výsledný kód má tedy podobu čtvercové sítě na bílém podkladu.

QR kód rozlišuje 40 verzí, každá verze má své specifické vlastnosti, jako jsou například velikost, kapacita, počet symbolů, atd. Matice kódu verze 1 má rozměr 21x21 bodů, pro verzi 40 platí rozměr 177x177 bodů. Porovnání verzí QR kódu je vidět na obrázku 2.6.



Obr. 2.6: Verze QR kódu.

Do QR kódu lze zakódovat poměrně velké množství informací, a to konkrétně pro verzi 40 až 7 089 numerických znaků, 4 296 alfanumerických znaků, 2 953 bytů a 1 817 japonských znaků Kanji. Přehled možné kapacity QR kódu je vidět v tabulce 2.2.

QR kód nabízí čtyři úrovně korekce chyb a to:

- Úroveň L (Low) – 7 % z celkové kapacity QR kódu. Správně se udává codewords[1]. Čím vyšší úroveň korekce chyb, tím vyšší počet codewords korekce chyb a nižší počet codewords použitelných pro vlastní data.
- Úroveň M (Small) – 15 %z codewords může být obnoveno.
- Úroveň Q (Quality) – 25 % z codewords může být obnoveno.
- Úroveň H (Hight) – 30 % z codewords může být obnoveno.

Pro korekci chyb se využívá ne-binární cyklický samoopravný algoritmus Reed-Solomonon. Tyto algoritmy dokáží detekovat více náhodných chyb při vytváření kódů a v případě chybějících dat je také doplnit.

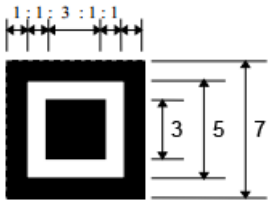
[1]codewords je myšleno zakódované 8bitové binární číslo, jehož velikost a tvar je dán maticí QR kódu.

Tab. 2.2: Vstupní kapacita QR kódu podle verze.

Verze	Korekční úroveň	Počet codewords	Počet bitů	Počet numerických znaků	Počet alfanumerických znaků
1	L	19	152	41	25
	M	16	128	34	20
	Q	13	104	27	16
	H	9	72	17	10
14	L	461	3 424	1 101	667
	M	365	2 920	871	528
	Q	261	2 088	621	376
	H	197	1 576	468	283
21	L	932	7 456	2 232	1 352
	M	714	5 712	1 708	1 035
	Q	512	4 096	1 224	742
	H	406	3 248	969	587
40	L	2 956	23 648	7 089	4 296
	M	2 334	18 672	5 596	3 391
	Q	1 666	13 328	3 993	2 420
	H	1 276	10 208	3 057	1 852

2.8.1 Symboly QR kódu

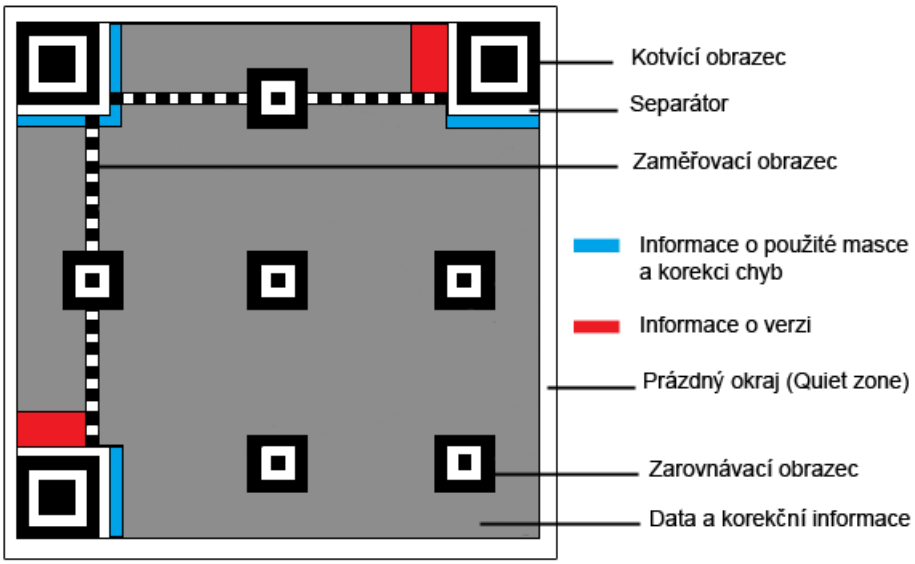
- *Finder pattern* – tzv. kotvící obrazec se skládá ze tří Position Detection Patterns, což jsou tři soustředné čtverce, které slouží pro rychlé lokalizování levého horního, pravého horního a spodního levého rohu.



Obr. 2.7: Findern pattern, tzv. kotvící obrazec QR kódu.

- *Alignment Patterns* – zarovnávací obrazce, snadno rozlišitelné referenční body, které slouží dekódovacím zařízením pro synchronizaci souřadnicového mapování modulů i v případě mírného zkreslení/natoční obrazu. S rostoucí verzí kódu roste počet Alignment sektorů, přičemž začínají od QR kódu verze 2.

- *Timing pattern* – zaměřovací obrazec, všem verzím QR kódu je vyhrazen 6. řádek a sloupec střídajících se modulů černé a bílé barvy, kdy první a poslední značka je černá (logická nula).
- *Separators* – separátor, slouží k oddělení refenčních bodů a zbytku symbolů.
- *Quiet zone* – prázdný okraj kódu, oblast po obvodu QR kódu o šířce 4 modulů, který má za účel rozlišit kód a okolí obrazu.
- Ostatní části – nachází se zde vlastní data, data o použité masce a data o typu korekce chyb.



Obr. 2.8: Popis částí QR kódu.

2.8.2 Postup kódování QR kódu

Kódování QR kódu závisí na formátu vstupních dat. V této práci jsou vstupní data alfanumerického typu, proto bude popsán postup pouze pro tento formát.

Vstupní data jsou převedena do bitového proudu, skládající se z hlavičky EIC (Extended Channel Interpretation). Hlavička EIC obsahuje:

- EIC mód indikátor (4 bity, alfanumerická data = 0010).
- EIC označení (8, 16 nebo 24 bitů).

Zbytek bitového proudu se skládá z částí, z nichž každá obsahuje mód indikátor, indikátor počtu znaků a vlastní data.

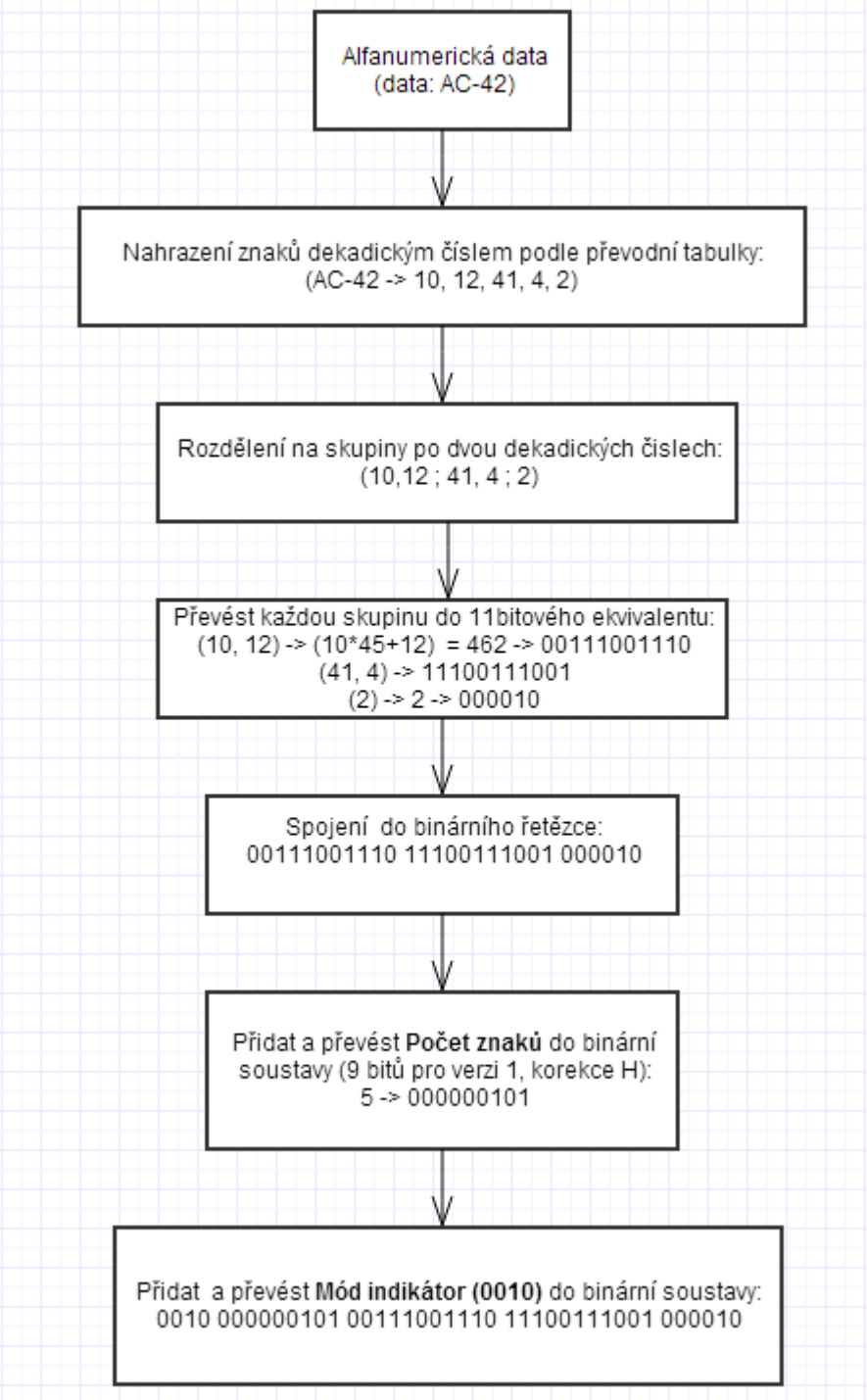
Přehled procesu kódování je znázorněn na obrázku 2.9.

Po procesu kódování následuje aplikace logické funkce XOR mezi vytvořenou binární maticí a maskou.

Tab. 2.3: Převodní tabulka pro alfanumerické kódování QR kódu.

Znak	**Hodnota**	**Znak**	**Hodnota**	**Znak**	**Hodnota**
0	0	F	15	U	30
1	1	G	16	V	31
2	2	H	17	W	32
3	3	I	18	X	33
4	4	J	19	Y	34
5	5	K	20	Z	35
6	6	L	21	SP	36
7	7	M	22	$	37
8	8	N	23	%	38
9	9	O	24	*	39
A	10	P	25	+	40
B	11	Q	26	-	41
C	12	R	27	.	42
D	13	S	28	/	43
E	14	T	29	:	44

Masky jsou vzory o velikosti QR kódu, které slouží pro zamezení vzniku souhlasných částí kódu (černé/bílé shluky). Existuje 8 masek, ze kterých se vybrere právě ta, která vytvoří výsledný obraz bez těchto nežádoucích jevů.



Obr. 2.9: Postup kódování alfanumerických dat QR kódu.

3 ASP.NET

3.1 Vývoj

Technologie ASP (aktivní serverové stránky – Active Server Pages) byla vyvinuta firmou Microsoft za účelem rychlého a snadného vytváření webových stránek. Web tvořila zpravidla jedna stránka, která obsahovala kombinaci různých značek a jazyků. Síla spočívala v tom, že bylo možné do stránky zahrnout instrukce kódu VBScriptu nebo JScriptu. Kód se následně vykonal na straně serveru ještě před odesláním stránky webovému prohlížeči klienta. Jednalo se o jednoduchou možnost vytváření dynamických webových stránek [1].

Když byla v roce 2000 představena beta verze .NET Frameworku, byl to začátek vytváření aplikací objektově orientovaným způsobem, namísto procedurálního postupu využívaného v ASP 3.0. S finální verzí rámce .NET Framework 1.0 došlo k přejmenování na ASP.NET.

Klíčovou ideou ASP.NET verze 1.0 byl model navrhování webové stránky zvaný webové formuláře (web forms), více v části 3.2. Verze 2.0 si zachovala stejné jádro abstrakce jako předchozí verze, ale přinesla nové víceúrovňové funkce, ke kterým patří například vzory stránek (master pages), motivy (themes), navigace (trees), ovládací prvky pro zdroje dat a další.

Ve verzi 3.5 přibyla sada rozšíření LINQ (Language Integrated Query) a AJAX (Asynchronous JavaScript and XML). LINQ umožňuje psát kód, který manipuluje s daty podobně jako dotazování do databáze. AJAX je skriptovací technika na straně klienta využívaná v prvcích jako je například menu. Princip AJAX spočívá v tom, že ve stránce nastane událost, prohlížeč spustí asynchronní volání ASP.NET a server vrátí pouze data, která se použila k aktualizaci stránky.

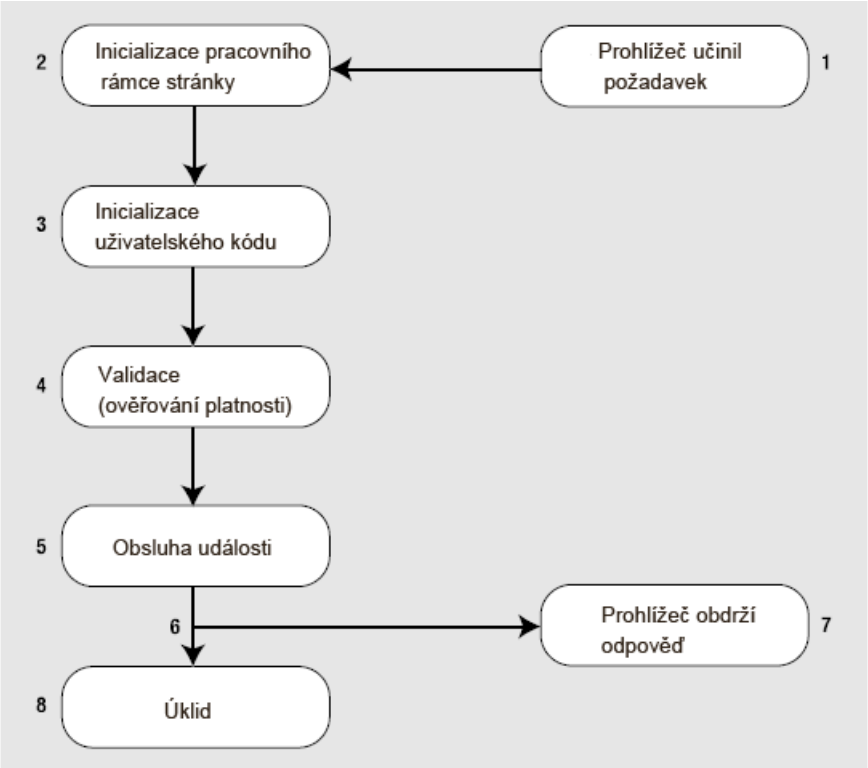
ASP.NET 4.0 dále vylepšuje již stávající systém o konzistentní realizace XHTML (Extensible HyperText Markup Language), aktualizovanou detekci prohlížeče, komprimace stavu relace, rozšiřitelné cachování a jiné. Mezi nejvýznamnější samostatné doplňky patří MVC (Model View Controller), ASP.NET Dynamic Data a Silverlight [8].

V dnešní době je technologie ASP.NET jednotný model vývoje webu, zahrnující služby nezbytné k sestavení rozsáhlých webových aplikací s minimálním množstvím kódu. Technologie ASP.NET je součástí platformy .NET Frameworku, která poskytuje své třídy. Aplikace je možné psát v libovolném programovacím jazyce, který je kompatibilní s modulem CLR (Common Language Runtime). Mezi nejpopulárnější jazyky patří například Microsoft Visual Basic a C# [12].

3.2 Stránky – webové formuláře

Stránky ASP.NET, zvané také jako webové formuláře (web forms), tvoří podstatnou část webové aplikace. Klienti ve svých prohlížečích přistupují na stránky serveru. Webové formuláře obsahují serverové ovládací prvky (více v části 3.3), pomocí kterých může uživatel provádět požadované operace. Když prohlížeč odešle požadavek na konkrétní stránku, ASP.NET vytvoří instanci objektu stránky a poté vytvoří objekty pro všechny ovládací prvky ASP.NET, které jsou uvnitř této stránky. Stránka a její řízení projde posloupností událostí jejího životního cyklu, realizuje finální HTML (hypertextový značkovací jazyk – HyperText Markup Language) a uvolní se z paměti [13], jak je vidět na obrázku 3.1.

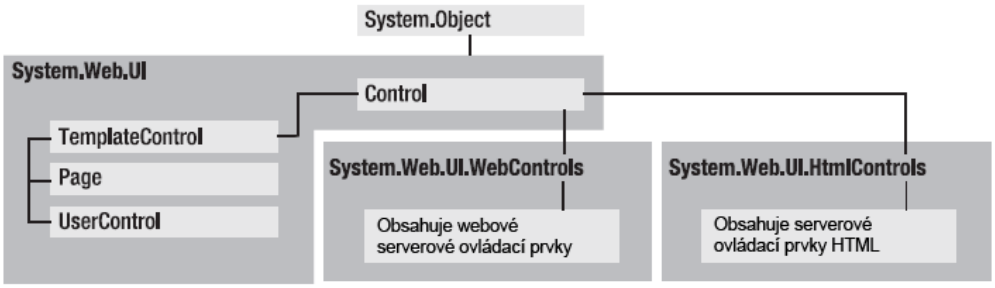
Stránky webových formulářů se dají rozdělit na dvě od sebe odlišitelné části – vizuální elementy formuláře a doprovodní logiku uživatelského rozhraní. Obě části se mohou ukládat buď do jednoho souboru `*.aspx` nebo separovat logiku do třídy `*.cs`. Všechny webové formuláře se odvozují ze třídy `Page` ze jmenného prostoru `System.Web.UI`.



Obr. 3.1: Životní cyklus stránky ASP.NET.

3.3 Serverové ovládací prvky

ASP.NET rozlišuje různé druhy serverových ovládacích prvků (server controls). Reprezentují vizuální komponenty ve webovém formuláři. Pro každý serverový ovládací prvek platí, že je v něm obsažen kód zobrazení v HTML a kód dodávající mu jeho vlastnosti, metody a události. Všechny serverové ovládací prvky jsou odvozeny ze základní třídy `Control` ze jmenného prostoru `System.Web.UI` [8], jak je vidět na obrázku 3.2.



Obr. 3.2: Hlavní větve dědičnosti serverového ovládacího prvku.

- **Datové ovládací prvky**. Zde se nachází všechny tabulky, mřížky a seznamy za účelem zobrazování většího množství dat. Na tyto prvky lze vázat různé datové zdroje. Patří k nim spousta funkcí, jako je například vložení, úprava, mazání a šablony dat. Jako příklad lze uvést *GridView* a *FormView*.
- **Navigační ovládací prvky**. Prvky *SiteMap* a *TreeView* slouží pro procházení stránek a navigaci po webu.
- **Přihlašovací ovládací prvky**. Tyto prvky podporují ověřování založené na formulářích. Pomocí providerů se sleduje stav uživatelů. Slouží k tomu komponenty jako jsou *Login*, *LoginName* a další.
- **Serverové ovládací prvky HTML**. Třídy, které obsahují standardní značky HTML a deklarují se atributem `runat="server"`. Příkladem může být *HtmlAnchor* pro značku `<a>`.
- **Validační ovládací prvky**. Sada prvků, pomocí kterých si vývojář může ověřovat definovaná pravidla a standardy. Pokud nejsou pravidla splněna, vývojář si může zvolit z řady možností (neodesílání stránky, vypsání chyby, vyskočení chybové hlášky, atd.). Příkladem je prvek *CompareValidator*.
- **Webové ovládací prvky**. Třídy, které také duplikují funkcionalitu základních značek HTML, mají však navíc bohatou sadu metod a vlastností.

3.4 Prvky cookies

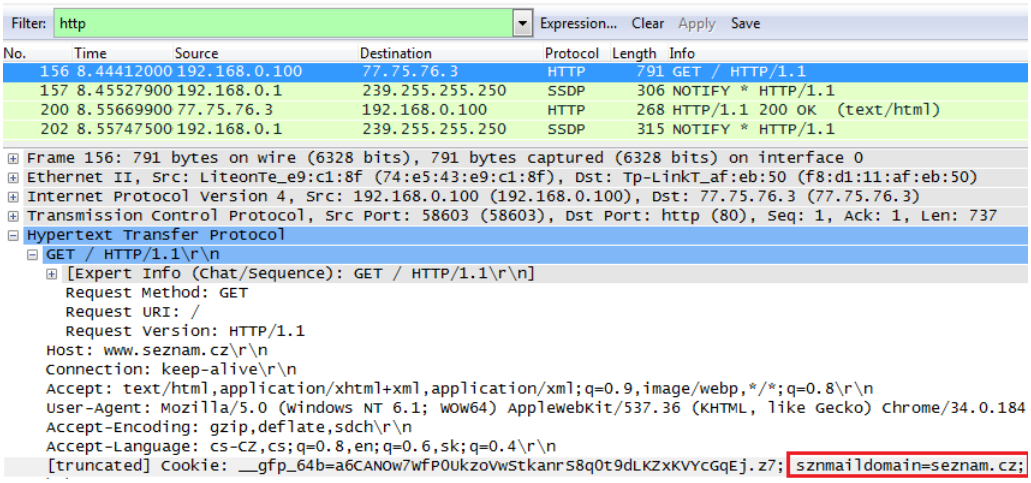
Webové stránky využívají ke komunikaci s uživatelem nestavový protokol HTTP. Na server přichází požadavky z různých IP adres, které jsou seřazené podle času vytvoření události uživateli. Pro využití určitého nastavení uživatele je nutné zakódovat data do jednotlivých požadavků klienta, k čemuž slouží právě cookies.

Cookie je určitý objem dat, která se přenáší z prohlížeče na webový server úplně s každým požadavkem. Prvky cookie se přenášejí v hlavičkách HTTP – konkrétně v hlavičkách *Cookie* [14].

Prvky cookie se oddělují středníkem a obsahují názvy a hodnoty, jak je vidět níže.

```
Cookie: BackgroundColor=Blue; Expires=Thu, 31-Dec-2014 12:00:00 GMT; path=/
```

Z výpisu HTTP protokolu na obrázku 3.3 je vidět, že v cookie je uložena vlastnost, která obsahuje název (*sznmaildomain*) a hodnotu (*seznam.cz*).



Obr. 3.3: Výpis z komunikace mezi klientem a serverem.

Mnoho serverů používá prvky cookie k identifikování vracejících se uživatelů, čímž předcházejí jejich opakovanému přihlášení na webové servery. ASP.NET používá prvky cookie ke spojení vracejících se návštěvníků s relacemi uloženými na webovém serveru. Cookie tak přenáší přihlašovací informace, uživatelské preference nebo specifická data serveru.

Webový server vytváří prvek cookie tím, že vrátí v hlavičce *Set-Cookie* reakce HTTP. Součást *Path* hlavičky `Set-Cookie` říká klientovi pro jakou skupinu adres v této doméně prvek cookie platí. Pokud je atribut `path = /`, znamená to, že se prvek cookie přenese v každém požadavku na tuto doménu. Způsob zacházení prohlížeče s cookie záleží na jeho typu [14].

- **Session cookie** – cookie relace, jsou platné pouze po dobu spuštění prohlížeče a obvykle se ukládají jen do paměti.
- **Persistent cookie** – trvalé prvky cookie mají přesně definovanou životnost (*Expires*), která nezáleží na prohlížeči. Jsou uloženy na pevném disku uživatele.

Prvek cookie, kterému chybí atribut `Expires`, je cookie relace. Prvek, který má tento atribut je trvalý cookie. Prvky cookie mohou rovněž zahrnovat atributy `Secure`, které zabraňují prohlížečům v jejich vysílání nešifrovanými kanály (jinými než HTTPS). Atribut se tedy využívá k přenášení citlivých dat. Vystupuje zde funkce `IPEndPoint`, která má dva konstruktory. Prvním je IP adresa (*Any* – znamená, že naslouchá všem adresám) a druhým je číslo portu.

Knihovna tříd .NET Framework zjednodušuje využití prvků cookie zapouzdřením do třídy `HttpCookie` jmenného prostoru `System.Web`.

```
HttpCookie cookie = new HttpCookie("BackgroundColor=Blue", "Blue");
```

Veřejné vlastnosti HttpCoockie nazvané `Domain, Path, Expires, Secure` obsahují atributy cookie stejných názvů. Následující příkaz nastavuje životnost cookie tak, že vyprší od doby vytvoření za týden.

```
cookie.Expires = DateTime.Now + new TimeSpan(7,0,0,0);
```

Pro odpověď stačí prohlížeči přidat prvek cookie do kolekce *Cookies* objektu `Response`. Ten zajistí, že se daný prvek vrátí v reakci HTTP.

```
HttpCookie cookie = new HttpCookie("BackgroundColor=Blue", "Blue");
Response.Cookies.Add(cookie);
```

Pro čtení příchozích požadavků cookie slouží objekt `Request`. Následující příklad znázorňuje čtení prvku cookie z požadavku. Ověřuje, zdali je prvek obsažen a pokud ano, extrahuje jeho hodnotu.

```
HttpCookie cookie = Request.Cookies["BackgroundColor"];
if(cookie != null){
    string BackgroundColor = cookie.Value;
```

Prvky cookie mohou být mazány uživateli nebo webovým serverem. Uživatel zničí cookie prostým zavřením prohlížeče. Trvalé cookie se ničí odstraněním souborů, v nichž jsou uloženy.

Server maže prvky cookie:

- vrácením prázdných hlaviček `Set-Cookie`. Tyto hlavičky obsahují názvy prvků cookie, které se mají vymazat.
- zahrnutím do těchto hlaviček `Set-Cookie`. Tyto hlavičky obsahují data vypršení, která specifikují čas v minulosti a jsou brány jako expirované.

4 NÁVRH WEBOVÉ APLIKACE

4.1 Analýza zadání a návrh řešení

Podle zadání diplomové práce je vytvořena webová aplikace v prostředí Microsoft Visual Studio 2010 v programovacím jazyce `C#`. Webová aplikace obsahuje volně dostupné stránky i zabezpečenou sekci určenou pro registrované uživatele. Pro přístup do zabezpečené sekce se mohou uživatelé přihlásit pomocí svého uživatelského jména a hesla. K tomu využívá webová aplikace Microsoft SQL (Structured Query Language) databázi. Ověření pomocí atributové autentizace a ověření pomocí QR kódu je realizováno pomocí autentizačního HM12 protokolu, který využívá hašovací funkci SHA-1. Atributové ověření využívá klientskou aplikaci s přidělenými tokeny pro vygenerování autentizačních dat do textového souboru. Ověření pomocí QR kódu probíhá za pomocí klientské aplikace, která pomocí webkamery zachytí QR kód. Pro zachycení QR kódu využívá klientská aplikace externí open-source knihovnu ZXING [21]. Následně klient zasílá ověřovací data na server. Spojení mezi serverem a klientem je šifrováno 128bitovým šifrováním. Připojení využívá protokol TLS 1.0.

4.2 Struktura webu

- *App_Code/***Code.cs** – třída poskytující metody pro generování náhodného alfanumerického řetězce.
- *App_Code/***Connections.cs** – třída určená pro TCP komunikaci mezi serverem a klientem při výměně dat QR kódu. Dále jsou v souboru metody pro ověření klienta za pomoci autentizačního protokolu HM12.
- *App_Code/***Protokol.cs** – třída obsahuje metody pro ověření klienta atributovou autentizací s využitím tokenu na klientském počítači.
- *App_Code/***SimpleAES.cs** – třída poskytující metody pro kódování a dekódování dat pomocí symetrické šifry AES.
- *App_Data/***ASPNETDTB.MDF** – vytvořená databáze obsahující veškeré informace o uživatelích a funkcích webu, více v části 4.3.
- *App_Data/***System.Numerics.dll** – knihovna pro zavedení datového typu `BigInteger`.
- *App_Themes/Default/***Skinfile.skin** – umožňuje ASP.NET komponentám nastavovat dynamicky HTML a CSS.
- *App_Themes/Default/***StyleSheet.css** – téma pro nastavení vzhledu definovaných částí.

- *Txt* – složka, do které se uploaduje textový soubor pro ověření tokenu a dočasné úložiště dat QR kódu klienta.
- *Users/***Data.aspx** – stránka s omezeným přístupem. Na tuto stránku mají přístup pouze registrovaní uživatelé. Zobrazuje důvěrné informace o projektech.
- **AboutUs.aspx** – stránka, která poskytuje základní informace o sídle a kontakty na pověřené osoby.
- **Default.aspx** – stránka obsahuje informace o webu a jeho obsahu. Jedná se také o výchozí stránku serveru.
- **Error.aspx** – stránka na kterou je uživatel přesměrován při zachycení neočekávatelné výjimky serveru.
- **Login.aspx** – stránka slouží pro přihlášení uživatelů, buď pomocí uživatelského jména a hesla, nebo pomocí vlastního tokenu. Dále stránka obsahuje funkce pro přijetí dat QR kódu od klientské aplikace.
- **Login.aspx.cs** – třída stránky Login.aspx, která obsahuje její funkční kód.
- **Redirect.aspx** – na tuto stránku je uživatel přesměrován pro přihlášení pomocí QR kódu. Stránka spolupracuje s klientskou aplikací *Protokol_QR_Web cam_Cient*, která je popsaná v části 4.8.
- **Redirect.aspx.cs** – třída stránky Redirect.aspx, která obsahuje její funkční kód.
- **Register.aspx** – stránka nabízí možnost registrování uživatelů.
- **MasterPage.master** – stránka slouží jako vzor pro všechny přidružené stránky. Vzor využívá obsah složky */Images*, vytvořený layout pro vzhled webu (header, content, footer), více v části 4.4.
- **web.config** – konfigurační soubor založený na XML (Extensible Markup Language). Obsahuje všechna nastavení (zabezpečení, providery, linkování stylů, dotazy do databáze, ...).
- **Web.sitemap** – mapa webu, slouží pro rychlý přechod mezi stránkami, více v části 4.5.

Celkový přehled struktury webové aplikace je vidět v příloze na obrázku 6.1.

4.3 Databáze

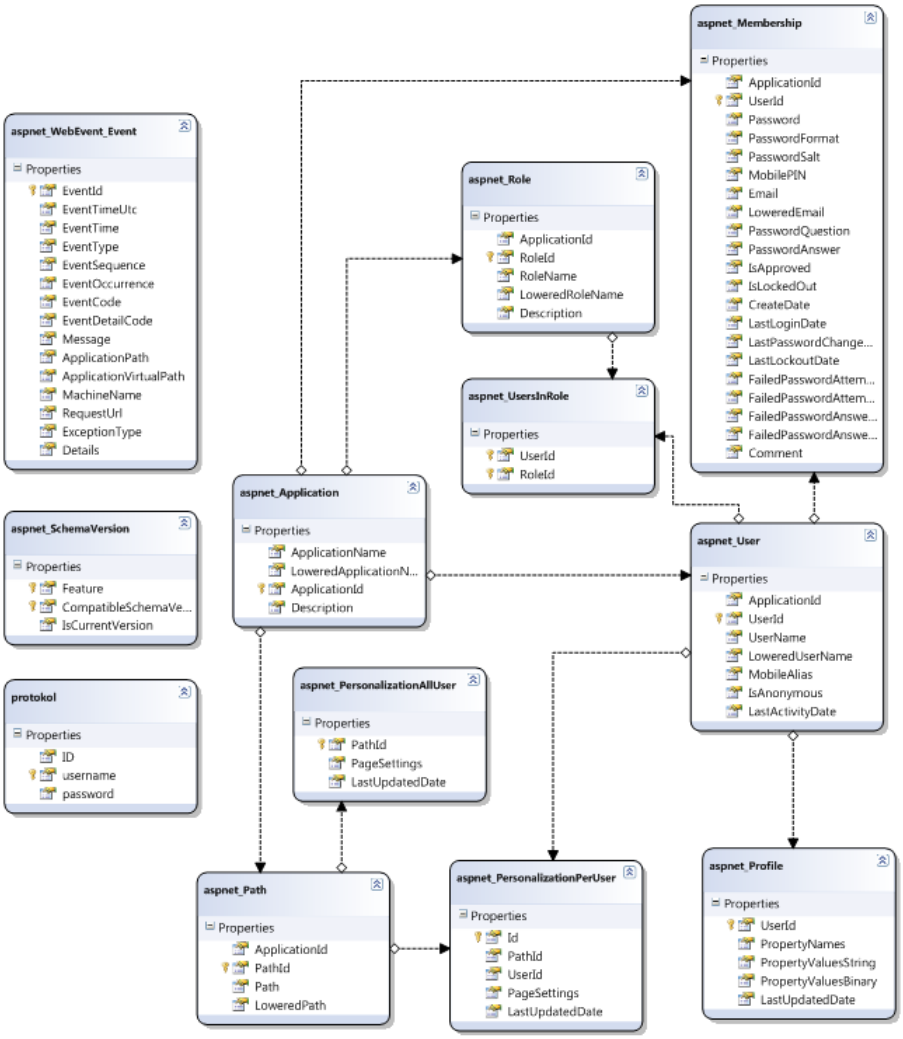
ASP.NET obsahuje model správy uživatelských účtů a oprávnění, který se skládá ze tří větších částí, které jsou definovány níže. Další tabulky obsahují různé události, stavy uživatelů a webu.

První částí je *Membership*. Tato část se stará o uživatelské účty, jména, hesla, jejich vytváření, změny a mazání. Stará se tedy o identifikaci a autentizaci uživatelů.

Druhou částí jsou *Role*. Každého uživatele lze přiřadit do několika různých „rolí“, podle nichž se poté provádí jeho autorizace. Pomocí rolí lze snadno přidělit práva (přístupu, zápisu, čtení, atd.) uživatele. Každý uživatel může být ve více rolích a každá role může obsahovat více uživatelů.

Třetí částí je *Profile*. Rozšiřuje možnosti uchovávání více dat o uživateli (jméno, adresa, telefon). Slouží pro přizpůsobení webu uživateli – využívá se např. u e-shopů.

ASP.NET obsahuje vestavěné providery – SqlMembershipProvider, SqlRoleProvider a SqlProfileProvider [5]. Ty standardně vytváří vlastní databázi a ukládají ji v adresáři *App_data*. Jak je vidět na obrázku 6.1, soubor ASPNETDB.MDF obsahuje vytvořenou databázi. Struktura této databáze a vzájemné provázání tabulek je znázorněno na obrázku 4.1.



Obr. 4.1: Výpis tabulek z databáze a jejich vztah mezi sebou.

Pro spojení aplikace s databází slouží direktiva `<connectionStrings>`. Celý kód je vidět v souboru web.config:

```
<connectionStrings>
    <add name="ASPNETDBConnectionString" connectionString="Data␣Source=.\↩
        SQLEXPRESS;AttachDbFilename=|DataDirectory|\ASPNETDB.MDF;Integrated␣↩
        Security=True;User␣Instance=True"
  providerName="System.Data.SqlClient" />
</connectionStrings>
```

Pro přihlášení pomocí QR kódu nebo uloženého tokenu je v databázi vytvořena tabulka *protokol*, která obsahuje uživatelské jméno a zašifrované heslo.

4.4 Vzorové stránky

Protože většina stránek webu má podobnou strukturu a liší se hlavně obsahem, lze využívat vizuální dědičnost pomocí vzorové stránky (master page). Ta může obsahovat libovolný počet webových ovládacích prvků a stylů HTML, které se poté aplikují na všechny další stránky webu (content pages). Pro vzory stránek se využívají zejména části hlavička (header) a patička (footer). Když koncový uživatel vyvolá jednu z podstránek, ve skutečnosti se dívá na zkompilovanou stránku z příslušné podstránky a vzorové stránky [1]. V ASP.NET se vzorová stránka dosahuje vytvořením stránky **.master*.

Ve webové aplikaci je vytvořený vzor stránky uložený v souboru MasterPage.master. Zde je každá část oddělená pomocí blokového elementu `<div>` s vlastním ID, které označuje blok stránky a umožňuje snadno formátovat celou část bloku, namísto každého prvku zvlášť.

V hlavičce stránky je zapotřebí mít webové ovládací prvky *<asp:LoginName>* a *<asp:LoginStatus>* pro informaci o přihlášeném uživateli.

```
<div id="loggedUser">
    <i>Přihlášený uživatel:</i>
    <asp:LoginName ID="LoginName1" runat="server" Font-Bold=true />
    <asp:LoginStatus ID="LoginStatus1" runat="server" />
</div>:
```

Vzorová stránka dále obsahuje menu *<asp:SiteMapDataSource>*.

```
<div id="menu">
    <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" ↩
        ShowStartingNode="false" />
    <asp:Menu ID="Menu2" runat="server" DataSourceID="SiteMapDataSource1">
    </asp:Menu>
</div>
```

Dynamický obsah stránky zajišťuje komponenta *<asp:ContentPlaceHolder>*.

```
<div id="content">
    <div id="innerContent">
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server"/>
    </asp:ContentPlaceHolder>
    </div>
</div>
```

V záhlaví je údaj o autorství stránky (copyright).

```
<div id="footer">
    &copy; Petr Vybíral
</div>
```

Struktura vzoru stránek je vidět na obr. 4.2.

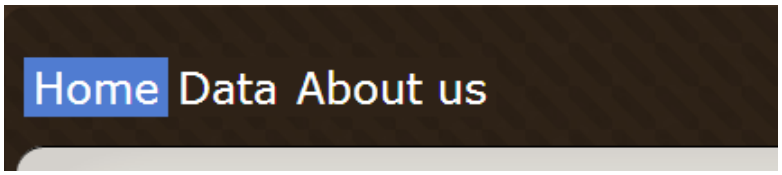
Obr. 4.2: Vzor stránky rozdělený do bloků.

4.5 Navigace po webu

Navigace je fundamentální komponenta každého webu. Hlavním cílem je přenést uživatele z jedné stránky do druhé za pomocí jednotného a přehledného navigačního systému. ASP.NET poskytuje vývojářům řadu navigačních ovládacích prvků, jedním z nich je mapa webu – Site map.

Mapa webu poskytuje navigační strukturu webu a umožňuje ji přímo svázat s webovými prvky. Výchozím bodem navigace je zprostředkovatel mapy webu – `XmlSiteMapProvider`, který generuje informace ze souboru XML [8].

Každé menu musí mít kořenovou složku (Root), od které se bude odvíjet stromová struktura stránek webu. Direktiva `SiteMapNode` značí položku menu, která



Obr. 4.3: Grafická podoba menu.

v sobě může mít libovolný počet elementů. Menu webové aplikace je znázorněno na obrázku 4.3.
Soubor pro vytvoření menu – **web.sitemap** obsahuje následující zdrojový kód:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~/" title="Root" description="">
        <siteMapNode url="~/Welcome.aspx" title="Home" description="" />
        <siteMapNode url="~/Users/Data.aspx" title="Data" description="" />
        <siteMapNode url="~/AboutUs.aspx" title="About us" description="" />
        </siteMapNode>
</siteMap>
```

4.6 Ověření uživatele pomocí formuláře

Formulářové ověřování je nejvíce užívaná metoda přístupu uživatelů k webu. Umožňuje umístit přihlašovací formulář přímo do aplikace, takže klient zadá své uživatelské jméno a heslo do určitého formuláře HTML zobrazeného v prohlížeči. Negativním aspektem tohoto ověřování je odesílání uživatelských jmen a hesel ve formě čitelného textu, pokud není zabezpečena komunikace protokolem HTTPS (TLS).

Ve stránce Login.aspx je přilašovací ovládací prvek *<asp:Login>* (obrázek 4.4), který slouží pro ověřování uživatele pomocí uživatelského jména a hesla. Tento prvek dále obsahuje odkaz na stránku Register.aspx, která slouží pro registraci uživatele pomocí komponenty *<asp:CreateUserWizard>*.

Pro ověřování pomocí formulářů je nutné implementovat následující kód v aplikačním souboru web.config, který specifikuje i stránku pro přihlášení:

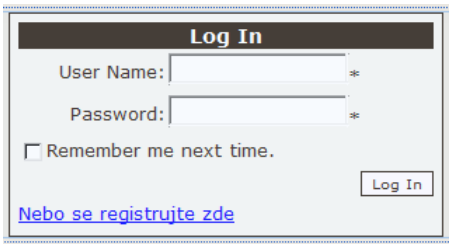
```
<authentication mode="Forms">
    <forms loginUrl="~/Login.aspx"/>
</authentication>
```

Následuje zabezpečení přístupu na stránku *Users/*Data.aspx, to se realizuje přidáním kódu:

```
<location path="~/Users">
    <system.web>
      <authorization>
        <deny users="?"/>
      </authorization>
    </system.web>
```

Kód způsobí, že anonymní uživatelé mají zakázaný přístup do adresáře, kde se daná stránka nachází. K této části mají přístup výhradně ověření uživatelé.

Pokusí-li se o přístup k aplikaci neověřený koncový uživatel, jeho požadavek je přesměrován na stránku (Login.aspx), kde jej lze ověřit a umožnit mu pokračování. Po zadání platných přihlašovacích údajů se uživatel vrátí na to místo, na které se má uložit prvek cookie využívaný k zachování přístupového tokenu ověřeného uživatele.



Obr. 4.4: Přihlašovací formulář.

4.7 Ověření atributovou funkcí

Žadatel může využít atributové autentizace pro přístup do zabezpečené sekce webové aplikace. Osoba žádající ověření se prokazuje vlastnictvím předmětu – tokenu. Token je generován klientskou aplikací podle protokolu HM12 (viz sekce 2.7). Jedná se o autentizaci uživatele bez prozrazení jeho identity.

Pro práci s velkými čísly je zavedena knihovna `System.Numerics.dll`, která podporuje datový typ `BigInteger` a výpočetní algoritmy s ním spojené. Klient i server využívá třídu Protokol.cs, která obsahuje metody pro výpočet příslušných parametrů protokolu HM12.

4.7.1 Klient

Klientská část je realizovaná konzolovou aplikací *Protokol_Client*, ve které je výstupem textový soubor, který uživatel uploaduje ve webové aplikaci na stránce Login.aspx. Textový soubor obsahuje autentizační informace, které se ověřují na serveru.

Žadateli je přidělen master key (w_{rr}, w_1 a w_2) a generační parametry (A_{seed}, g_1, g_2, g_3). Následně si vygeneruje náhodné bitové čísla fixních velikostí (r_1, r_2, r_3, r_S, K_S). Tato čísla použije pro výpočet autentizačního tokenu [4]. Jednosměrnou hašovací funkcí SHA-1 s výstupem 160 bitů vypočítá haš e. Poté dopočítá odpověď serveru (z_1, z_2, z_3, z_S).

Postup generování autentizačních informací žadatele:
Veřejné parametry: $A_{seed}, g_1, g_2, g_3, n$.
Uživatelský klíč (master key): w_1 a w_2 jsou náhodně generované parametry pro každého uživatele, w_{rr} je jedinečný parametr uživatele.
Nejdříve se vytvoří klíč relace K_S (náhodně vygenerované binární číslo o délce 79 bitů).

$$K_S \in_R \{0,1\}^{79}. \tag{4.1}$$

Poté se spočítají parametry podle uživatelského K_S

$$A = A_{seed}^{Ks} \ mod \ n. \tag{4.2}$$

Závazek C_1 uchovává příspěvek uživatele w_{RR}

$$C_1 = g_3^{Ksw_{rr}} \ mod \ n. \tag{4.3}$$

Závazek C_2 nese informaci o klíči K_S

$$C_2 = g_3^{Ks} \ mod \ n. \tag{4.4}$$

Následně se zvolí 3 velká náhodná bitová čísla příslušných velikostí r_1, r_2, r_3

$$r_1 \in_R \{0,1\}^{480}, \tag{4.5}$$

$$r_2 \in_R \{0,1\}^{400}, \tag{4.6}$$

$$r_3 \in_R \{0,1\}^{320}. \tag{4.7}$$

Dopočítají se čísla

$$\bar{A_{seed}} = g_1^{r_1} \ g_2^{r_2} \ g_3^{r_3} \ mod \ n, \tag{4.8}$$

$$\bar{A} = A_{seed}^{r_S} \ mod \ n, \tag{4.9}$$

$$\bar{C_1} = g_3^{r_3} \ mod \ n, \tag{4.10}$$

$$\bar{C_2} = g_3^{r_S} \ mod \ n. \tag{4.11}$$

Hašovací funkcí SHA-1 se vypočte haš s výstupem 160 bitů

$$e = H(A, \ \bar{A}, \ A_{seed}, \ \bar{A_{seed}}, \ C_1, \ C_2, \ \bar{C_1}, \ \bar{C_2}). \tag{4.12}$$

Před přenosem ověřovateli se spočítají parametry odpovědi na základě výstupního haše a uživatelských parametrů

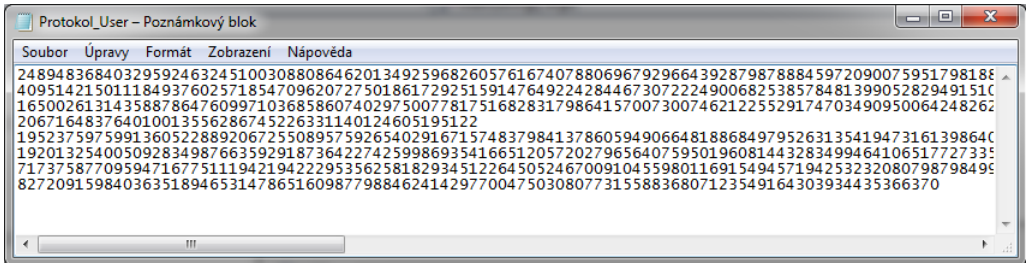
$$z_1 = r_1 - eK_S w_1, \tag{4.13}$$

$$z_2 = r_2 - eK_S w_2, \tag{4.14}$$

$$z_3 = r_3 - eK_S wrr, \tag{4.15}$$

$$z_S = r_S - eK_S. \tag{4.16}$$

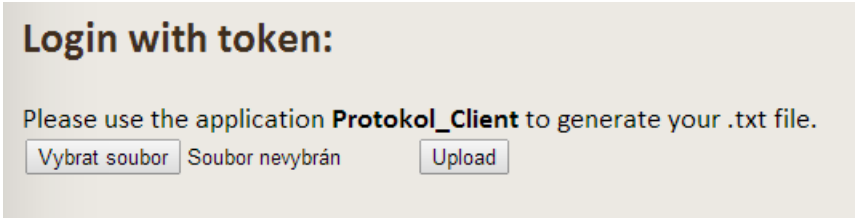
Po provedení výpočtů jsou autentizační parametry vygenerovány do textového souboru (obrázek 4.5 – na každém řádku jeden parametr).



Obr. 4.5: Výpis vygenerovaného textového souboru klienta.

4.7.2 Server

Ve stránce Login.aspx je umístěna komponenta *<asp:FileUpload/>* jak je vidět na obrázku 4.6, kde má uživatel možnost atributového ověření. Komponenta slouží pro nahrání souborů do webové aplikace. Ověření probíhá až po vyvolání události `Click` tlačítka *Upload*.



Obr. 4.6: Atributové ověření klienta.

Žadatel si vygeneruje klientskou aplikací textový soubor, ve kterém jsou obsaženy autentizační informace, který následně uploaduje na server.

Postup ověření žadatele:
Ověřovatel z přijatého textového souboru extrahuje čísla (A, C_1 , C_2), kontrolní součet (e) a odpověď (z_1, z_2, z_3, z_S). Poté si dopočítá čísla $\bar{A_{seed}}$, $\bar{A}$, $\bar{C_1}$ a $\bar{C_2}$.
Veřejné parametry: $A_{seed}, g_1, g_2, g_3, n$.
Výpočet požadovaných čísel:

$$\bar{A_{seed}} = A^e g_1^{z_1}\ g_2^{z_2}\ g_3^{z_3}\ mod\ n, \tag{4.17}$$

$$\bar{A} = A^e A_{seed}^{z_S}\ mod\ n, \tag{4.18}$$

$$\bar{C_1} = C_1^e g_3^{z_3}\ mod\ n, \tag{4.19}$$

$$\bar{C_2} = C_2^e g_3^{z_S}\ mod\ n. \tag{4.20}$$

Následuje výpočet haše z vypočtených čísel. Pokud haš souhlasí s přijatým, je proces autentizace úspěšně dokončen a uživatel ověřen.

$$e = H(A,\ \bar{A},\ A_{seed},\ \bar{A_{seed}},\ C_1,\ C_2,\ \bar{C_1},\ \bar{C_2}). \tag{4.21}$$

4.8 Ověření pomocí QR kódu

Webová aplikace rozšiřuje možnost autentizace o ověření pomocí QR kódu. Do QR kódu jsou zapsána autentizační data, která se po sejmutí odesílají na server. Autentizační data, založená na protokolu HM12, jsou vygenerována v mobilní aplikaci telefonu využívající operační systém Android. Klientská aplikace *Protokol_QR_Webcam_Client* slouží pro sejmutí QR kódu interní nebo externí webkamerou. Pro zaslání načtených dat je vytvořena socketová komunikace mezi serverem a klientem využívající protokol TCP. Server jako odpověď posílá náhodně vygenerovaný řetězec, kterým se žadatel musí prokázat při ověřování společně s QR kódem.

4.8.1 Klient

Klientská část pracuje v prostředí Windows Forms (WinForms), které poskytuje GUI (grafické uživatelské rozhraní – Graphical User Interface). Aplikaci tvoří dvě hlavní části, první je sejmutí QR kódu a druhá je vytvoření spojení se serverem. QR kód je vygenerován aplikací v telefonu podle autentizačního protokolu HM12.

Postup vytvoření autentizačních dat
Veřejné parametry: $A_{seed}, g_1, g_2, g_3, n$.
Uživatelský klíč (master key): w_1 a w_2, w_{rr}.
Postup vygenerování potřebných parametrů je podle výpočtů (4.1) až (4.11).
Odlišností od předchozího generování informací (4.7.1) je přidání časové známky.

Známka (timeStamp) je časový údaj v sekundách od 1.1.1970 00:00 do aktuálního času. Známka se společně s parametry přidá do vstupu hašovací funkce. Před konvertováním jsou vstupní data převedena do 8bitového formátu ASCII (American Standard Code for Information Interchange).

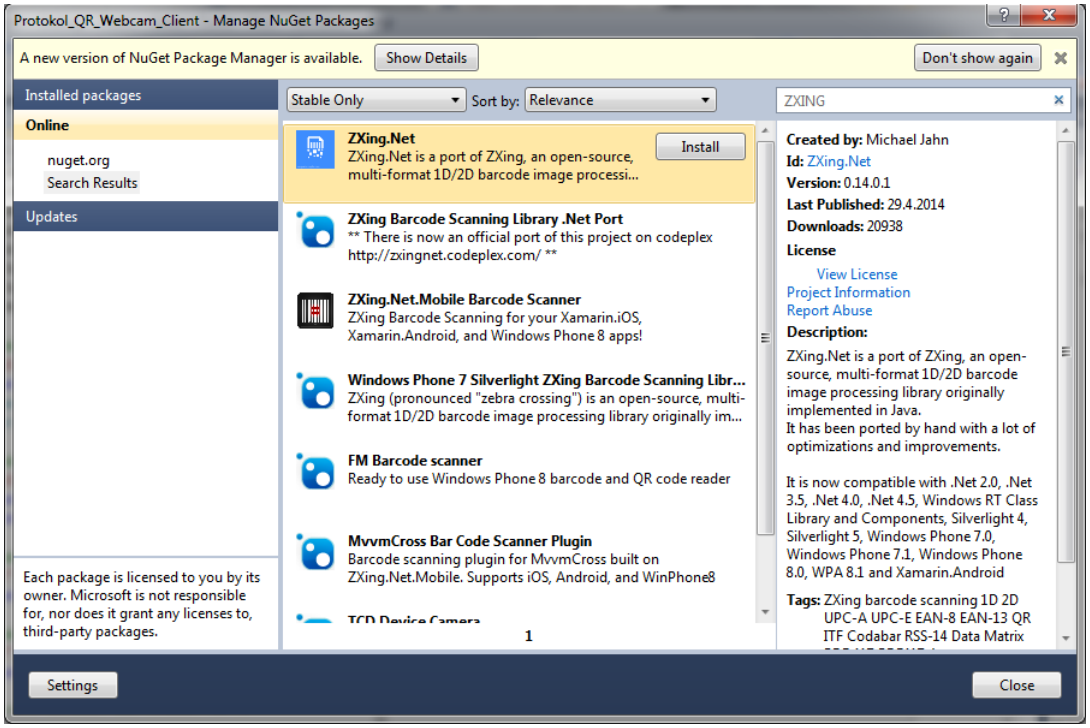
$$e = H(A,\ \bar{A},\ A_{seed},\ \bar{A_{seed}},\ C_1,\ C_2,\ \bar{C_1},\ \bar{C_2},\ timeStamp) \tag{4.22}$$

Vypočítá se odpověď serveru podle výpočtů (4.13) až (4.16).
Formát informačních dat QR kódu je následující:

```
"1_" + zs.ToString() + "_", z3.ToString() + "_", z2.ToString() + "_",
z1.ToString() + "_", e.ToString() + "_", C2.ToString() + "_", C1.ToString()
+ "_", A.ToString() + "_";
```

Sejmutí QR kódu pomocí webkamery

Klientská aplikace využívá externí open-source knihovnu ZXING [21] pro práci s QR kódy. Tato knihovna umožňuje zpracování a dekódování QR kódů pomocí webkamery. Přidání knihovny pomocí NUGET balíčku je znázorněno na obrázku 4.7.



Obr. 4.7: Přidání knihovny ZXING do projektu klientské aplikace.

Rozhraní webkamery využívá pro své rozhraní komponentu *<asp:PictureBox>* a pro výpis dat komponentu *<asp:RichTextBox>*. Třída WebCam.cs obsahuje metody pro práci s webkamerou.

Zjednodušený princip zachycení QR kódu je popsán níže.

Uživatel stiskne tlačítko pro aktivování webkamery. Následuje vytvoření objektu třídy `WebCam` s parametrem výstupního rozhraní.

```
wCam = new WebCam { Container = picWebCam };
```

Využije se metoda `Load()`, která zkontroluje připojení externího záznamového zařízení.

```
wCam.Load();
```

Následně je vyvolána událost `OpenConnection()`, která si zjistí ID záznamového hardwaru a spustí zachytávání.

```
wCam.OpenConnection();
```

Nyní se aktivuje časovač, který stanoví interval (100 ms) zachycení snímků.

```
webCamTimer = new Timer();
webCamTimer.Tick += webCamTimer_Tick;
webCamTimer.Interval = 100;
webCamTimer.Start();
```

Z předešlého kódu je vidět spuštění dalšího vlákna `webCamTimer_Tick`, které má za úkol najít a dekódovat QR kód obsažený v bitmapovém zachyceném snímku webkamery.

```
void webCamTimer_Tick(object sender, EventArgs e)
{
    var bitmap = wCam.GetCurrentImage();
    if (bitmap == null)
        return;
    var reader = new BarcodeReader();
    var result = reader.Decode(bitmap);
    if (result != null)
        txtTypeWebCam.Text = result.BarcodeFormat.ToString();
        DataRTB.Text = result.ToString();
        webCamTimer.Stop();
        webCamTimer = null;
        wCam.Dispose();
        wCam = null;
```

Nejdříve probíhá načtení aktuálního snímku z webkamery. Poté je snímek analyzován pomocí události `Decode` knihovny ZXING [21]. Pokud metoda vrátí nenulový výsledek analýzy, výsledek se vypíše do *RichTextBoxu* a jsou vyvolány události pro uvolnění prostředků webkamery.

Vytvoření socketové komunikace klienta

Úkolem serveru je nastavit koncový bod pro klienty a pasivně čekat na spojení. Klient musí znát IP adresu (hostname) počítače a číslo portu, na kterém pracuje

Obr. 4.8: Načtení QR kódu vygenerovaného mobilní aplikací.

server. Jakmile je navázáno spojení mezi serverem a klientem, mohou komunikovat přes jejich vlastní sockety. Klient odesílá autentizační informace načtené z QR kódu k ověření na server [18].

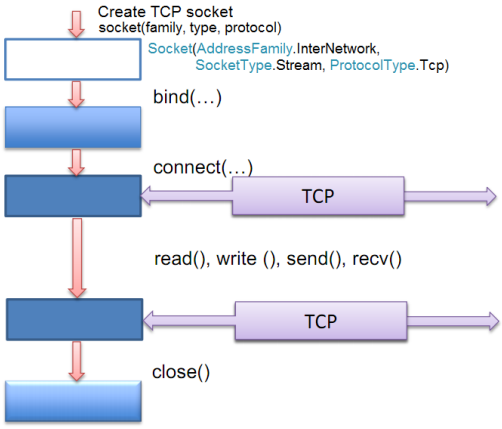
Hlavní třídy pro práci se sockety:

- `System.Net.Sockets` – jmenný prostor nabízí řízené plnění Windows Sockets (Winsock), rozhraní pro vývojáře, kteří potřebují pevně řídit přístup do sítě. TcpClient, TcpListener a UdpClient třídy zapouzdřují podrobnosti o vytvoření TCP a UDP připojení k internetu.
- `TCPListener` – specifikuje lokální adresu a port, poté volá metodu Start(). Tento socket čeká na příchozí spojení na určitém portu. Opakovaně volá metodu AcceptTcpClient() pro další příchozí klienty.
- `TCPClient` – poskytuje jednoduché metody pro připojení k odesílání a přijímání dat přes TCP spojení. Metoda GetStream() umožňuje přístup k NetworkStreamu, tedy k abstraktnímu odesílání a přijímání dat.
- `NetworkStream` – poskytuje základní proud dat pro přístup k síti. Využívá komunikace serveru s klientem pomocí read() a write() metody [9].

Klientská aplikace využívá třídu Connection.cs, která obsahuje metody pro vytvoření spojení, ukončení spojení, odesílání a přijímání informací:

```
socClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ↩
    ProtocolType.Tcp);
```

Vystupují zde funkce pro nastavení socketu klienta. Nejprve je to `AddressFamily`, která se používá pro místní síť, kde se pracuje s protokolem IPv4. Poté se definuje



Obr. 4.9: Příklad vytváření socketu TCP klientem.

`SocketType`, díky kterému probíhá komunikace obousměrně (Stream). Na konci je uveden `ProtocolType`, který nastavuje práci s protokolem TCP transportní vrstvy TCP/IP modelu [12].

Kód níže slouží pro nastavení koncového bodu, tedy socketu klienta pro připojení k asynchronnímu serveru.

```
System.Net.IPAddress remoteIPAddress = System.Net.IPAddress.Parse(IPAddres);
System.Net.IPEndPoint remoteEndPoint = new System.Net.IPEndPoint(↩
    remoteIPAddress, alPort);
```

Událost pro vytvoření připojení k serveru.

```
socClient.Connect(remoteEndPoint);
```

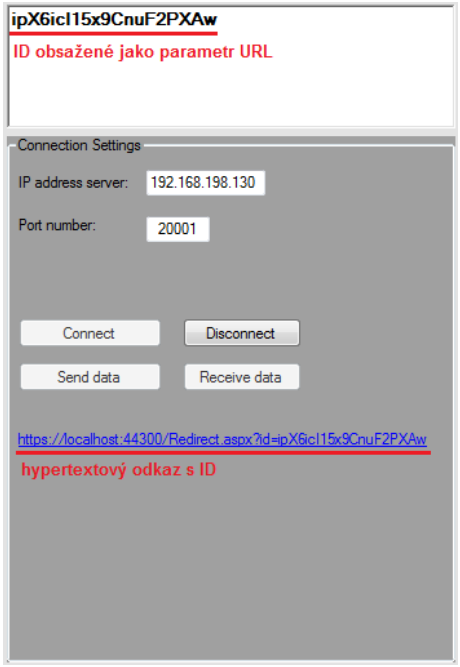
Po připojení klienta k serveru může začít vlastní komunikace. Uživatel odesílá data při vyvolání následujícího kódu.

```
byte[] byData = System.Text.Encoding.ASCII.GetBytes(Data.ToString());
socClient.Send(byData);
```

Data jsou před odesláním převedena do ASCII kódu. Pomocí metody `Send` jsou data odeslána na server. Obdobně je tomu i pro příjem dat.

```
byte[] buffer = new byte[1024];
int SUM = socClient.Receive(buffer);
char[] chars = new char[SUM];
System.Text.Decoder data = System.Text.Encoding.UTF8.GetDecoder();
int charLen = data.GetChars(buffer, 0, SUM, chars, 0);
System.String pdata = new System.String(chars);
```

Nejdříve se vytvoří vyrovnávací pamět pro příchozí data, poté začne klient přijímat data pomocí metody `Receive` a na závěr se převede bytové pole na řetězec. Výsledná data jsou převedena zpět na formát UTF-8.

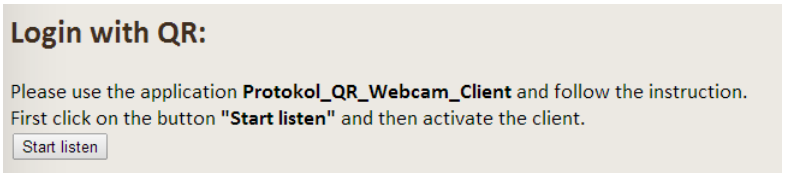


Obr. 4.10: Průběh komunikace se serverem.

4.8.2 Server

Na přihlašovací stránce Login.aspx nalezne uživatel možnost ověření pomocí QR kódu, jak je zobrazeno na obrázku 4.11.

Při stisknutí tlačítka *Start listen* je volána funkce ke generování jednorázového ID klienta. ID se zapíše do objektu `Session`. Poté je vytvořen objekt třídy Connections.cs, která obsahuje metody pro nastavení socketové komunikace.



Obr. 4.11: Přihlášení pomocí QR kódu, zahájení naslouchání.

```
Code ses = new Code();
string myId = ses.GetRandomAlphanumericString(20);
Session["ID"] = myId;
Connections server = new Connections();
server.Listen(myId);
ListenBtn.Enabled = false;
```

Server je postaven s asynchronními sockety, takže serverová aplikace není pozastavena, zatímco čeká na spojení od klienta. Děděné funkce třídy `System.Net.Sockets`

využívané v asynchronní komunikaci na straně serveru [18]:

- `AddressFamily` – určuje adresové schéma, které instance socket-třídy mohou používat.
- `SocketType` – určuje typ socketu.
- `ProtocolType` – určuje protokoly, které socket-třída podporuje.
- `AsyncCallback` – odkazuje na metodu, která bude volána, když odpovídající asynchronní operace bude dokončena.
- `IPEndPoint` – představuje koncový bod sítě jako je IP adresa a číslo portu.
- `IAsyncResult` – představuje stav asynchronní operace [12].

Server asynchronně přijme klientská data a vytvoří odpověď v podobě požadavku URL, do kterého zapíše klientovo ID. Poté se odesílá odpověď klientovi. Uživatel je následně přesměrován na server, kde dochází k ověření ID a obsahu přijatých dat.

Vytvoření socketové komunikace serveru

První je volána metoda `Listen`, třídy Connections.cs, která obsahuje následující kód.

```
listenSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ↩
    ProtocolType.Tcp);
int fPort = System.Convert.ToInt32(port, 10);
IPEndPoint localIP = new IPEndPoint(IPAddress.Any, fPort);
listenSocket.Bind(localIP);
listenSocket.Listen(4);
listenSocket.BeginAccept(new AsyncCallback(ClientConnect), null);
```

Nastavení socketu se uskutečňuje pomocí `AddressFamily`, která pracuje s místní sítí využívající protokol IPv4. Poté následuje definice protokolu `SocketType`, která určuje transportní protokol TCP. Funkce `Bind` asociuje lokální IP adresu a socket. Příkazem `Listen` začíná naslouchání serveru, kde parametrem je fronta (4 – maximální počet souběžně připojených klientů). Nyní se mohou klienti připojit pomocí `BeginAccept`. Začíná asynchronní operace přijetí klientovy žádosti o připojení. `AsyncCallback` – odkaz na metodu `ClientConnect`, která bude volána, když bude odpovídající asynchronní operace dokončena.

```
asyncResult = listenSocket.EndAccept(asyn);
Data(asyncResult);
```

Funkce `.EndAccept` přijímá asynchronně příchozí pokusy o připojení a vytváří nový socket pro komunikaci s klientem. Po navázání spojení s klientem začíná vlastní přenos dat. Nastavení socketu dat je následovné.

```
if (asyncCallBack == null)
{
    asyncCallBack = new AsyncCallback(DataReceived);
}
SocketPacket SocPacketData = new SocketPacket();
SocPacketData.thisSocket = soc;
soc.BeginReceive(SocPacketData.dataBuffer, 0, SocPacketData.dataBuffer.Length↩
    , SocketFlags.None, asyncCallBack, SocPacketData);
```

V kódu je nejdůležitější metoda `.BeginReceive`, která má parametry [12]:

- *buffer* – pole typu byte, kde je umístění pro ukládání přijatých dat,
- *offset* – nultá pozice v bufferu, od kterého začíná uložení dat,
- *size* – počet bytů pro příjem dat,
- *socketFlags* – bitová kombinace SocketFlags hodnot,
- *callback* – zpětné volání `AsyncCallback` delegáta, který odkazuje na metodu `Invoke`, když je dokončena operace,
- *state* – uživatelem definovaný objekt, který obsahuje informace o přijímaném provozu. Tento objekt je předán `EndReceive` po dokončení operace.

Nyní nastává samotný příjem dat.

```
SocketPacket theSockId = (SocketPacket)asyn.AsyncState;
int SUM = 0;
SUM = theSockId.thisSocket.EndReceive(asyn);
char[] chars = new char[SUM + 1];
System.Text.Decoder dec = System.Text.Encoding.ASCII.GetDecoder();
BigInteger charslen = dec.GetChars(theSockId.dataBuffer, 0, SUM, chars, 0);
System.String szData = new System.String(chars);
Save(szData);
Data(asyncResult);
```

Metoda `EndReceive` ukončí asynchronní čtení dat. Přijatá data se dokódují ASCII kódem. `GetChars` dekóduje posloupnost bytů z určeného bytového pole do určeného typu – `BigInteger`. Následně se volá metoda `Save`, která dočasně uloží přijatá data na server. Metoda využívá objekt `StreamWriter`, kde prvním parametrem je cesta do kořenového adresáře serveru a druhým parametrem je návratová hodnota.

```
using (System.IO.StreamWriter file = new System.IO.StreamWriter("C:/Users/↩
    VPHP4540S/Documents/Visual␣Studio␣2010/WebSites/WebSite2/Txt/Protokol_QR.↩
    txt", false))
{
    file.WriteLine(myData);
}
    SendData();
```

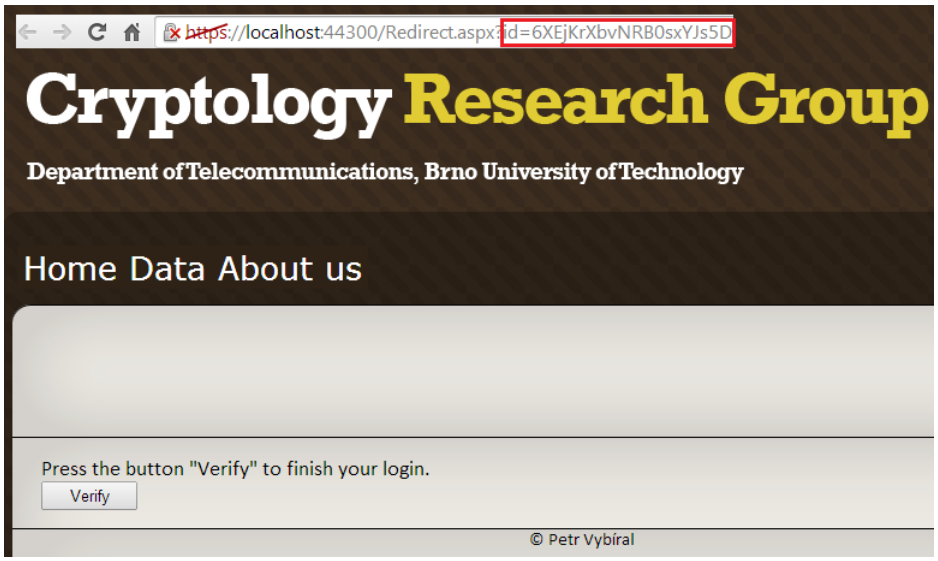
Po uložení přijatých dat je volána metoda `SendData`, která vytvoří odpověď klientovi.

```
byte[] byData = System.Text.Encoding.ASCII.GetBytes(code);
asyncResult.Send(byData);
```

Odpověď obsahuje vygenerované klientské ID. Nejprve se data převedou do bytového pole a poté odešlou metodou `Send`.

Postup ověření žadatele

K ověření uživatele slouží stránka Redirect.aspx (obrázek 4.12), na kterou je uživatel přesměrován z klientské aplikace. Uživatel předává své informace pomocí dotazovacího řetězce (query string) v URL. Dotazovací řetězec je ta část URL, která následuje za znakem otazník. Předností dotazovacího řetězce je to, že nijak nezatěžuje server a dá se snadno transportovat z jedné stránky na druhou.



Obr. 4.12: Stránka pro dokončení ověření klienta.

Server ověří nejdříve klienské ID z parametru URL.

```
string query = Request.QueryString["id"];
    if((string)Session["ID"] == query)
```

Pokud ID souhlasí s uloženým objektem `Session`, následuje ověření přijatých dat.

Ověřovatel z přitatých dat QR kódu extrahuje čísla (A, C_1 , C_2), kontrolní součet(e) a odpověď (z_1, z_2, z_3, z_S).
Veřejné parametry: $A_{seed}, g_1, g_2, g_3, n$.
Následně si ověřovatel dopočítá parametry podle vzorců (4.17) až (4.20).
Server si vygeneruje časovou zmámku *timeStamp*. Časovou známku vloží do vstupu haše (vzorec (4.22)) a následně jej porovná s přijatým hašem. Pokud se rovnají, uživatel je ověřen. Pokud se haše nerovnají, je hodnota časové známky snižována v časovém intervalu dvou minut, tedy 120 sekund.

4.8.3 Průběh komunikace mezi klientem a serverem

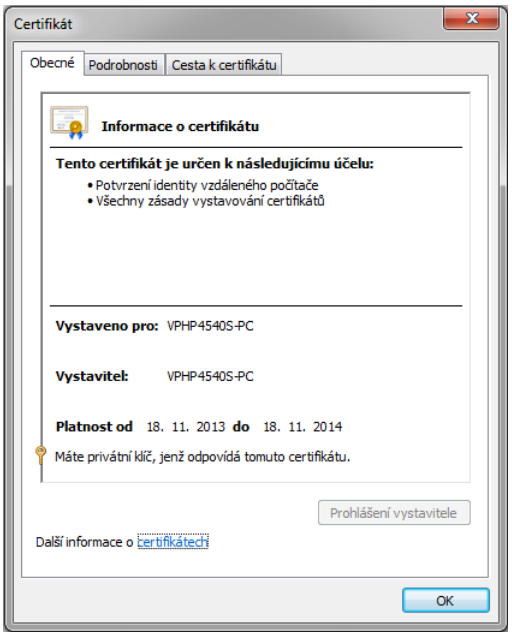
1. Klient ve stránce Login.aspx vytvoří požadavek, kterým začne naslouchání serveru, jak je vidět na obrázku 4.11.
2. Uživatel aktivuje webkameru a před snímač předloží mobilní telefon s vygenerovaným kódem. Po zachycení QR kódu je uživateli přístupný panel pro nastavení připojení se serverem. Sejmutí QR kódu je znázorněno na obrázku 4.8.
3. V panelu nastavení připojení vytvoří uživatel socketové spojení se serverem pomocí IP adresy serveru a portu, na kterém server naslouchá. Klient následně odesílá informace dekódované z QR kódu na server.
4. Server asynchronně přijímá požadavky a po přijetí informací od klienta vytvoří jednorázovou odpověď. Odpověď je obratem zasílána klientovi a má podobu URL (Uniform Resource Locator) odkazu na ověřovací stránku. Součástí URL je parametr (query string) s údajem vygenerovaného ID klienta.
5. Klient přijme odpověď a vypíše adresu URL serveru, na kterou se má přesměrovat. Příchozí odpovědi vypsané v klientské aplikaci jsou vidět na obrázku 4.10.
6. Server po vyvolání události uživatele ověří z URL adresy parametr klientského ID. Následně ověří příchozí data podle protokolu HM12, jak je uvedeno v části 4.8.2.

4.9 Zabezpečení protokolem HTTPS

Protokol HTTPS je popsán v části 2.6. Pro využití protokolu HTTPS je zapotřebí získat certifikát serveru a nastavit průběh komunikace na příslušném zabezpečeném portu.

4.9.1 Vytvoření certifikátu

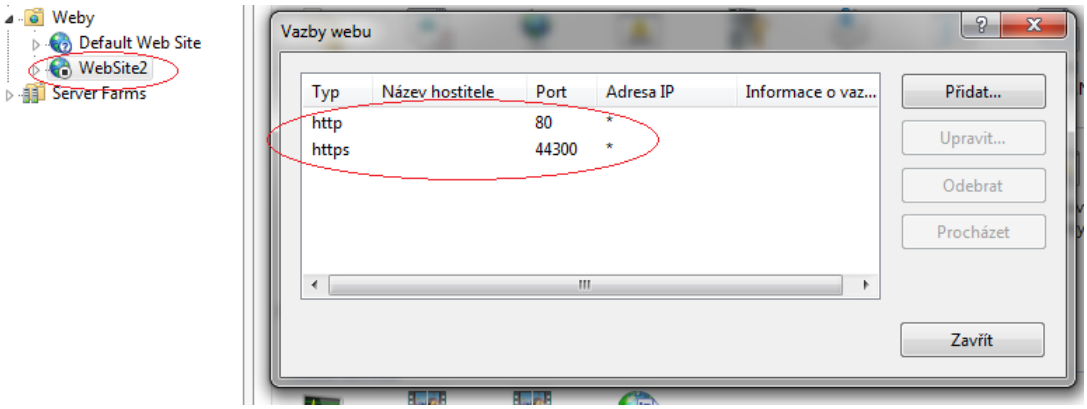
Certifikát je digitálně podepsaný veřejný šifrovací klíč, který vydává certifikační autorita [2]. Pro vygenerování certifikátu a nastavení komunikace v síti localhost slouží ISS (Internet Information Service). Certifikát je podepsán sám sebou, proto prohlížeč upozorní na možnou hrozbu. Vlastnosti vygenerovaného certifikátu jsou vidět na obrázku 4.13.



Obr. 4.13: Vygenerovaný certifikát.

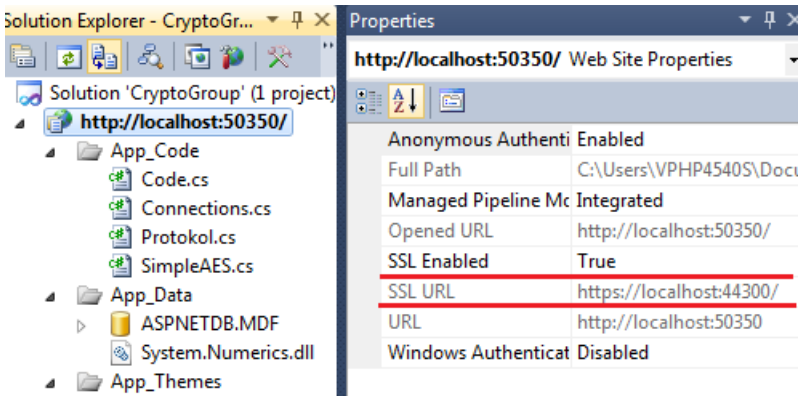
4.9.2 Zabezpečení komunikace protokolem HTTPS

Protokol HTTPS pracuje výhradně na portu 443. Pro průběh veškeré komunikace je zvolen volný port 44300, kterému je nastaveno používání výhradně protokolu HTTPS.



Obr. 4.14: Nastavení portů pro komunikaci v IIS.

V konfiguračním souboru webové aplikace je nutné definovat výstupní port pro zabezpečenou komunikaci. Na obrázku 4.15 je vidět použití protokolu SSL a nastavení výstupního portu na 44300.



Obr. 4.15: Nastavení portů pro komunikaci ve VS 2010.

Pro spuštění aplikace pod portem 44300 je nutné zadat adresu do prohlížeče v následující podobě: https://localhost:44300/Default.aspx.

5 ZÁVĚR

V této práci jsou představeny různé zabezpečovací techniky pro autentizaci uživatelů a zabezpečení přenosu informací. Vytvořená webová aplikace (*CryptoGroup*) slouží pro formulářovou a atributovou autentizaci uživatelů.

Formulářová autentizace je typu výzva – odpověď. Uživatel prokazuje znalost uživatelského jména a hesla. Server ověřuje znalost shodou v databázi, jak je uvedeno v sekci 4.6.

Atributová autentizace je realizována pomocí protokolu HM12, který je vyvíjen na VUT v Brně. Protokol umožňuje anonymní autentizaci s prokázáním osobních údajů. Protokol je založen na konceptu nulové znalosti a závazkových schématech.

Ověření atributové funkce se uskutečňuje za pomocí klientské aplikace. Aplikace s využitím protokolu HM12 vygeneruje uživatelské atributy a unikátní klíč (uložené tokeny). Po využití jednosměrné hašovací funkce jsou data převedena na haš zprávy. Poté aplikace dopočítá odpověď pro server. Potřebné parametry jsou uloženy do textového souboru, který uživatel uploaduje na server. Ověřovatel si soubor uloží a extrahuje z něj potřebná autentizační data. Vypočítané parametry se svými tokeny spojí do bytového pole. Pole je vstup hašovací funkce, která vytvoří haš a ten následně porovná s přijatým hašem od uživatele. Pokud haš souhlasí, je proces autentizace úspěšný. Klientská část je realizována konzolovou aplikací *Protokol_Client*. Celý proces autentizace je popsán v části 4.7.

Ověření pomocí QR kódu je realizováno s využitím WinForm klientské aplikace (*Protokol_QR_Webcam_Client*) a mobilní aplikace (*QRgenWeb*). Generování QR kódu probíhá v mobilním zařízení v aplikaci, kterou vytvořil Michal Celeng. QR kód je snímán webkamerou s pomocí externí open-source knihovny ZXING, která podporuje generování a dekódování čárových kódů. Knihovna je limitována v zachycení QR kódu o maximální verzi 24. Knihovna ZXING poskytuje své metody pro práci s webovou kamerou, čímž je způsobeno omezení verze QR kódu.

Po zachycení QR kódu je mezi klientem a serverem vytvořena komunikace, která je popsána v části 4.8.3. Klient prokazuje vlastnictví přístupového atributu díky vytvořeným autentizačním datům a přidělenému klientskému ID. Postup ověření klienta je podobný jako u atributového ověřování s tím rozdílem, že se používá časová známka. Známka umožní expiraci QR kódu do dvou minut. Kód musí být platný od vygenerování přes sejmutí až po ověření.

Zabezpečení kanálu komunikace je realizováno pomocí protokolu HTTPS (TLS). Spojení mezi serverem localhost a klientem je šifrováno 128bitovým šifrováním. Využívá se vygenerovaného certifikátu (více v části 4.9).

Atributová autentizace je daleko mocnější a bezpečnější způsob ověřování uživatele než formulářová autentizace. Fakt, že žadatel nemusí prozradit svoji identitu

k ověření má velký potenciál. Autentizace pomocí QR kódu poskytuje bezpečný způsob ověření, který využívá i společnost Google při dvoufaktorové autentizaci. Nevýhodou autentizace pomocí QR kódu je klienstká aplikace, ve které musí uživatel vytvářet spojení se serverem. Možnost vylepšení by spočívalo ve využití technologie Silverlight. Technologie Silverlight umožňuje pracovat s webkamerou přímo ve webové aplikaci, ale není vhodná pro práci s databázemi. Z tohoto hlediska nebyla technologie použita v řešení práce. Prvky cookies slouží jako vhodné datové úložiště, bohužel je nelze vytvářet v klientských aplikacích, a proto nejsou využity v této práci.

Všechny dosažené výsledky byly simulovány ve virtuálním stroji VMware. Virtuální stroj obsahuje IIS, SQL server a technologii ASP.NET. Simulován byl webový server, klientské programy a jejich vzájemná spolupráce. Záznam simulace je přiložený k výsledkům v souboru Instruction.avi.

LITERATURA

[1] ANDERSON, Richard. *Active Server Pages 3.0. profesionálně.* Vyd. 1. Překlad David Brůha. Praha: Computer Press, 2000, 1164 s. ISBN 80-860-9747-1.

[2] BURDA, Karel. *Bezpečnost informačních systémů.* Brno, 1.11.2005. Skripta. FEKT Vysoké učení technického v Brně.

[3] EASTLAKE, D. 3rd; JONES P. *US Secure Hash Algorithm 1 (SHA1)*, RFC 3174, September 2001.

[4] HAJNÝ, Jan; MALINA Lukáš. *Unlinkable Attribute-Based Credentials with Practical Revocation on Smart- Cards. In Smart Card Research and Advanced Applications.* Lecture Notes in Computer Science. LNCS. Berlin: Springer-Verlag, 2013. s. 62-76. ISBN: 978-3-642-37287- 2. ISSN: 0302- 9743.

[5] HERCEG, Tomáš; Tomáš JECHA. *Píšeme webovou aplikaci v asp.net krok za krokem (část 1).* www.dotnetportal.cz [online]. 2008 [cit. 2013-12-14]. Dostupné z: http://www.dotnetportal.cz/clanek/105/Piseme-webovou-aplikaci-v-ASP-NET-krok-za-krokem-cast-1-

[6] HORÁLEK, Josef Jan. *Sockety* [online]. [s.l.], 2011. 12 s. Přednáška. FIM Univerzita Hradec Králové. Dostupné z WWW:<http://www.horalek.org/fim/OS2/Lecture08.pdf>.

[7] ISO/IEC 18004. *ISO/IEC 18004:2000(E).* Pittsburgh, 2000. Dostupné z: http://raidenii.net/files/datasheets/misc/qr_code.pdf

[8] MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. *ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 1.* Vyd. 1. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 880 s. Encyklopedie Zoner Press. ISBN 978-80-7413-131-8.

[9] MAKOFSKE, David B.; DONAHOO, Michael J.; CALVERT, Kenneth L. *TCP/IP Sockets in C# : Practical Guide for Programmers.* San Francisco : Elsevier, 2004. 188 s. ISBN 0-12-466051-7.

[10] MAO, Wenbo. *Modern Cryptography: Theory and Practice.* 5th ed. Upper Saddle River: Prentice Hall, 2004, 707 s. ISBN 01-306-6943-1.

[11] MRNKA, Ladislav. *Bezpečnost v ASP.NET.* In: Programování v prostředí .NET [online]. 2007 [cit. 2014-05-17]. Dostupné z: http://herakles.zcu.cz/education/net/lectures/ASPNET_security.pdf

[12] MSDN. *Microsoft Developer Network.* www.msdn.microsoft.com [online] 2014 [cit. 2014-05-17]. Dostupné z: http://msdn.microsoft.com/cs-cz

[13] PAYNE, Chris. *Naučte se ASP.NET za 21 dní.* Vyd. 1. Překlad David Brůha. Praha: Computer Press, 2002, xxii, 763 s. ISBN 80-722-6605-5.

[14] PROSISE Jeff. *Programování v Microsoft .NET: webové aplikace v .NET Framework, C# a ASP.NET.* Vyd. 1. Brno: Computer Press, 2003, 712 s. ISBN 80-722-6879-1.

[15] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z.* 2. aktualiz. vyd. Brno: Computer Press, 2006, 430 s. ISBN 80-251-1278-0.

[16] SCHNEIER, B. *Applied Cryptography. Vyd. 1.* Překlad David Brůha. New York: Computer Press, 1996, 758 s. ISBN 04-711-1709-9.

[17] STALLINGS, William. *Cryptography and network security: principles and practice.* Vyd. 7. Upper Saddle River: Pearson Education, 2007, xix, 731 s. ISBN 01-333-5469-5.

[18] VYBÍRAL, Petr *Webové aplikace a webové služby v C#: semestrální projekt.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011. 46 s. Vedoucí práce byl doc. Ing. Ivo Lattenberg, Ph.D.

[19] YANG, Herong *SHA1 Message Digest Algorithm Overview.* Cryptography Tutorials - Herong's Tutorial Examples [online]. 2012, 2013 [cit. 2013-12-12]. Dostupné z: http://www.herongyang.com/Cryptography/SHA1-Message-Digest-Algorithm-Overview.html

[20] ZEMAN, Václav. *MVDP přednáška 8. 2013/14: Kryptografické metody zabezpečení datových přenosů [online].* 2013, 1 - 56 [cit. 2013-20-12]. Dostupné z: https://www.vutbr.cz/elearning/file.php/133982/2013/MVDP_8_2013_sifrovani.pdf

[21] *ZXing.Net* [online]. 2014 [cit. 2014-05-18]. Dostupné z: http://zxingnet.codeplex.com/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AJAX Asynchronous JavaScript and XML

AES Advanced Encryption Standard

ASCII American Standard Code for Information Interchange

ASP aktivní serverové stránky – Active Server Pages

CLR Common Language Runtime

CSS kaskádové šablony stylů – Cascading Style Sheets

EIC Extended Channel Interpretation

GUI grafické uživatelské rozhraní – Graphical User Interface

HTML hypertextový značkovací jazyk – HyperText Markup Language

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

IP Internet Protocol

ISS Internet Information Service

LINQ Language Integrated Query

MVC Model View Controller

NIST National Institute of Standards and Technology

NSA National Security Agency

PKI infrastruktura veřejných klíčů – Public Key Infrastucture

QR Quick Response

SHA Secure Hash Algorithm

SQL Structured Query Language

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

URL Uniform Resource Locator

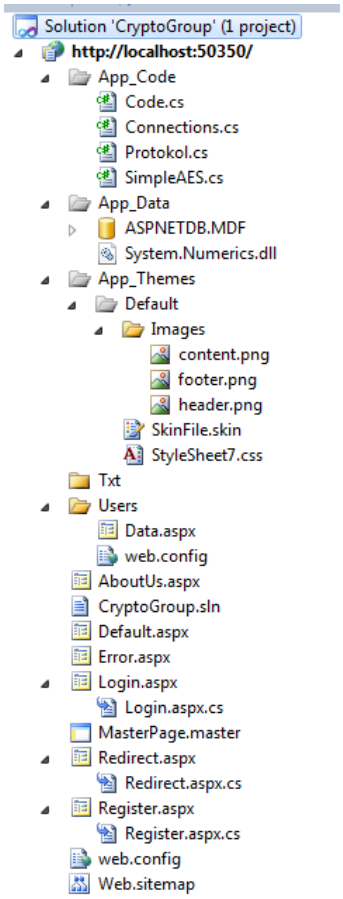
XHTML Extensible HyperText Markup Language

XML Extensible Markup Language

6 PŘÍLOHY

6.1 Struktura webové aplikace

Na obrázku 6.1 je vidět výpis souborů webové aplikace *Cryptogroup*.



Obr. 6.1: Struktura projektu webové aplikace.

6.2 Seznam souborů na přiloženém CD

Všechny programy byly vytvořeny pomocí Microsoft Visual Studia 2010.

1. **Webová aplikace (server)**
   - *CryptoGroup* - obsahuje všechny použité soubory ve webové aplikaci.
2. **Konzolová aplikace (klient)**
   - *Protokol_Client* - projekt konzolové aplikace pro vytváření autentizačního tokenu s výstupen do textového souboru.
3. **WinForm aplikace (klient)**
   - *Protokol_QR_Webcam_Client* - projekt klientské aplikace s GUI pro sejmutí QR kódu a ověření na serveru.