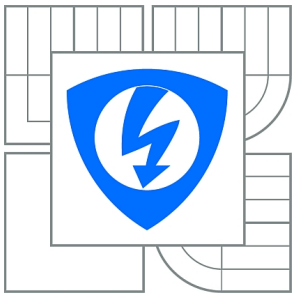
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

KRYPTOGRAFIE NA VÝPOČETNĚ OMEZENÝCH ZAŘÍZENÍCH

CRYPTOGRAPHY ON COMPUTATIONALLY LIMITED DEVICES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

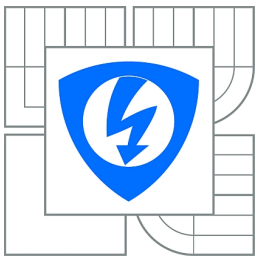
AUTOR PRÁCE — Bc. DALIBOR HAMPL

AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ MALINA

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Dalibor Hampl	***ID:***	98489
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Kryptografie na výpočetně omezených zařízeních

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s tzv. lehkou kryptografií. Jedná se o kryptografická primitiva a schémata, která jsou vhodné pro výpočetně omezená či slabá zařízení, jako jsou mikrokontroléry (8-bit/32-bit) či programovatelné čipové karty (smartkarta .NET V2, Javacard). Zhodnoťte a porovnejte používaná schémata a primitiva nejen z pohledu časové, paměťové náročnosti, ale také z výkonové náročnosti na vybraných zařízeních. Navrhněte a realizujte efektivní protokol vzájemné autentizace s ustanovením šifrovacího klíče na vybraném zařízení.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and Network Security. 4th edition. [s.l.] : [s.n.], 2006. 592 s. ISBN 0131873164.
[2] MENEZES, Alfred, VAN OORSCHOT, Paul, VANSTONE, Scott. Handbook of applied cryptography. Boca Raton : CRC Press, 1997. 780 s. ISBN 0849385237.
[3] COLE, Peter, RANASINGHE, Damith. Networked RFID systems and lightweight cryptography: raising barriers to product counterfeiting. 2008. 355 s. ISBN 3540716408

Termín zadání:	6.2.2012	***Termín odevzdání:***	24.5.2012

Vedoucí práce: Ing. Lukáš Malina
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Diplomová práce se zaměřuje na kryptografické prostředky zařízení s nižším výpočetním výkonem a na vzájemnou autentizaci uživatele a serveru pomocí smart karty.

V první části této diplomové práce je vysvětlena obecná kryptografie, kryptografická primitiva, kryptografické cíle, bezpečnostní modely a kryptografické prostředky na zařízeních s nižším výpočetním výkonem.

Druhá část práce je zaměřena na druhy zařízení s nižším výpočetním výkonem jako jsou RFID tagy, technologie NFC, mikrokontroléry a smart karty, do kterých patří .NET karty, java karty, MIFARE karty.

Praktická část práce se zabývá srovnáním vybraných zařízení s nižším výpočetním výkonem a změřením časové a výkonové náročnosti na smart kartě Gemalto .NET Smart Card V2+ pomocí speciálně navržené aplikace pro šifrování a dešifrování pomocí různých kryptografických algoritmů.

Dále rozebírá a objasňuje tři pokročilá autentizační schémata vhodná pro vzájemnou autentizaci uživatele a vzdáleného serveru s pomocí smart karty. Z postupů těchto autentizačních schémat vychází návrh nového autentizačního schématu, které se snaží eliminovat svým návrhem možné bezpečnostní útoky a zároveň zůstat efektivní. Autentizační schémata včetně navrhovaného schématu byla implementována na smart kartu Gemalto .NET Smart Card V2+. Pro všechna čtyři autentizační schémata je dále naprogramována aplikace pro otestování časové náročnosti daných schémat při autentizaci uživatele a serveru s pomocí smart karty.

KLÍČOVÁ SLOVA

Šifrování, kryptografická primitiva, hash, smart karta, .NET karta, zařízení s nižším výpočetním výkonem, autentizační schéma, vzájemná autentizace.

ABSTRACT

The thesis focuses on cryptographic algorithms of low performance devices, and mutual authentication of authentication server and user using smart cards.

In the first part of this thesis the cryptography, cryptographic primitives, cryptographic goals, security models and cryptographic algorithms of low performance devices are presented.

The second part focuses on low performance devices as RFID tags, NFC technology, microcontrollers and smart cards (.NET cards, java cards, MIFARE cards).

The practical part deals with the comparison of chosen low performance devices and measure the time required for encryption and decryption using different cryptographic algorithms on Gemalto .NET Smart Card V2+.

This thesis describes and explains the three authentication schemes for mutual authentication of remote server and user using smart cards. The new authentication scheme, which is based on the second related scheme, attempts to eliminate possible security attacks and keeps efficiency. For all four authentication schemes the application is implemented to test required time for authentication of server and user using smart cards.

KEYWORDS

Encryption, Cryptographic Primitives, Hash, Smart Card, .NET Card, Low Performance Devices, Authentication Scheme, Mutual Authentication.

BIBLIOGRAFICKÁ CITACE PRÁCE:

HAMPL, D. *Kryptografie na výpočetně omezených zařízeních.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 50 s, 1 příloha. Vedoucí diplomové práce Ing. Lukáš Malina.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Kryptografie na výpočetně omezených zařízeních 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 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.

V Brně dne: …… ……

podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Lukáši Malinovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne: …… ……

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

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

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Obsah

1 Úvod

V současné době si nedokáže většina lidí představit život bez takové samozřejmosti, jako jsou čipové karty a jiné technologie poskytující autentizaci pomocí předmětu. Mají široké spektrum použití v různých oblastech, například karty platební (bankovnictví), identifikační (kontrolované přístupy do chráněných prostor budov či informačních systémů), předplacené (televizní, telefonní, parkování, ve zdravotnictví), mobilních telefonů (SIM).

Kvůli ochraně osobních údajů a služeb samotných musí být veškerá komunikace a data chráněna. Z tohoto důvodu jsou implementovány kryptografické prostředky.

Kryptografie je vědní obor, který se zabývá převodem informací s aplikací matematických technik do podoby, která není pro ostatní čitelná bez znalosti určitého tajemství (soukromý klíč, dešifrovací tajný klíč atd.). Moderní kryptografické algoritmy jsou většinou postaveny na matematických problémech, které současná matematika nedokáže efektivně, nejlépe v reálném čase, řešit. Bezpečné šifrování musí být tak silné, aby se útočníkovi nevyplatilo toto šifrování prolomit. Časová náročnost prolomení šifry musí být adekvátní k důležitosti informace. Šifrování musí probíhat v přijatelném čase a množství výpočetních prostředků nutných k zašifrování a dešifrování by mělo odpovídat stupni bezpečnosti šifry a utajení přenášené informace. Velikost přenášených informací se nesmí šifrováním neúměrně zvyšovat a implementace algoritmu musí být co možná nejjednodušší.

Výpočetní náročnost kryptografických algoritmů podporovaných čipovými kartami závisí na velikosti paměti a mikroprocesoru čipové karty. Vzhledem k tomu, že výpočetní výkon mikroprocesorů v čipových kartách je nízký, používají se vybrané kryptografické algoritmy a schémata, které efektivně běží na těchto zařízeních, jedná se o tzv. lehkou kryptografii, se kterou se blíže seznámíme v 2. kapitole.

Ve 3. kapitole si objasníme, co jsou zařízení s nižším výpočetním výkonem, jaké druhy těchto zařízení máme v dnešní době k dispozici a zkráceně si popíšeme, jak pracují.

Cílem práce je bližší seznámení s lehkou kryptografií, jejím využitím na výpočetně omezených či slabých zařízeních (programovatelné čipové karty, RFID tagy, NFC a mikrokontroléry), což je ve 4. kapitole ukázáno na praktickém testu časové náročnosti kryptografických algoritmů na čipové kartě.

Dále je práce zaměřena v 5. kapitole na popis tří různých autentizačních schémat, založených na hash funkcích, operaci XOR a modulární aritmetice.

Dalším cílem této práce je návrh autentizačního schématu pro vzájemnou autentizaci, který bude bezpečný, ale i časově a výkonově efektivní a porovnaný s ostatními schématy po jejich implementaci v 7. kapitole.

2 Kryptografické prostředky na zařízeních s nižším výpočetním výkonem

V této kapitole si ilustrujeme, co jsou to kryptografické prostředky na zařízeních s nižším výpočetním výkonem a jaké se zde používají šifrovací metody, jaká zařízení je podporují a jaké jsou jejich výhody a nevýhody.

2.1 Co jsou kryptografické prostředky na zařízeních s nižším výpočetním výkonem

Kryptografické prostředky na výpočetně omezených zařízeních neboli lehká kryptografie je směr kryptografie, který se zaměřuje na vývoj rychlých a účinných kryptografických mechanismů pro výpočetně omezená (slabá) zařízení. Parametry této kryptografie musí počítat s omezeným množstvím energie, omezeným výpočetním výkonem, omezenou velikostí paměti, šířkou komunikačního pásma, velikostí a komplexností provedení těchto zařízení. Lehká kryptografie poskytuje právě takový stupeň bezpečnosti, který je co nejvíce možný, ale zároveň uskutečnitelný v takto omezených zařízeních. Proto se lehká kryptografie používá k vytváření bezpečných šifrovacích řešení pro levné RFID systémy (Radio Frequency Identification Systems), čipové karty nebo mikrokontroléry. [5]

Podle požadované funkce daného zařízení se aplikuje symetrické či asymetrické šifrování. Symetrické šifry slouží především pro kontrolu integrity dat, autentizaci subjektu a důvěryhodnost dat. Asymetrické šifry poskytují navíc nepopiratelnost dat a jsou výpočetně mnohem náročnější na hardware i software. [5]

2.1.1 Kryptografické cíle a bezpečnostní modely

Nejdůležitější bezpečnostní služby poskytované kryptografickými prostředky [5]:

- Důvěrnost - zajišťuje udržení přenášených nebo uložených dat v tajnosti. Pouze oprávněné entity mají umožněn přístup k přenášeným či uloženým datům.
- Integrita dat - zaručuje, že přenášená nebo uložená informace nebyla pozměněná po cestě k příjemci nebo v úložišti. Je tedy nutná detekce neoprávněné změny informace.
- Autentizace - jedná se o ověření identity, tj. ověření, že obdržené informace jsou odeslány od zdroje, se kterým si myslíme, že opravdu komunikujeme.
- Kontrola přístupu - schopnost řídit a omezit přístup do systému s informacemi.
- Nepopiratelnost - zajišťuje, že odesílatel informace nemůže popřít, že ji odeslal a příjemce nemůže popřít, že ji obdržel. Poskytuje tedy doklad o vyslání a obdržení informace.

Pro popis bezpečnosti kryptografických primitiv slouží bezpečnostní modely [5]:

- Dokonalá bezpečnost. Tento bezpečnostní model předpokládá neomezený výpočetní výkon protivníka. Proto, aby dané kryptografické primitivum spadalo do této kategorie, nesmí existovat algoritmus na prolomení tohoto primitiva bez ohledu na výpočetní výkon protivníka. Nejjednodušším příkladem takového zašifrování je zpráva upravená pomocí operace XOR s unikátním klíčem stejné délky jaké je původní zpráva. Protivník není schopen kryptogram bez znalosti šifrovacího klíče rozluštit. Jelikož musí být délka klíče stejná jako délka zprávy, je možné uvažovat o nasazení v zařízeních s nižším výpočetním výkonem, protože zde nejsou zprávy příliš dlouhé.
- Výpočetní bezpečnost. Tento model předpokládá polynomiální výkon protivníka. Kryptografické primitivum spadá do tohoto modelu, jestliže není znám algoritmus k prolomení tohoto primitiva v polynomiálním čase. Moderní kryptografická primitiva mají spadat právě do tohoto modelu.
- Praktická bezpečnost. Tento model také počítá s výpočetním výkonem protivníka. Na rozdíl od modelu výpočetní bezpečnosti zde neexistují žádné relativní hranice. Pro primitiva spadající do tohoto modelu nesmí existovat algoritmus pro zlomení tohoto primitiva, který vyžaduje méně než N operací, jejichž počet je dostatečně vysoký. Právě moderní kryptografická primitiva nabízejí praktickou bezpečnost.
- Prokazatelná bezpečnost znamená, že složitost prolomení daného kryptografického primitiva je ekvivalentní vyřešení těžkého matematického problému, jako problému faktorizace či problému diskrétního logaritmu. Do této kategorie spadají kryptografická primitiva založená na asymetrické kryptografii veřejného klíče.

2.1.2 Obecná kryptografie a kryptografická primitiva

Kryptografická primitiva můžeme dělit do tří velkých skupin: kryptografická primitiva bez klíče, symetrická kryptografická primitiva, asymetrická kryptografická primitiva. [1, 2, 5]

Kryptografická primitiva bez klíče obsahují kryptografické nástroje, které nejsou vůbec založeny na šifrovacích klíčích. Patří sem například hash funkce, jednosměrné funkce a generátory náhodných čísel. Samy o sobě nezabezpečí například danou zprávu, ale jsou součástí bezpečnostního systému či kryptografického protokolu. [1, 2, 5]

Hash funkce je výpočetně efektivní matematická funkce, která převede binární řetězec libovolné délky na binární řetězec určité pevné délky. Pro hash funkci musí být splněny tyto požadavky: musí se jednat o jednosměrnou matematickou funkci (ze

zprávy z je možné lehce vypočítat hash této zprávy $h(z)$, ale výpočet zprávy z z hash hodnoty $h(z)$ je nemožný), musí být odolné vůči kolizi (toto znamená, že nesmí být 2 zprávy z_1 a z_2 takové, že jejich hash hodnoty se rovnají, tedy $h(z_1) = h(z_2)$. Nicméně při tomto neomezeném počtu vstupů a omezeném počtu výstupů je nevyhnutelné, že dojde ke kolizi. Proto se v praxi použijí takové funkce, u kterých je výpočetně náročné najít kolizi) a za třetí hash funkce by měla být náhodné zobrazení (prakticky to znamená, že by mělo být výpočetně nemožné odlišit hash funkci od náhodného zobrazení). Mezi nejznámější algoritmy pro výpočet hash hodnoty patří MD5, SHA (SHA-0, SHA-1, SHA-256, SHA-512).

Jednosměrná funkce je matematické zobrazení $f: Y \to Z$, které lze snadno spočítat, ale jeho inverze je velmi těžká. Jinak řečeno, existuje algoritmus pro výpočet zobrazení v polynomiálním čase. Existence algoritmu pro výpočet inverze v polynomiálním čase je velmi nepravděpodobná. Jednosměrné funkce jsou velmi důležitá primitiva v kryptografii a mnohá další kryptografická primitiva jsou na nich založena. Například právě hash funkce je založena na jednosměrné funkci.

Generátory náhodných čísel se vyskytují v mnoha kryptografických primitivech používajících šifrovací klíče a kryptografických protokolech jako náhodné posloupnosti. Náhodnost je založena na rovnoměrném rozložení a nezávislosti. Nezávislost se týká posloupnosti náhodných čísel. Generátory s náhodnou posloupností generují náhodnou posloupnost z náhodných fyzikálních dějů. Generátory s pseudonáhodnou posloupností generují z daného výchozího stavu posloupnost podle stanovených pravidel. Tato posloupnost se jeví jako náhodná. Existují hardwarové i softwarové varianty.

Symetrická kryptografická primitiva používají pro šifrování a dešifrování stejný klíč, nebo dva klíče (dešifrovací klíč je snadno odvoditelný ze šifrovacího klíče). Zpráva je zašifrována šifrovacím algoritmem, který používá vygenerovaný klíče. Zašifrovaná zpráva se odešle příjemci. U příjemce dojde k jejímu dešifrování pomocí stejného klíče, jakým byla zpráva zašifrována. Pro dobrou bezpečnost těchto kryptografických primitiv je velmi důležitá délka klíče. Dnes je považována za dostatečnou délka 128 nebo až 256 bitů (2^{128} nebo 2^{256} možných klíčů). Symetrická kryptografická primitiva používají pro šifrování takové matematické algoritmy, že i když je známá vstupní i zašifrovaná zpráva je velmi těžké zjistit šifrovací klíč. Symetrická kryptografická primitiva jsou rychlá s malou výpočetní náročností. Problém nastává při uchování klíčů v tajnosti při jejich vytvoření a přenosu. Dále se vzrůstajícím počtem komunikujících stran kvadraticky vzrůstá i počet klíčů. Lze použít pouze jeden klíč, což má však za následek podstatné zvýšení bezpečnostního rizika. Dělí se na proudové a blokové šifry. [1, 2, 5]

Proudové šifry – šifrování u nich probíhá po jednotlivých bitech. Jejich hlavní výhodu je, že nešíří chyby nebo je šíří jen omezeně. Jsou schopny se vypořádat

s převrácenými, chybějícími nebo vloženými bity v závislosti na jejich konkrétní implementaci. Mezi nejznámější proudové šifry patří RC4 a FISH.

Blokové šifry – při šifrování dojde k rozdělení zprávy na bloky s pevně daným počtem bitů. Hodnota zašifrovaného bitu závisí i na dalších bitech zprávy. Výhodou blokových šifer je jejich bezpečnost, která je vyšší než u proudových. Vůči nim jsou mnohem pomalejší. Mezi známé blokové šifry patří DES, 3DES, AES atd. DES vychází z krátkých 64 bitových bloků a 56 bitových klíčů, což už je dnes z hlediska bezpečnosti nedostatečné. 3DES vychází z DES. U zprávy určené k zašifrování dochází k trojnásobné aplikaci DES s třemi různými klíči (celkem je délka klíče 168 bitů, ale stále zůstává zachována velikost bloku 64 bitů). 3DES dnes nahrazuje AES, který má 128 bitovou velikost bloků a velikost klíče se může volit 128, 192 nebo 256 bitů. Šifrování probíhá ve čtyřech krocích. Prvním krokem šifrování AES je substituce, ve které je každý byte v matici nahrazen jiným podle předem daného klíče. Ve druhém kroku první řádek matice zůstává stejný a v každém dalším řádku se cyklicky přesouvají byty doleva. Ve třetím kroku probíhá násobení pevným polynomem. Ve čtvrtém kroku probíhá pomocí operátoru XOR kombinace bytů. Tím se odvodí částečný klíč. Kombinací bytu částečného klíče s bytem původní zprávy dochází k šifrování zprávy.

Asymetrická kryptografická primitiva užívají pro šifrování a dešifrování soukromý a veřejný klíč. Tyto klíče jsou odlišné a s výpočetním výkonem používaným dnes prakticky neodvoditelné. Asymetrická kryptografická primitiva jsou založená na jednosměrných funkcích, které jsou výpočetně snadné, ale jejich inverze je velmi náročná. Základ tvoří současně vygenerovaný soukromý a veřejný klíč. Jedním klíčem se zpráva zašifruje a druhým se dešifruje. Kvůli použitým specifickým matematickým funkcím a specifickým druhům použitých čísel musí být délka klíčů mnohem větší než u symetrických kryptografických primitiv. Pro n účastníků je nutné mít pouze $2n$ klíčů. Např. Odesílatel zašifruje zprávu veřejným klíčem příjemce. Ten ji dešifruje pomocí svého soukromého klíče, který má jen on, tím je zaručena důvěrnost informace. Výhodou je přenášení pouze veřejných klíčů (nedojde k vyzrazení soukromého klíče). Nevýhodou je velká výkonová náročnost pro dešifrování kvůli velikosti klíčů. Tímto jsou mnohem pomalejší než symetrická kryptografická primitiva. Patří sem DH, ElGamal, RSA, ECC, DSA, DSS. [1, 2, 5]

RSA algoritmus využívá problému faktorizace velkých prvočísel. Vytvoříme součin y dvou velkých prvočísel r, s. Vezmeme číslo t, pro které platí, že $t \leq (r-1)(s-1)$ a je nesoudělné s tímto součinem. Nalezneme číslo u, pro které platí, že $tu = 1[\mathrm{mod}(r-1)(s-1)]$. $(y,\ t)$ je veřejný klíč a (y, u) soukromý klíč. Pro šifrovanou zprávu c platí, že $c = m^t (\mathrm{mod}\ y)$, kde m je zpráva určená k šifrování. Pro dešifrování platí, že $c^u (\mathrm{mod}\ y) = m$. Modulo n a soukromí klíč v RSA mají délku 1024 nebo 2048 bitů.

ECC neboli kryptografie eliptických křivek nabízí techniky, které jsou mnohem účinnější než tradiční algoritmy. U ECC jsou výpočty založeny na bodech eliptických křivek. Hlavní je získat bod P, který splňuje rovnici $P = k.Q$ pro daný bod Q a dané číslo k. Jedná se o jednosměrnou funkci. Bezpečnost ECC je založená na problému diskrétního logaritmu eliptických křivek (ECDLP). Hledá se tedy k z $P = kQ$ pro dané P a Q. Řešení ECDLP je považováno za výpočetně nemožné, pokud je k dostatečně velké. S ohledem na dnešní výpočetní výkon se považuje za dostatečnou velikost klíče vyšší než 224 bitů.

Digitální podpis – pro digitální podpis může být použit standard šifrovací algoritmus DSS (maximální délka klíče 1024 bitů), DSA (délka klíče až 4096 bitů) nebo i RSA. Digitální podpis je nutný k zajištění autorizace šifrované zprávy. Odesílatel zprávu zašifruje svým soukromým klíčem. Kdokoliv může tuto šifrovanou zprávu dešifrovat použitím veřejného klíče od odesílatele. Kvůli možnosti zneužití veřejného klíče odesílatele je nutné posílat s podepsanou zprávou také certifikát od certifikační autority identifikující odesílatele. Certifikát obsahuje soukromým klíčem certifikační autority podepsaný hash údajů uživatele a jeho veřejný klíč. Digitální podpis se kvůli rychlosti neprovádí pro celou zprávu, ale pouze pro hash této zprávy, který zaručí její autentičnost.

3 Zařízení s nižším výpočetním výkonem

Do těchto zařízení patří zařízení s malým výpočetním výkonem a malou paměťovou kapacitou. V této kapitole si přiblížíme a objasníme technologie vztažené k čipovým kartám (smart card), jejich programovací platformy (Java Card, .NET Card), MIFARE karty, RFID tagy, NFC a mikrokontroléry.

3.1 Programovatelné čipové karty

Smart Card neboli „chytré“ smart karty jsou čipové karty, které obsahují integrovaný obvod uložený nejčastěji v plastovém pouzdru. Integrovaný obvod se skládá z mikroprocesoru se vstupním a výstupním rozhraním, pamětí EEPROM, RAM a ROM. EEPROM má v sobě uložené aplikační programy. Paměť RAM se používá pro ukládání dočasných dat a ROM obsahuje základní software od výrobce. V některých čipových smart kartách je také implementován kryptografický koprocesor, který podporuje některý ze šifrovacích algoritmů a tím provede kryptografické výpočty rychleji než standardní mikroprocesor karty. Výhodou čipových smart karet je, že data nejenom uloží, ale také podle požadovaného záměru zpracují. [3, 6]

Smart karty mají širokou oblast využití. Například každý člověk, který vlastní mobilní telefon používá SIM kartu. Další využití je v oblasti financí, která vyžaduje informační bezpečnost pro bankovní operace – různé platby, kreditní debetní karty atd. Také se smart karty uplatňují při zajišťování ochrany identifikace uživatelů počítačových sítí, při evidenci docházky a při vstupních kontrolách do budov či jiných chráněných prostor atd.

Podle způsobu komunikace s kartou se dělí na 3 typy: kontaktní, bezkontaktní a s dvojitým rozhraním.

- Kontaktní smart karta obsahuje všechny výše popsané části smart karty a kontaktní oblast čtvercového nebo oválného tvaru s 6 nebo 8 kontakty.
- Bezkontaktní smart karta využívá technologii podobnou RFID. Obsahuje rovněž všechny výše popsané části, dále vestavěnou anténu pro bezkontaktní komunikaci, kterou zprostředkovávají elektromagnetické vlny. Napájecí energie se získává přes anténu nebo ze zabudované baterie. Karta funguje ve vzdálenosti od 10 cm do 1 m, má vysokou přenosovou schopnost a její používání je pohodlnější než u kontaktní smart karty.
- Smart karty s dvojitým rozhraním jsou multifunkční, tj. podle potřeby se využívají buď jako kontaktní nebo bezkontaktní.

Ze softwarového pohledu se na smart kartě nachází: operační systém, aplikace a v kartách obsahujících interpret jsou to běhové prostředí a zavaděč.

Operační systém smart karet je systém založený na příkazech, na které je karta schopna reagovat. Tento systém se stará o řízení komunikace a paměťových operací mezi aplikací a čipem. Dále spravuje souborový systém a poskytuje rozhraní pro kryptografický hardware, pokud je jím smart karta vybavena.

Aplikace jsou programovány jazykem zvoleným podle operačního systému použitého na dané smart kartě.

Běhové prostředí je komponenta, která obsahuje interpret. Stará se o běh aplikací na čipové kartě a také o knihovny těchto aplikací.

Zavaděč obstarává nahrávání a odstraňování aplikací ze smart karty.

3.1.1 Java Card

Technologii Java Card vyvinula společnost Oracle Corporation. Dává možnost spouštět aplikace v jazyce Java na smart kartách a jiných obdobných zařízeních, které mají vlastní paměť. Java programuje pro určité zařízení určitou aplikaci. Je tedy univerzální, protože není závislá na zařízení, na kterém běží. Proto vznik, zkoušení a uvádění aplikace do praxe je velmi rychlé a není problém s využitím v souborných projektech řešených v Javě. Více aplikací v ní může běžet zároveň, jedná se tedy o multiplikační prostředí. Další výhodou je velká bezpečnost dat. Zajišťují ji následující bezpečnostní prvky: před instalační analýza a verifikace kódu, applet firewall (bezpečné oddělení každé aplikace), zajištění integrity kódu mezi verifikací a instalací tohoto kódu na kartu, užití autentizace a kryptografických algoritmů, transakce, správa kryptografických klíčů a užívání pin kódu [16, 21].

Základem technologie Java Card jsou tři komponenty Java Card Virtual Machine, Java Card Runtime Enviroment a Java Card Application Programming Interface.

- Java Card Virtual Machine (JCVM) specifikuje část jazyka Java a definuje takový virtuální stroj, aby mohl být použit na smart kartě.
- Java Card Runtime Enviroment (JCRE) popisuje chování běhového prostředí při kompilaci a spuštění programu v jazyce Java, včetně správy paměti, appletů, bezpečnosti, atd.
- Java Card Application Programming Interface (JCAPI) obsahuje popis základních tříd či balíčků, určených pro programování aplikací v Java Card.

V technologii Java Card jsou aplikace odděleny od vlastního systému smart karty. Technologie Java Card dodává pro aplikace základní třídy a standardní systém pro aplikace. Programování aplikací s použitím JCAPI je možné použít v různých architekturách smart karet.

Vznik aplikace pomocí technologie JavaCard lze popsat takto: kompilátor přeloží zdrojový kód v jazyce Javě, dále proběhne konverze do CAP souboru (converted applet). Jedná se o Java archiv, který obsahuje spustitelný binární kód všech tříd, které

tvoří balík (package). Dále proběhne kontrola správnosti CAP souboru, kterou provádí Off-Card Verifier (OCV). Jedná se o podstatný bezpečnostní prvek, neboť technologie JavaCard jako taková neprovádí kontrolu kódu na smart kartě. Poté dojde k nahrání ověřeného kódu na smart kartu, applet je na smart kartu nainstalován.

Virtuální stroj se dělí kvůli omezeným výpočetním prostředkům smart karty (není možné provádět všechny operace na smart kartě) na interpret a konvertor. Interpret je obsažen na smart kartě a konvertor na PC nebo terminálu.

Technologie Java Card (také i virtuální stroj) obsahuje jen určitou část jazyka Java. Nejsou zde podporovány například velké datové typy, řetězce, dynamické zavádění tříd, vícerozměrná pole a vlákna, atd.

3.1.2 .NET Card

Technologie .NET Card je softwarová platforma vyvinutá firmou HiveMinded, která ji implementovala do smart karet jako .NET Card framework, který vznikl na popud firmy Microsoft. Jde o multiplikační prostředí, tj. více aplikací na smart kartě spolupracuje mezi sebou. Aplikace pro technologii .NET Card lze programovat pomocí různých programovacích jazyků, například C++, C#, Visual Basic atd. Je také možné použít v jednom kódu jedné aplikace více programovacích jazyků zároveň, protože je daný kód aplikace následně překompilován do .NET kódu. Stejně jako technologie Java Card obsahuje .NET Card virtuální stroj využívající konceptu aplikační domény k izolování právě běžících aplikací a zabraňujícímu neoprávněnému sdílení dat mezi aplikacemi. [6]

.NET Card framework je velmi podobný Microsoft .NET frameworku. Hlavními rozdíly jsou: .NET Card framework má běhové prostředí CLR (Common Language Runtime) upravené přímo pro řízení běhu aplikací na smart kartě. CLR nepodporuje implicitní typy. Obsahuje speciální binární formát pro aplikace na smart kartě, který je úspornější a tím se lépe hodí pro smart karty. Nejsou povolena vícerozměrná pole a datové typy pracující s plovoucí čárkou. Neumožňuje práci s vlákny. Nejsou podporována asynchronní volání. .NET Card framework podporuje speciální práci s paměťovým modulem na smart kartách. Aplikace je uložena ve stálé paměti a je aktivována, až když je volána externí aplikací. [10]

Obsah .NET karty je následující: .NET Card framework, pod kterým si můžeme představit jeho běhové prostředí CLR a knihovny. Na .NET kartě se dále nachází servery, souborový systém v EEPROM paměti a konfigurační soubory. Servery jsou velmi důležité v komunikaci s kartou. Každá aplikace nahraná na kartu představuje server. Po nahrání aplikace na kartu se registruje server a registrují se jeho služby. Právě skrze tyto registrované služby se komunikuje se serverovou aplikací. Pod pojmem služba si můžeme představit třídu a její metody takové, jaké jsou u objektových programovacích jazyků. Každá služba, která běží, je popsána unikátním URI (Uniform

Resource Identifier). Klientská aplikace, která je spuštěná na PC, může prostřednictvím URI přistupovat k těmto metodám a volat je.

Běhové prostředí CLR zajišťuje běh aplikací na smart kartě v podobě bytekódu, který se nazývá CIL (Common Intermediate Language). .NET karta obsahuje interpret jazyka CIL. Prostředí CLR má naimplementován CLI (Common Language Infrastructure). CLI dovoluje spouštět aplikace naprogramované v některém z vysokoúrovňových jazyků v prostředí smart karty, aniž by tyto aplikace musely být přepsány a upraveny přímo pro dané prostředí.

3.1.3 MIFARE karta

MIFARE karta patří také mezi bezkontaktní čipové karty. Byla vyvinuta společností NXP Semiconductors a patří mezi nejvyužívanější čipové karty na světě. MIFARE karta vychází z technologie RFID, která je blíže vysvětlená v části 3.2. Výhodou oproti RFID je, že má mnohem větší paměť a vyšší úroveň zabezpečení, neboť se všechna komunikace mezi čtečkou a kartou šifruje. Pro operaci s daty je třeba prokázat znalost některého z klíčů uložených na kartě. Pro přenos dat používá karta frekvenci 13,56 MHz. [17, 18, 19]

Existuje více druhů karet MIFARE (Classic, Ultralight, Ultralight C, DESFire EV1 a Plus).

MIFARE Classic – používá proprietární protokol kompatibilní se standardy smart karet [19], s vlastním bezpečnostním protokolem NXP pro autentizaci a šifrování. Využívá metody šifrování Crypto1. Je vhodná pro uchovávání dat. Velikost paměti je 1 kB nebo 4 kB. Patří mezi levnější karty, a proto se hodně používala. Již došlo k jejímu prolomení.

MIFARE Ultralight – má podobné využití jako MIFARE Classic, je s ní kompatibilní. Paměť má pouze 512 b a bez jakékoliv ochrany. Lze použít na vzdálenost až 10 cm bez nutnosti baterie. Rychlost přenosu dat je 106 kb/s.

MIFARE Ultralight C – vychází z MIFARE Ultralight. Obsahuje 192B EEPROM paměti, přístup k datům je chráněn 3DES algoritmem.

MIFARE DESFire EV1 – obsahuje mnohem vyšší hardwarovou i softwarovou bezpečnost než MIFARE Classic. Obsahuje již i 3DES/AES kryptografický koprocesor, 2 kB, 4 kB nebo 8 kB paměť. Maximální přenosová vzdálenost mezi kartou a čtečkou činí 10 cm.

MIFARE Plus – je nástupce karty MIFARE Classic. Používá pro vzájemnou autentizaci karty, snímače a dat metodu šifrování Crypto1, a tím zajišťuje bezproblémový přechod z technologie MIFARE Classic. Jako novou používá pro autentizaci standardní metodu šifrování AES s klíčem 128 b. Komunikační rychlost MIFARE Plus mezi snímačem a kartou je až 848 kbit/s. Velikost paměti je opět 2 kB nebo 4 kB. [17, 18, 19]

3.2 RFID tagy

RFID tagy mají široké využití v různých oblastech našeho života. Všude, kde se eviduje docházka osob, nebo se otvírají dveře vstupními kartami, se využívají RFID tagy. Obdobně se uplatňují parkovací karty nebo karty v předplatních a stravovacích systémech, kde se navíc eviduje stav účtu pomocí informačního systému poskytovatele. Ve zdravotnictví slouží k identifikaci osob a krevních konzerv na základě RFID štítku. Výhodou je minimalizace chyb způsobených lidským faktorem. V obchodech se využívají nejjednodušší RFID tagy o velikosti 1 bit. Slouží jako ochrana před nedovoleným odcizením zboží.

Nosič informací v RFID systému nazýváme tag nebo jinak transpondér. Primární funkcí RFID tagu je uložení dat do vnitřní paměti a jejich poskytnutí RFID systému.

Forma, tvar, materiál a rozměry tagů se mohou velmi lišit. Tag obsahuje mikročip a anténu. Čím větší je použitá frekvence, tím menší může být anténa. [5]

Každá implementace RFID technologie obsahuje tagy pro označení objektů, čtecí zařízení a řídicí systém, který zajišťuje hromadné zpracování všech načtených tagů v dosahu čtecích zařízení a přenesení zpracovaných dat do návazného informačního či řídicího systému.

RFID tagy dělíme podle možnosti zápisu a podle napájení. [5]

Podle možnosti zápisu se tagy dělí na:

- Tag Read only – RO – je určen pouze pro čtení, neboť má pouze sériové číslo, které je zakódované při jeho výrobě a nelze je měnit. Jde o obdobu čárového kódu. Paměť je do 512 b. Při načítání více tagů je rychlost čtení 1000 tag/s.
- Tag Write Once Read Many – WORM – je určen také jen pro čtení jako tag Read only, ovšem není naprogramován při výrobě, ale až prodejcem nebo dodavatelem. Takto zapsaná data se již nedají přepsat. Paměť je do 512 bit. Rychlost čtení při načítání více tagů je 200 tag/s – vhodné pro etiketu na zboží.
- Tag Read/Write – RW – jak již název napovídá, lze tento tag mnohokrát přepsat. Paměť je mnohem větší než u tagů výše, záleží na pasivitě či aktivitě tagu (rozdíl vysvětlen níže). U pasivního tagu je paměť do 8 Kb a u aktivního do 2 Mb. Paměť je adresovatelná a snadno měnitelná. Rychlost čtení více tagů je 1000 tag/s.

Podle napájení se tagy dělí na:

- Aktivní tag – má vlastní miniaturní baterii. Jeho nevýhodou je složitost a s tím související vyšší cena a větší váha. Díky použití baterie se také snižuje jeho životní cyklus. Tento zdroj energie se používá pro napájení

integrovaného čipu a k zesílení signálu pro komunikaci s RFID čtečkou. Přenosová vzdálenost je kolem 100 metrů.

- Pasivní tag – nemá vlastní baterii a napájí se přímo z pole RFID čtečky. Tento tag má díky těmto vlastnostem velmi dlouhou životnost a mnohem nižší cenu. Přenosová vzdálenost je však mnohem nižší (od 10 cm do několika metrů). Záleží na použité frekvenci a velikosti antény.
- Semiaktivní tag – mají baterii, která ovšem slouží pouze k zvýšení dosahu čtení. Výhodou je vyšší výdrž baterie než u aktivního tagu.

3.3 Technologie NFC

NFC (Near Field Communication) je nezabezpečená bezdrátová komunikační technologie s přenosem dat mezi elektronickými zařízeními na velmi krátkou vzdálenost. Tato technologie je rozšířením standardu smart karet a RFID ISO/IEC 14443. NFC technologie právě kombinuje rozhraní smart karet a bezdrátové komunikační zařízení RFID. Podle standardu o NFC ISO/EIC 13157 je přenosová vzdálenost této technologie do 0,2 m, přenosová frekvence 13,56 MHz, half-duplexní přenos a jeho rychlost až 424 kbit/s. NFC technologie je také kompatibilní s pasivními RFID tagy. [24]

Pokud má NFC zařízení svůj zdroj energie, označuje se jako aktivní. Pokud nemá svůj zdroj napájení, jedná se o pasivní NFC zařízení a energii získává stejně jako v RFID systému, viz část 3.2. Komunikovat spolu mohou dvě aktivní zařízení (peer to peer) nebo aktivní s pasivním.

Mezi pasivní NFC zařízení patří NFC tag, tedy čip s anténou (s malým výpočetním výkonem), v němž jsou uloženy informace, které jsou načteny v případě přiblížení nějakého NFC aktivního zařízení. [23]

Technologie NFC má široké použití. Nejvíce ve sféře mobilních telefonů. Uživatel může využít následujících služeb: provádět platby pomocí mobilního telefonu (nebo jiného zařízení) s podporou NFC a příslušného platebního terminálu, provést výměnu přihlašovacích údajů k navázání spojení přes Wi-Fi či bluetooth pro přenos souborů mezi aktivními zařízeními stejné skupiny (například 2 mobilní telefony, notebook atd.), parkovat na placených parkovištích, kdy platba bude provedena při odjezdu mobilním telefonem u terminálu, požádat o informace na informačních místech k tomuto účelu vhodně vybavených, zakoupit lístek po stáhnutí aktivního plakátu do mobilu, rezervovat si hotel nebo let, vyměňovat informace dotykem s jiným zařízením atd. V takových případech mohou přístroje s NFC nahradit velké množství RFID tagů, smart karet, papírových lístků, atd. Hlavní využití spočívá v zastoupení funkcí karet i čtecích zařízení a to pouhým přiblížením dvou zařízení s technologií NFC.

Z bezpečnostního hlediska, jak již bylo výše zmíněno, není technologie NFC zabezpečená. Spoléhá se na kryptografické zabezpečení aplikace, která technologii NFC

využívá pro své účely, což je možnost ochrany této technologie proti útokům. Tato technologie je zranitelná pro útok odposlechnutím, změnou přenášených dat nebo jejich poškozením. Proveditelnost těchto útoků je však velmi ztížena krátkou přenosovou vzdáleností této technologie.

3.4 Mikrokontroléry

Mikrokontroléry se používají pro různé kontrolní a řídící úlohy v různých oborech. Jsou určeny pro embedded (vestavné) zařízení, mikrokontrolér je řídící jednotkou nějakého elektronického přístroje nebo je součástí nějakého zařízení, kde splňuje specifickou funkci. Jedná se o samostatnou jednotku schopnou samostatné interakce a komunikace s okolím. Kvůli jejich univerzálnosti, nízké spotřebě a ceně jsou hodně používány v různých současných elektronických zařízeních. Zařízení s mikrokontroléry, kde je nutné použít kryptografické prostředky, se nacházejí například v čidlech, senzorech, bezdrátových senzorových sítích, u průmyslových kontrolních a řídících prvků, dále v různých systémech ochran automobilů proti krádeži atd. [7, 8, 9]

Mikrokontroléry neboli jednočipové mikropočítače mají všechny podstatné části mikropočítače implementované v jediném pouzdře, integrovaném obvodu. Tento čip tedy obsahuje:

- Řadič a aritmetickou jednotku. Podle typu se používá délka slova 4, 8, 16, 32 bitů.
- Paměť programu neboli kódovou paměť. Jedná se o typ EPROM, Flash nebo u mikrokontrolérů vyrobených s pevně daným programem, využitých ke konkrétní aplikaci, pak typu ROM.
- Paměť dat typu RAM, někdy také doplněnou o paměť EEPROM.
- Periferní obvody pro vstup a výstup dat.

Dále jsou většinou do mikrokontrolérů integrovány: generátor hodinového signálu, obvody pro kontrolu činnosti mikrokontroléru, obvody pro programování kódové paměti přímo v aplikaci, A/D, D/A převodníky, řadiče přerušení, DMA řadiče atd.

Mikrokontroléry se vyrábí v různých velikostech a s různými výkony.

U různých typů mikrokontrolérů je velikost pamětí pro paměť programu od 1 kB do 256 kB a pro paměť dat od 32 bytů do 16 kB.

Mikrokontroléry dělíme podle organizace paměťových adresních prostorů na tyto dvě skupiny:

- Mikrokontroléry s lineárním uspořádáním adresního prostoru. Takovéto organizování paměti se označuje jako Von Neumannova architektura. U mikrokontrolérů se je označována jako SFR (Special Function Registers). U mikrokontrolérů s touto architekturou je tedy použit jediný adresní prostor s namapovanou pamětí programu, pamětí dat a registry pro řízení IO

obvodů. Tuto architekturu používají mikrokontroléry Motorola, Intel 80196, Hitachi a další výrobci.

- Mikrokontroléry s odděleným adresním prostorem pro paměť programu a paměť dat. Tato architektura se jmenuje Harwardská koncepce. Používá fyzicky oddělenou paměť programu a dat, oddělené sběrnice pro přístup k instrukcím a datům. Díky tomuto nemusí mít programová paměť a datová paměť stejně dlouhé datové slovo. Používají ji mikrokontroléry AVR firmy Atmel, mikrokontroléry 8051, 8052 firmy Intel, mikrokontroléry PIC firmy Microchip, atd.

Periferní obvody u mikrokontrolerů jsou různé podle toho, pro jakou konkrétní třídu aplikací jsou určeny. Firma vyrábí jeden typ mikrokontroléru ve více variantách lišících se právě různými periferními obvody.

Mikrokontroléry používají standardně tyto periferní obvody:

- Paralelní IO porty. Většinou mikrokontrolér obsahuje jeden nebo více paralelních portů použitých pro vstup a výstup binárních dat.
- Sériové rozhraní. Toto rozhraní používají mikrokontroléry ke komunikaci s dalšími mikrokontroléry nebo s ovládacími panely. Mikrokontroléry mají jednu nebo až pět sériových linek. Tyto linky podporují standardní asynchronní přenos dat a protokoly, například SPI, I2C, CAN atd. Sériové rozhraní podporující tyto protokoly se používá pro připojení vnějších pamětí, obvodů RTC (Real Time Clock), A/D převodníků atd.
- Čítače, časovače. Jedná se o standardní části mikrokontrolérů. V základu mají jeden až dva 8 nebo 16 bitové čítače, časovače. U speciálních mikrokontrolerů může být rozšířen počet čítačů na 6 až 8. Tyto čítače mohou podporovat speciální funkce, například generování obdélníkových průběhů dané střídy a frekvence atd.

Výhodou mikrokontroléru je, že umožňuje nahradit hodně logických obvodů a součástek, které byly nutné při výrobě podobných zapojení v dřívějších dobách a pomocí různých periferií integrovaných v mikrokontroléru umožňuje nahradit i další integrované obvody. Toto vše pomáhá snížit náklady na výrobu zařízení a dovoluje zdokonalovat a dodávat do zařízení další funkce pouze změnou programu, aniž by se musela měnit konstrukce mikrokontroléru. [7, 8, 9]

Každý typ mikrokontroléru má svou instrukční sadu. Instrukční sada obsahuje seznam strojových instrukcí, podle kterých se řídí činnost mikrokontroléru. Tyto strojové instrukce se zapisují v jazyce symbolických adres neboli v assembleru. V tomto programovacím jazyce každý příkaz vyjadřuje právě jednu strojovou instrukci. Protože jsou příkazy assembleru různé podle typu mikrokontroléru, nelze přenášet programy

napsané v tomto jazyce pro konkrétní typ mikrokontroléru na jiný typ mikrokontroléru. Proto se k vytváření zdrojového kódu pro mikrokontroléry používá univerzálnější programovací jazyk C (používá se i C++). V oblasti programování mikrokontrolérů je tento jazyk nejvíce využíván. Jedná se o univerzální jazyk nízké úrovně, který není zaměřený na určitou oblast využívání. Je čitelnější než assembler a lépe přenositelný mezi různými architekturami. Efektivita kódu v jazyce C je vzhledem k jeho jednoduchosti velká. Například mikrokontroléry z rodiny AVR firmy Atmel mají vytvořenu instrukční sadu tak, aby efektivně využila kód vytvořený v jazyce C. Pro vytvoření programu je u nich možné použít vývojové prostředí AVR Studio obsahující také překladač kódu a simulátor. [22]

Podle typu operačního systému mohou mikrokontroléry obsahovat naimplementované kryptografické algoritmy AES, DES/TDES, RSA, ECC a algoritmy pro výpočet hash hodnoty MD5 nebo SHA-1.

4 Podporované kryptografické prostředky na zařízení s nižším výpočetním výkonem

V předchozí kapitole jsme se seznámili se zařízeními s nižším výpočetním výkonem v obecném měřítku. Nyní si představíme parametry konkrétních typů smart karet v podobě Gemalto .NET Smart Card V2+, SmartCafe Expert 144K a mikrokontrolérů ATmega128, Stellaris LM3S8962, viz tabulka 1. [7, 8, 10, 11]

Tabulka 1: parametry konkrétních zařízení s nižším výpočetním výkonem

	Gemalto .NET V2+	SmartCafe Expert 144K	ATmega128	Stellaris LM3S8962
Typ zařízení	.NET smart card	Java card	mikrokontrolér	mikrokontrolér
Typ OS	.NET	Sm@rtCafé	RTOS, RTKER, Femto atd	RTOS, RTXC atd
Symetrické algoritmy	DES (64 b), 3DES (192 b), AES (128 b,256 b)	DES (64 b), 3DES (192 b), AES (128 b,256 b)	Podle typu OS	Podle typu OS
Asymetrické algoritmy	RSA (256 b až 2048 b)	RSA (do 2048 b)	Podle typu OS	Podle typu OS
Hash	SHA1, SHA256, MD5	SHA-256	Podle typu OS	Podle typu OS
CPU	RISC 32 bit	RISC 32 bit	AVR RISC 8 bit	ARM Cortex-M3 RISC 32 bit
Frekvence	66 MHz	33 MHz	16 MHz	50 MHz
RAM	16 kB	12 kB	4 kB	64 kB
ROM	80 kB	72 kB	-	-
EEPROM, Flash	400 kB, 0 kB	144 kB, 0 kB	4 kB, 128 kB	0 kB,256 kB
Kryptografický koprocesor	Ano	Ano	Ne	Ne
Generátor náhodných čísel	Ano	Ano	Ne	Ne

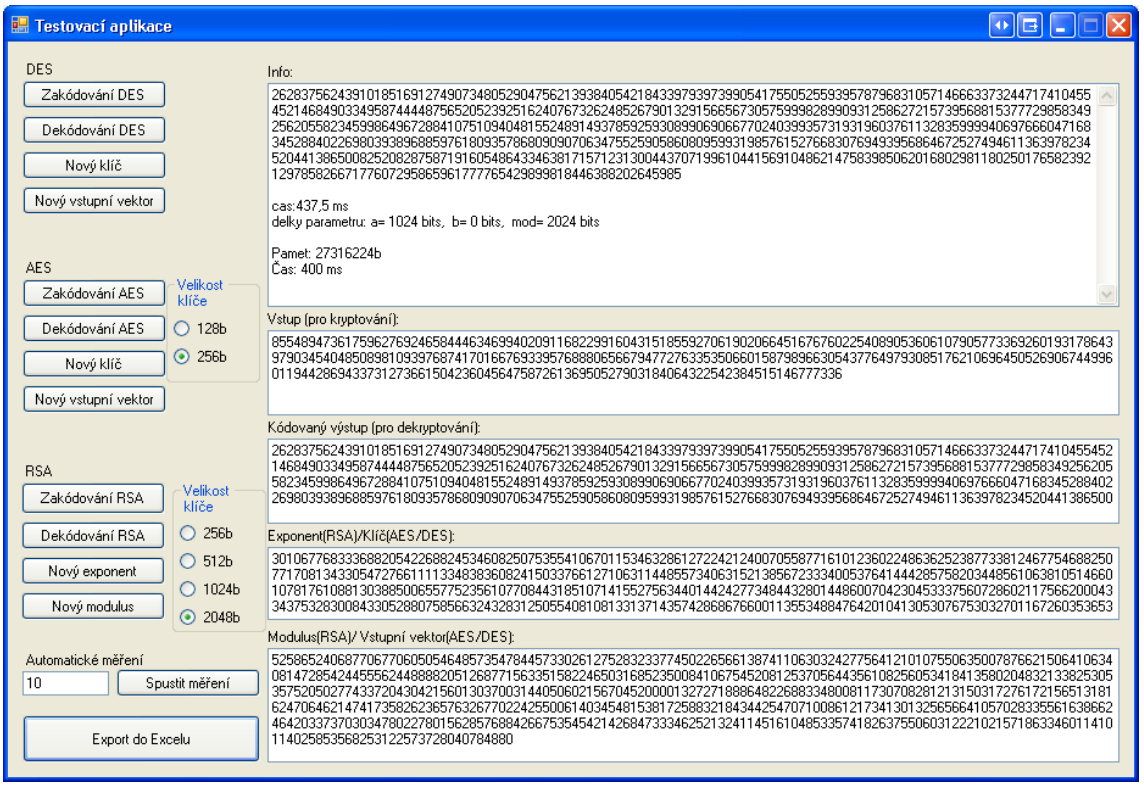
Na dnes používaných čipových kartách lze v podstatě využít všechna výše zmíněná kryptografická primitiva, ale s omezenými parametry, například délkou šifrovacích klíčů. Nejvíce se dnes na čipových kartách využívají právě algoritmy RSA, 3DES, AES a SHA, které jsou přímo podporovány jejich kryptografickými koprocesory, viz tabulka 1. Podle uvedených hardwarových parametrů z tabulky 1 je vidět, že karta Gemalto .NET V2+ je výpočetně výkonnější než SmartCafe Expert 144K.

U mikrokontrolérů chybí nativní podpora kryptografických algoritmů, neboť neobsahují kryptografický koprocesor ani generátor náhodných čísel. U mikrokontrolérů lze kryptografické algoritmy implementovat v jazyce C, viz část 3.4. Důsledkem toho bude vyšší zatížení procesoru a delší doba výpočtu.

4.1 Praktický test s kartou Gemalto .NET Smart Card V2+

V této podkapitole si změříme délku trvání zašifrování a dešifrování zprávy o délce 1024 bitů pomocí kryptografických algoritmů na kartě Gemalto .NET Smart Card V2+. Ze symetrických kryptografických primitiv se jedná o algoritmy AES a DES, z asymetrických o algoritmus RSA.

Klientská testovací aplikace na PC umožňuje změnu velikostí šifrovacích klíčů u DES na 128 nebo 256 bitů. U RSA lze vybrat 256, 512, 1024 nebo 2048 bitů. Se všemi třemi algoritmy lze provést šifrování a dešifrování vstupní zprávy o délce 1024 bitů pomocí tlačítek *Zakódování* a *Dekódování*. V okně *Info* se zobrazuje výstup zašifrování či dešifrování a čas, který tyto výpočty kartě zabraly. Šifrovací klíče pro DES a AES si generuje karta po stisknutí tlačítka *Nový klíč*. V okně *Vstup (pro kryptování)* je zadaná vstupní zpráva o délce 1024b. Exponent a modulus pro RSA jsou generovány klientskou aplikací po stisku tlačítek *Nový Exponent* a *Nový modulus*. V okně *Kódovaný výstup (pro dekryptování)* se zobrazí výsledná zpráva po zašifrování. Lze provést automatická měření po zadání počtu opakování a stisku tlačítka *Spustit měření*. Všechny výsledky je možné exportovat do programu Microsoft Excel po stisku tlačítka *Export do Excelu*. Uspořádání položek v grafickém uživatelském rozhraní klientské aplikace je vidět na obrázku 1.



Obrázek 1: Grafické uživatelské rozhraní klientské aplikace na PC.

4.1.1 Výsledky měření

Měření bylo prováděno pětkrát pro šifrování/dešifrování kryptografickými algoritmy AES, DES a šifrování RSA. Vše pro různé hodnoty šifrovacích klíčů. Všechny výsledky jsou uvedeny v následujících tabulkách 2 až 6, kde je také uvedena délka vstupní zprávy a časová náročnost pro každý výpočet.

Tabulka 2: RSA - šifrování

Délka vstupní zprávy [bit]	Délka klíče [bit]	Čas výpočtu [ms]	Medián času výpočtu [ms]
1024	256	1093,750	
1024	256	781,250	
1024	256	734,375	734,375
1024	256	703,125	
1024	256	687,500	
1024	512	1109,375	
1024	512	953,125	
1024	512	921,875	921,875
1024	512	921,875	
1024	512	906,250	
1024	1024	1578,125	
1024	1024	1468,750	
1024	1024	1437,500	1437,500
1024	1024	1437,500	
1024	1024	1437,500	
1024	2048	765,625	
1024	2048	640,625	
1024	2048	625,000	625,000
1024	2048	593,750	
1024	2048	609,375	

Tabulka 3: DES - šifrování.

Délka vstupní zprávy [bit]	Délka klíče [bit]	Čas výpočtu [ms]	Medián času výpočtu [ms]
1024	64	437,500	
1024	64	375,000	
1024	64	375,000	375,000
1024	64	390,625	
1024	64	375,000	

Tabulka 4: DES - dešifrování.

Délka vstupní zprávy [bit]	Délka klíče [bit]	Čas výpočtu [ms]	Medián času výpočtu [ms]
1088	64	484,375	
1088	64	406,250	
1088	64	421,875	421,875
1088	64	406,250	
1088	64	453,125	

Tabulka 5: AES - šifrování.

Délka vstupní zprávy [bit]	Délka klíče [bit]	Čas výpočtu [ms]	Medián času výpočtu [ms]
1024	128	640,625	640,625
1024	128	640,625	
1024	128	593,750	
1024	128	640,625	
1024	128	609,375	
1024	256	281,250	265,625
1024	256	281,250	
1024	256	265,625	
1024	256	265,625	
1024	256	265,625	

Tabulka 6: AES - dešifrování.

Délka vstupní zprávy [bit]	Délka klíče [bit]	Čas výpočtu [ms]	Medián času výpočtu [ms]
1152	128	640,625	640,625
1152	128	609,375	
1152	128	656,250	
1152	128	593,750	
1152	128	656,250	
1152	256	281,250	296,875
1152	256	296,875	
1152	256	296,875	
1152	256	281,250	
1152	256	296,875	

Medián doby zašifrování zprávy o délce 1024 b pomocí algoritmu RSA pro velikost klíče 256 b je 734,375 ms, pro velikost klíče 512 b je 921,875 ms, pro velikost klíče 1024 b je 1437,500 ms a pro velikost klíče 2048 b je 625,000 ms. Z výsledků měření je patrné, že je karta optimalizovaná (optimalizovaný kryptografický koprocesor) pro použití klíče o velikosti 2048 b u algoritmu RSA. Proto je nejvýhodnější použít tuto délku klíče vzhledem k poskytnuté vyšší bezpečnosti a nejkratší době potřebné k zašifrování.

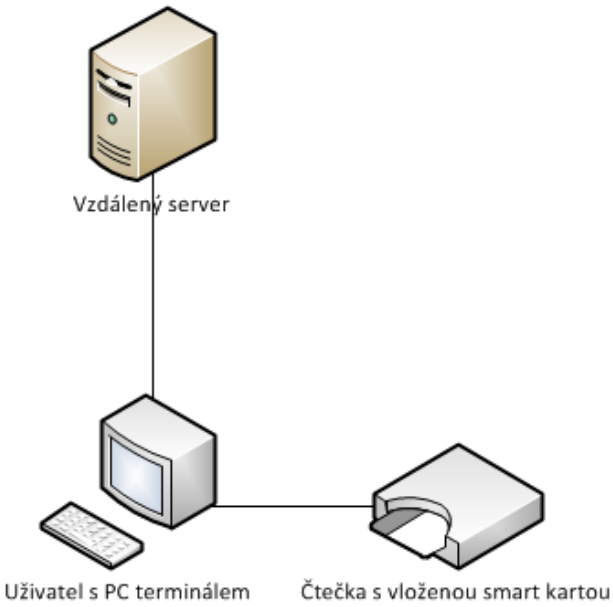
U algoritmu DES, který používá jednu délku klíče, je medián doby zašifrování 375,000 ms a dešifrování 421,875 ms.

Pro algoritmus AES se karta chovala při výpočtech podobně jako v případě RSA, tedy pro velikost klíče 256 b byl medián doby šifrování 265,625 ms. Pro délku klíče 128 b byl medián doby šifrování 640,625 ms. Obdobných výsledků bylo dosaženo při dešifrování, tj. pro délku klíče 256 b byl medián 296,875 ms a pro délku klíče 128 b byl medián 640,625 ms. Z těchto výsledků je opět patrné, že je karta optimalizovaná (optimalizovaný kryptografický koprocesor) pro 256 b klíč.

Vzhledem k délce trvání výpočtů DES a AES je mnohem výhodnější používat algoritmus AES s délkou klíče 256 b z důvodu vyšší bezpečnosti.

5 Autentizační schémata využívající čipové karty

V této kapitole boudou vysvětlena tři autentizační schémata použitelná na smart kartách [25, 26, 27]. Tato autentizační schémata lze v praxi použít pro vzájemnou autentizaci uživatele a vzdáleného serveru pomocí smart karty, což je vidět na obrázku 2. Slovo server je v následujícím textu bráno jako vzdálený server, a ne jako serverová aplikace spuštěná na smart kartě.



Obrázek 2: Komunikace uživatele se vzdáleným serverem s pomocí smart karty.

5.1 Autentizační schéma podle Chiena, Jana a Tsenga

Bylo vybráno schéma z odborného článku An Efficient And Practical Solution to Remote Authentication: Smart Card – autoři Hung-Yu Chien, Jinn-Ke Jan, Yuh-Min Tseng [25].

Bezpečnost tohoto schématu závisí pouze na jednocestné hash funkci MD5 nebo SHA-1. Toto schéma se skládá ze tří fází: registrační, přihlašovací a ověřovací.

Registrační fáze

Při registraci uživatel vyplní do aplikace svoji identitu ID a svoje heslo PW. Systém vypočítá $R = h(ID \oplus x) \oplus PW$, kde x je tajný klíč vedený systémem a $h()$ je hash funkce. Hodnoty $h()$ a R pošle na smart kartu, která si je uloží ve své paměti.

Přihlašovací fáze

Uživatel při přihlášení vloží smart kartu do čtecího zařízení, zadá do aplikace svoji identitu ID a svoje heslo PW. Smart karta následně vypočítá $C_1 = R \oplus PW$, získá aktuální časové razítko T a vypočítá $C_2 = h(C_1 \oplus T)$. Dále odešle zprávu $\{ID, T, C_2\}$ na server.

Ověřovací fáze

Po obdržení zprávy s požadavkem na ověření $\{ID, T, C_2\}$, server a smart karta provedou následující úkony pro umožnění vzájemné autentizace mezi uživatelem a serverem.

1. Systém zkontroluje platnost ID a ověří časový interval mezi T a $T^{'}$ k zabránění odeslání zprávy útočníkem ještě jednou s předchozí hodnotou, případně s odchycenou hodnotou (replay attack), kde $T^{'}$ je časové razítko při přijetí zprávy s požadavkem na ověření.
2. Systém vypočítá $C_1^{'} = h(ID \oplus x)$ a ověří, jestli $C_2 = h(C_1^{'} \oplus T)$. Pokud ověření selže, tak systém vrátí požadavek. Pokud ověření projde, tak systém akceptuje uživatelův požadavek a pokračuje bodem 3.
3. Systém získá aktuální časové razítko $T^{''}$ a vypočítá $C_3 = h(C_1^{'} \oplus T^{''})$. Systém pošle zpět zprávu $\{T^{''}, C_3\}$.
4. Po obdržení zprávy $\{T^{''}, C_3\}$, uživatel ověří platnost časového razítka $T^{''}$. Poté uživatel ověří, jestli $C_3 = h(C_1 \oplus T^{''})$. Pokud ověřování proběhne v pořádku, uživatel je přesvědčen, že odpověď je součástí reálného systému a vzájemná autentizace je dokončena. Pokud by ověřování neproběhlo v pořádku, uživatel zruší spojení.

Bezpečnost

Toto schéma je bezpečné vůči opakovaným útokům, a to buď opakováním předchozí zprávy s žádostí, nebo opakováním zprávy s odpovědí $\{T^{''}, C_3\}$, protože platnost těchto zpráv může být zkontrolována skrze časová razítka.

Pokud útočník zkouší požádat o ověření (autentizaci), připraví si zprávu $\{ID, T, C_2\}$. Tento postup však není možný, neboť útočník nemá k dispozici hodnotu $h(ID \oplus x)$ k výpočtu platné hodnoty C_2.

Pokud by útočník měl platnou zprávu s žádostí $\{ID, T, C_2\}$, nemohl by odvodit platnou zprávu $\{ID, T^{'}, C_2^{'}\}$ z důvodu použití bezpečné jednocestné hash funkce.

Pokud by se pokusil útočníkův server vydávat za pravý a pokusil se podvést uživatele, musel by připravit platnou zprávu $\{T^{''}, C_3\}$. Nicméně je to nemožné, neboť nemá žádný způsob, jak odvodit hodnotu $h(ID \oplus x)$ k výpočtu hodnoty C_3, z důvodu jednocestné hash funkce a opakovanou zprávu lze odhalit pomocí časového razítka.

Efektivnost

Toto autentizační schéma vyžaduje pouze několik hash operací na smart kartě i na serveru a požaduje pouze malý přenos dat, jejichž velikost je závislá na délce zvolených vstupních parametrů (ID, PW). V přihlašovací fázi i v ověřovací fázi provede smart karta jen jednu hash operaci. Kromě parametrů ID a T je počet přenesených bitů při

přenosu zprávy s žádostí o přihlášení roven $|h()|$. Při přenosu zprávy s odpovědí je přeneseno jen $|h()|$ bitů, kromě dat časového razítka $T^{''}$. Toto schéma nepotřebuje ověřovací tabulky, podporuje vzájemnou autentizaci a uživatel má možnost zvolit si libovolné heslo.

5.2 Autentizační schéma podle Chena, Hsianga a Shiha

Jako druhé bylo vybráno schéma z odborného článku Security Improvement on a Remote User Authentication Scheme Using Smart Cards – autoři Tien-Ho Chen, Han-Cheng Hsiang, Wei-Kuan Shih [26]. Tento článek obsahuje 2 způsoby – metoda Hsiang-Shih a její rozšířená verze (An Enhanced Scheme), která byla vybrána.

Bezpečnost tohoto schématu závisí také pouze na jednocestné hash funkci MD5 nebo SHA-1. Toto schéma se opět skládá ze tří fází: registrační, přihlašovací a ověřovací.

Registrační fáze

Při registraci uživatel vyplní do systému svoji identitu ID a svoje heslo PW. Uživatel dále vybere libovolné číslo b, vypočítá hash $h(b \oplus PW)$ a poté odešle serveru bezpečným kanálem $\{ID, h(PW), h(b \oplus PW)\}$. Jakmile server přijme zprávu od uživatele a rozhodne se akceptovat uživatelův požadavek, vypočítá a odešle bezpečným kanálem uživateli zprávu $\{V, R, h()\}$, kde $EID = h(ID \parallel n)$, $P = h(EID \oplus x)$, $R = P \oplus h(b \oplus PW)$, $V = h\big(P \oplus h(b \oplus PW)\big)$, kde x je trvalý tajný klíč serveru, značka $\parallel$ znamená operaci spojení řetězců za sebe a n je počet, kolikrát bylo přeregistrováno heslo. Na smart kartu se uloží data obsahující $(V, R, h())$.

Přihlašovací fáze

Po tom co vyplní uživatel při přihlašování své ID a PW, provede smart karta následující kroky: uživatel musí nejprve udělat ověření $V = h\big(h(EID \oplus x) \oplus h(b \oplus PW)\big)$, a poté generovat R_i a vypočítat $C_1 = R \oplus h(b \oplus PW)$ a $C_2 = h(C_1 \parallel R_i \parallel T_u)$, kde T_u označuje uživatelovo aktuální časové razítko. Uživatel odešle serveru zprávu obsahující $\{ID, C_2, T_u, R_i\}$.

Ověřovací fáze

Po příjmu zprávy s požadavkem na ověření $\{ID, C_2, T_u, R_i\}$, server provede následující operace vedoucí k ověření uživatele:

Server zkontroluje, jestli $T_s - T_u < \Delta T$ v čase T_s (kde T_s označuje aktuální časové razítko a ΔT časový interval pro přenosové zpoždění), a poté vypočítá $C_2^{'} = h(h(EID \oplus x) \parallel R_i \parallel T_u)$ ke kontrole, jestli $C_2^{'}$ je rovno C_2 nebo ne. Jestliže ano, tak platnost uživatele může být zaručena a vypočítá $C_3 = h\big(h(EID \oplus x) \parallel h(T_s)\big)$. Poté server pošle uživateli zprávu $\{C_3, T_s\}$ společným kanálem.

Uživatel zkontroluje, jestli $T_s = T_u$. Pokud ano, tak uživatel ukončí relaci. Jinak uživatel vypočítá $C_3' = h(C_1 \| h(T_s))$ pro kontrolu, jestli C_3' se rovná C_3. Pokud ano, platnost serveru může být zaručena a $h(C_1 \| R_i \| T_u \| h(T_s))$ může být použitý jako token relace pro následnou soukromou komunikaci.

Bezpečnost

Odolá opakovaným útokům: útočník není schopen provést opakovaný útok, pro který potřebuje opakovat zprávu s žádostí od uživatele k serveru $\{ID, C_2, T_u, R_i\}$ nebo od serveru k uživateli $\{C_3, T_s\}$, neboť server zkontroluje, jestli $T_s - T_u < \Delta T$ v čase T_s a útočník nemůže poslat zprávu s žádostí $\{ID, C_2, T_u, R_i\}$ mezi časovým intervalem T_u a T_s. V případě opakování zprávy s žádostí od serveru k uživateli $\{C_3, T_s\}$, opět útočník nemá šifrovací kód C_3, kde $C_3 = h(h(EID \oplus x) \| h(T_s))$ a nemůže generovat C_3 v čase T_s. Což tedy znamená, že toto schéma odolá opakovaným útokům.

Odolá paralelním útokům: předpokládejme, že útočník, který se nacházel mezi serverem a uživatelem, zachytil přihlašovací zprávu od uživatele k serveru $\{ID, C_2, T_u, R_i\}$ a zprávu od serveru k uživateli $\{C_3, T_s\}$. Útočník se bude chtít vydávat za uživatele a přihlásit se k serveru v čase T^*. Útočník vypočítá $T^* = h(T_s) \oplus T_u$ a $C_2^* = C_3$, kde $C_3 = h(h(EID \oplus x) \| h(T_s))$ a poté pošle serveru zprávu $\{ID, C_2^*, T^*, R_i\}$. Protože je T^* platné, server pokračuje výpočtem $C_2' = h(h(EID \oplus x) \| R_i \| T^*)$, což povede k $C_2' = h(h(EID \oplus x) \| R_i \| h(T_s) \oplus T_u)$. Ale C_2' se nerovná $h(C_2^* \| R_i \| T_u)$. Server tedy neakceptuje požadavek na přihlášení od útočníka.

Odolnost vůči útokům zcizení ověření: jelikož v systému neexistuje ověřovací tabulka a uživatelova autentizační data uložená na smart kartě jsou $R = P \oplus h(b \oplus PW)$ a $V = h(P \oplus h(b \oplus PW))$, musel by útočník po krádeži R a V znát ještě heslo PW a hodnotu b, aby mohl projít do další fáze.

Odolnost vůči útokům při zcizení karty: předpokládejme, že útočník získá smart kartu a dešifruje z ní V a R. Útočník by musel získat uživatelovo heslo a libovolné číslo b k výpočtu $C_1 = R \oplus h(b \oplus PW)$ a $C_2 = h(C_1 \| R_i \| T_u)$ pro postup do ověřovací fáze. Na smart kartě není uložené heslo ani hodnota b, tudíž není možná autentizace útočníka.

Odolnost vůči útokům uhodnutím: předpokládejme, že se útočník pokusí zjistit heslo pro přístup k serveru. Útočník by musel odeslat zprávu s žádostí C_2 serveru, kde $C_2 = h(C_1 \| R_i \| T_u)$, kde T_u je aktuální časové razítko. Jelikož útočník nemůže najít stejné náhodné R_i, které je generováno uživatelem v čase T_u, je tak nemožné tento typ útoku provést.

Bezpečnost tohoto schématu je vyšší než u schématu z předchozí podkapitoly 5.1. Je odolné vůči různým útokům. Správce serveru nemůže přečíst heslo uživatele.

Efektivnost

Na serveru nejsou uloženy žádná hesla ani ověřovací tabulky. Uživatel si může svobodně zvolit heslo. Toto autentizační schéma podporuje vzájemnou autentizaci, vyžaduje pouze několik hash operací na smart kartě i na serveru a požaduje malý přenos dat.

5.3 Autentizační schéma podle Lee-Chiua

Třetí variantou se stalo Lee-Chiuovo schéma popsané pod bodem 2 v článku An improved smart card based password authentication scheme with provable security – autoři Jing Xu, Wen-Tao Zhu a Deng-Guo Feng [27].

Bezpečnost tohoto schématu závisí na problému diskrétního logaritmu a na jednocestné hash funkci MD5 nebo SHA-1. Toto schéma se také skládá ze tří fází: registrační, přihlašovací a ověřovací.

Registrační fáze

Uživatel vloží na server svoji identitu ID a svoje heslo PW přes bezpečný kanál. Po obdržení žádosti o registraci server vypočítá $A = h(ID \parallel x)$ a $B = g^{A.h(PW)} \bmod p$, kde p je velké prvočíslo, které zvolí server, g je generátor a taktéž ho volí server a x je tajný klíč serveru. Server uloží na smart kartu $\{ID, A, B, h(), p, g\}$ a vydá ji uživateli.

Přihlašovací fáze

V této fázi vloží uživatel svou smart kartu do čtečky a zadá svoje ID a PW^*. Smart karta následně vykoná tyto kroky.

Vypočítá $B^* = g^{A \cdot h(PW^*)} \bmod p$. Srovná B^* s B. Pokud se nerovnají, zastaví proceduru. Vypočítá $Z = B^* A \bmod p$ a $C_1 = h(T \oplus B^*)$, kde T je aktuální čas připojeného zařízení. Odešle zprávu s žádostí o přihlášení $m = \{ID, Z, C_1, T\}$ vzdálenému serveru.

Ověřovací fáze

Tuto fázi vykonává server kvůli kontrole, zda se může uživatel přihlásit nebo ne. T' je čas, kdy server přijal zprávu s žádostí o přihlášení m: Zkontroluje, jestli ID je platný identifikátor uživatele. Pokud ne, tak vrátí žádost o přihlášení. Zkontroluje, jestli rozdíl mezi T s T' je v předem definovaném rozsahu. Pokud ne, tak vrátí žádost o přihlášení. Vypočítá $A^* = h(ID \parallel x)$ a ${C_1}^* = h[T \oplus (Z / A^* \bmod p)]$. Pokud se $C_1 = {C_1}^*$, akceptuje žádost o přihlášení. Jinak ji vrátí zpět.

Bezpečnost

Útočník nemůže zjistit tajný klíč x serveru z $A = h(ID \parallel x)$, protože je použitá jednocestná hash funkce.

Útočník nemůže vytvořit platnou zprávu $m = \{ID, Z, C_1, T\}$, protože by musel získat B^* k výpočtu C_1. K tomu by potřeboval znát uživatelovo heslo PW a tajný klíč serveru x.

Dále je toto schéma odolné vůči opakovanému útoku, kdy server zkontroluje prodlevu díky časovým razítkům s předem definovaným rozsahem jejich rozdílu.

Je citlivé na útoky založené na padělání, pokud útočník získá informace uložené na kartě.

Efektivnost

Při zápisu na kartu je větší přenos dat než u předchozích dvou metod. Výpočetní operace jsou pro kartu náročnější z důvodu operací v modulární aritmetice než v předchozích schématech. Uživatel si může svobodně zvolit heslo.

6 Vlastní návrh autentizačního schématu

Při návrhu tohoto čtvrtého autentizačního schématu bylo vycházeno z druhého autentizačního schématu z podkapitoly 5.2, které se jeví jako nejbezpečnější ze všech tří.

6.1 Popis schématu

Návrh tohoto schématu se skládá ze tří fází: registrační, přihlašovací a ověřovací.

Registrační fáze

Uživatel si zvolí svoje ID, heslo PW a číslo b, které je vyžadováno při přihlášení. Při registraci se provede operace XOR vybraného čísla b s heslem PW, vypočítá hash se a odešle se zpráva obsahující ID, hash z hesla PW a hash $b \oplus PW$. Server při registraci vytváří postupně tyto proměnné. EID, což je hash operace spojení řetězců ID a n, tedy vypočítá se hash $ID \parallel n$, kde n je počet, kolikrát bylo přeregistrováno heslo. Další proměnná je $P = h(EID \oplus x)$, kde x je opět systémová proměnná. Dále vypočítá další proměnnou $R = P \oplus h(b \oplus PW)$ a výsledně V, což je hash hodnoty R. Na smart kartu se uloží V, R a $h(EID \oplus x)$.

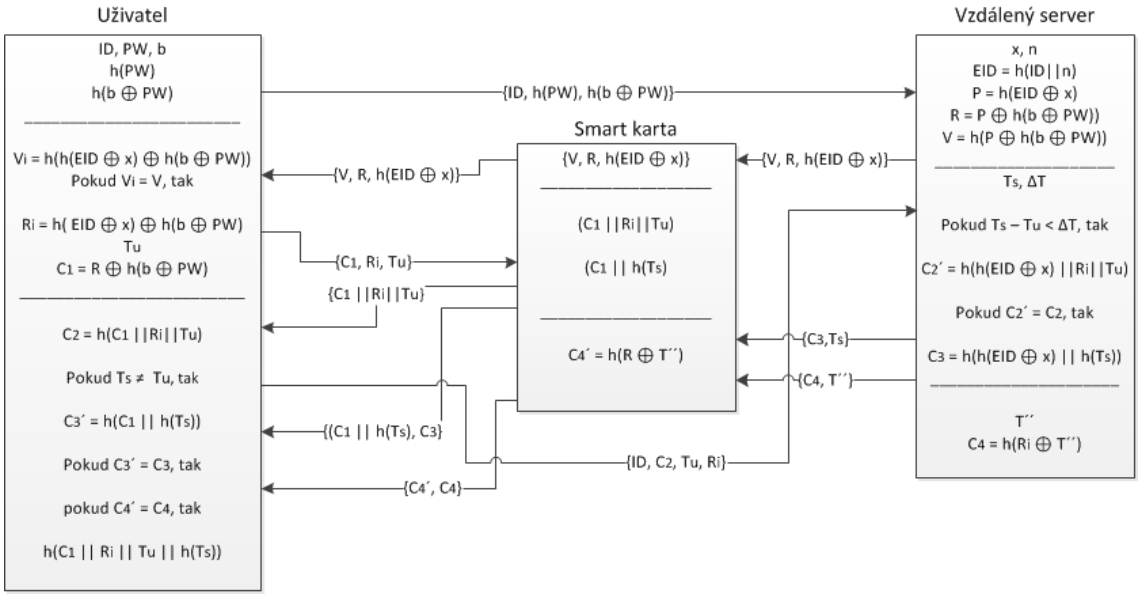
Přihlašovací fáze

Při přihlašování uživatel vyplní ID, heslo PW a číslo b. S těmito údaji se vypočítá zrcadlové V_i a smart karta načte uložené V a porovnají se. Vytvoří se i zrcadlové R_i a vypočítá se C_1 jako $R \oplus h(b \oplus PW)$ a vrátí C_2 jako $(C_1 \parallel R_i \parallel T_u)$, z čehož systém poté vypočítá hash. T_u je aktuální časové razítko. Toto se posílá do ověřovací fáze.

Ověřovací fáze

Opět se porovnávají časy, tento krát i jestli nejsou stejné – v případě paralelního útoku. Vytvoří se zrcadlové $C_2^{´}$ z hodnot $h(h(EID \oplus x) \parallel R_i \parallel T_u)$ a $C_2^{´}$ se porovná s C_2. Výsledně je vytvořeno C_3 jako $h\big(h(EID \oplus x) \parallel h(T_s)\big)$, které se pošle na smart kartu, která vrátí $C_3^{´}$ jako $h\big(C_1 \parallel h(T_s)\big)$. Pokud se $C_3^{´}$ a C_3 rovnají, tak pokračuje dál. Je připojeno další časové razítko T´´. Server vypočítá $C_4 = h\big(R_i \oplus T^{´´}\big)$. Ověření razítka potom probíhá i opětovně na kartě výpočtem $C_4^{´} = h\big(R \oplus T^{´´}\big)$ z přijaté zprávy od serveru $\{C_4, T^{´´}\}$. Pokud se $C_4^{´}$ rovná C_4, vše proběhlo v pořádku a vytvoří se token $h\big(C_1 \parallel R_i \parallel T_u \parallel h(T_s)\big)$ pro další komunikaci. Toto další ověření zabezpečí ukradnutí v době poslední komunikace – relace tokenu, kdy ještě před jeho předáním ověří výsledek novějšího času sama smart karta, což je ochrana i v případě útočníka vydávajícího se za server.

Struktura tohoto autentizačního schématu je zobrazena na obrázku 3.



Obrázek 3: Autentizační schéma podle vlastního návrhu.

6.2 Bezpečnostní analýza schématu

Odolnost vůči útokům zcizení ověření: pokud by útočník získal R ($R = P \oplus h(b \oplus PW)$) a V ($V = h\big(P \oplus h(b \oplus PW)\big)$), která jsou uložená na smart kartě, musel by znát ještě uživatelovo heslo PW a hodnotu čísla b, aby mohl pokračovat v autentizaci.

Odolnost vůči útokům při zcizení karty: pokud by útočník získal smart kartu a extrahoval z ní V a R, potřeboval by stále znát uživatelovo heslo PW a hodnotu čísla b, protože tyto hodnoty nejsou na smart kartě uloženy, aby mohl vypočítat $C_1 = R \oplus h(b \oplus PW)$ a $C_2 = h(C_1 \parallel R_i \parallel T_u)$ pro další postup v autentizaci do ověřovací fáze.

Odolá opakovaným útokům: pro opakovaný útok má útočník jako první možnost opakovat zprávu s žádostí od uživatele k serveru $\{ID, C_2, T_u, R_i\}$, což se mu nepodaří, neboť server si zkontroluje, jestli $T_s - T_u < \Delta T$ v čase T_s. Útočník by musel poslat zprávu s žádostí $\{ID, C_2, T_u, R_i\}$ mezi časovým intervalem T_u a T_s, což útočník nezvládne. Jako druhou možnost, může útočník opakovat zprávu s žádostí od serveru k uživateli $\{C_3, T_s\}$. Ale jelikož útočník nemá vypočítanou hodnotu C_3, kde $C_3 = h\big(h(EID \oplus x) \parallel h(T_s)\big)$, a ani nemůže generovat C_3 v čase T_s, nepodaří se mu tento útok provést.

Odolá paralelním útokům: pokud by útočník, který se nachází mezi serverem a uživatelem, zachytil přihlašovací zprávu od uživatele k serveru $\{ID, C_2, T_u, R_i\}$ a zprávu od serveru k uživateli $\{C_3, T_s\}$ a vydával se za uživatele a chtěl se přihlásit k serveru v čase $T^{´´´}$, musel by vypočítat $T^{´´´} = h(T_s) \oplus T_u$ a $C_2^{´´´} = C_3$, kde $C_3 = h\big(h(EID \oplus x) \parallel h(T_s)\big)$. Pak by odeslal serveru zprávu $\big\{ID, C_2^{´´´}, T^{´´´}, R_i\big\}$. Protože je $T^{´´´}$ platné, server by pokračoval výpočtem $C_2^{´} = h\big(h(EID \oplus x) \parallel R_i \parallel T^{´´´}\big)$, což by vedlo

k $C_2^{'} = h(h(EID \oplus x) \| R_i \| h(T_s) \oplus T_u)$. Ale $C_2^{'}$ by se nerovnalo $h(C_2^{'''} \| R_i \| T_u)$ a z tohoto důvodu by server odmítnul žádost o přihlášení od útočníka.

Odolnost vůči útokům uhodnutím: pokud by útočník chtěl získat heslo pro přístup k serveru, musel by poslat zprávu s žádostí C_2 serveru, kde $C_2 = h(C_1 \| R_i \| T_u)$, T_u je aktuální časové razítko. Protože útočník nemůže vypočítat stejné R_i, které generuje uživatel v čase T_u, je tak tento druh útoku neproveditelný.

Odolnost vůči útokům založených na padělání: pokud by se chtěl útočník ve fázi ověřování vydávat za server nebo za uživatele, nevypočítá stejnou hodnotu $C_4 = h(R_i \oplus T^{''})$ jako smart karta $C_4^{'} = h(R \oplus T^{''})$, bez znalosti časového razítka $T^{''}$, v případě vydávání se za uživatele, nebo bez znalosti hodnoty R, v případě vydávání se za server.

Bezpečnost tohoto schématu je vyšší než u schémat z předchozí kapitoly 5. Je odolné vůči všem výše zmíněným útokům.

Při návrhu autentizačního schématu bylo více přihlédnuto k bezpečnosti než k rychlosti. Vzhledem k tomu, že využívá hlavně operace XOR a hash funkcí, zůstává toto autentizační schéma stále efektivní.

7 Implementace čtyř autentizačních schémat do jedné aplikace

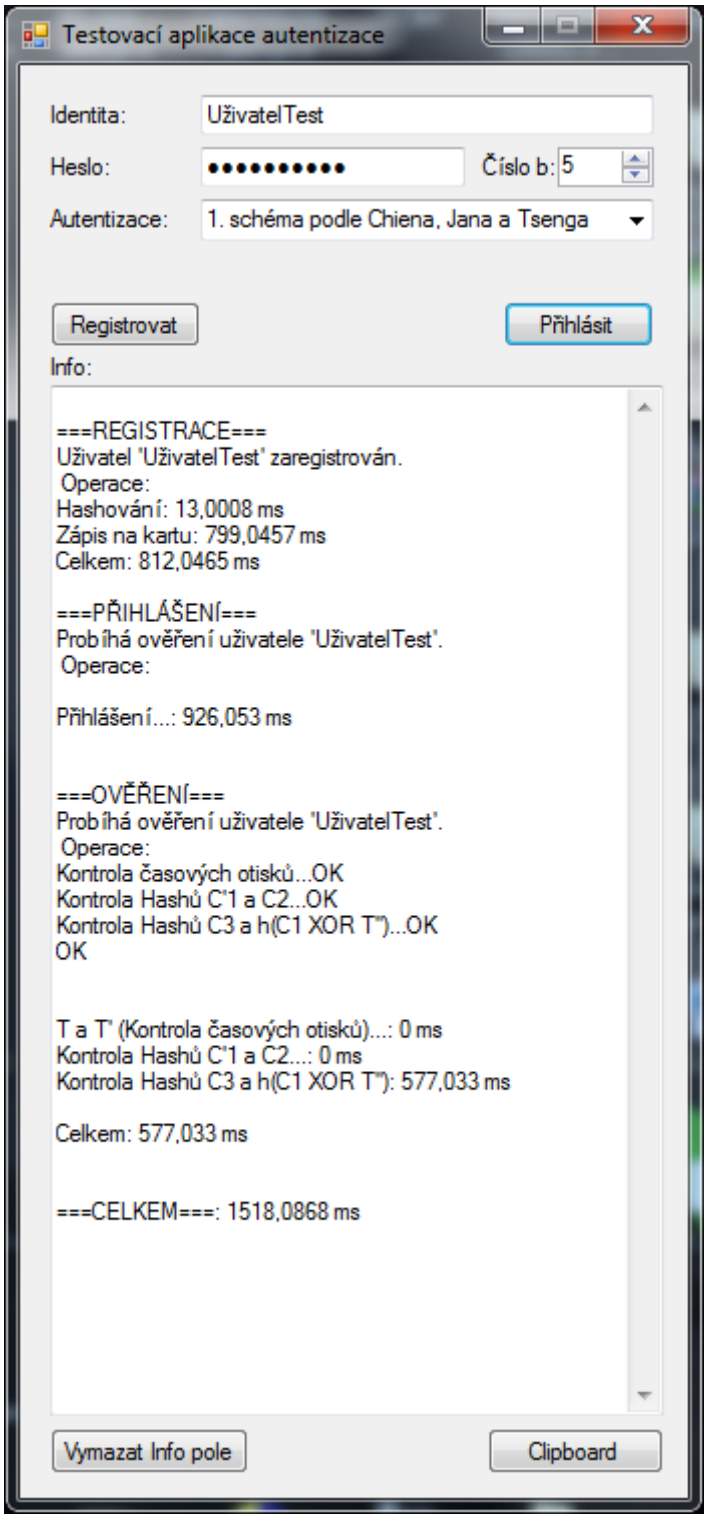
V této kapitole je ukázána implementace všech čtyř autentizačních schémat do jednoho grafického rozhraní v Microsoft Visual Studiu 2008 s doinstalovaným .NET SmartCard Framework SDK pro Gemalto .NET Smart Card V2+, která je na obrázku 4. Všechna schémata byla naprogramována podle postupů jednotlivých fází autentizace popsaných v 5. a 6. Kapitole.

Klientská aplikace na PC nám slouží jako vzdálený server i jako registrační a přihlašovací aplikace na PC. Serverová aplikace na smart kartě má na starosti všechny potřebné procedury, které je nutné provádět na kartě podle všech autentizačních schémat.

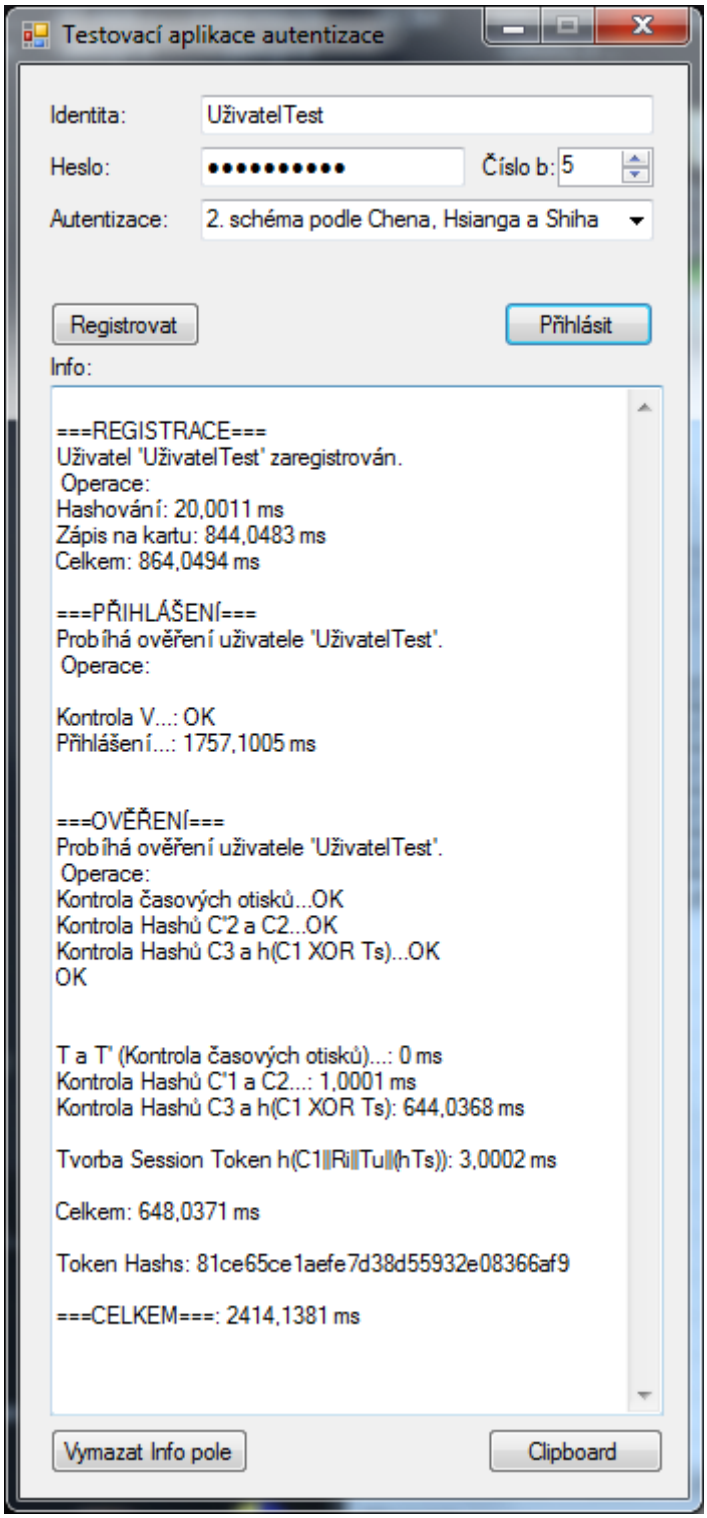
Obrázek 4: Čtečka smart karet (vlevo) a karta Gemalto .NET Smart Card V2+ (vpravo).

V klientské aplikaci na PC s názvem *Testovací aplikace autentizace* uživatel napíše do textového pole s názvem *Identita* například svoje jméno a do pole *Heslo* své heslo, vybere si z rozbalovacího seznamu *Autentizace* autentizační schéma, které bude chtít použít (*1. schéma podle Chiena, Jana a Tsenga, 2. schéma podle Chena, Hsianga a Shiha, 3. schéma podle Lee-Chiua , Návrh 4. autentizačního schématu*). Při použití autentizačních schémat *2. schéma podle Chena, Hsianga a Shiha*, *Návrh 4. autentizačního schématu* je dobré zvolit hodnotu čísla *b* v příslušném poli s názvem *Číslo b*. Toto číslo spolu s identitou a heslem si musí uživatel pamatovat. Nyní stiskem tlačítka *Registrovat* dojde k zaregistrování příslušného uživatele. Po vyplnění Identity, hesla uživatele dojde po stisku tlačítka *Přihlásit* k proběhnutí zbývajících fází autentizace, tedy k přihlášení a k verifikaci (Identita, heslo i číslo *b* zůstává vyplněno, dokud není změněno uživatelem z důvodu rychlejší práce s aplikací). Vše je oznamováno v poli *Info*, kde se zobrazuje textový výstup s popisem jednotlivých fází autentizačních schémat a čas, který tyto fáze a jejich výpočetní části zabraly. Tento text je možné zkopírovat do schránky pomocí tlačítka *Clipboard* nebo vymazat pomocí tlačítka *Vymazat Info pole*.

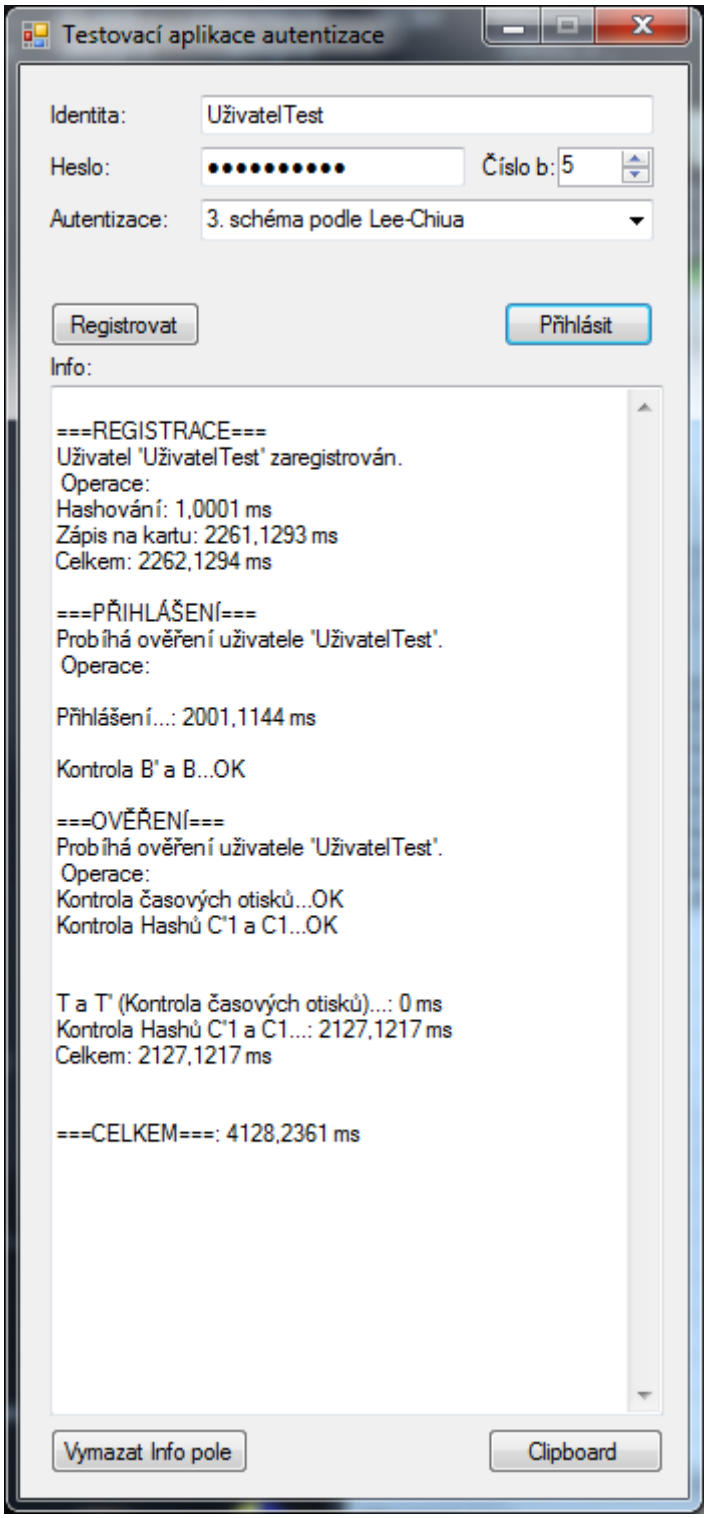
Uspořádání položek v grafickém uživatelském rozhraní klientské aplikace a její výstupní data po proběhlých fázích vzájemné autentizace jednotlivých autentizačních schémat je vidět na obrázcích 5 až 8.



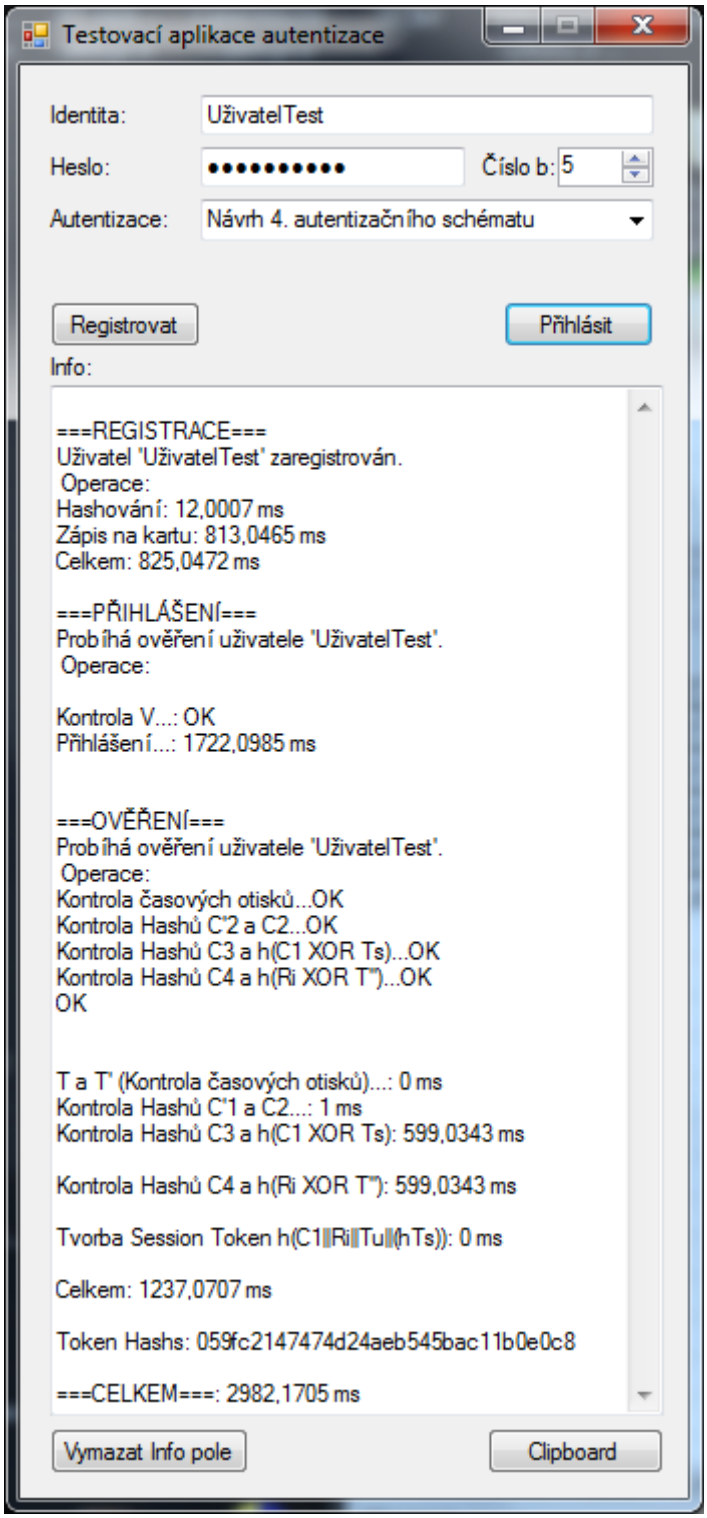
Obrázek 5: Uživatelské rozhraní klientské aplikace s výstupními daty autentizačního schématu podle Chiena, Jana a Tsenga.



Obrázek 6: Uživatelské rozhraní klientské aplikace s výstupními daty autentizačního schématu podle Chena, Hsianga a Shiha.



Obrázek 7: Uživatelské rozhraní klientské aplikace s výstupními daty autentizačního schématu podle Lee-Chiua.



Obrázek 8: Uživatelské rozhraní klientské aplikace s výstupními daty vlastního návrhu autentizačního schématu.

7.1 Výsledky měření

Měření délky trvání jednotlivých fází autentizace bylo prováděno pětkrát pro každé autentizační schéma na notebooku ASUS M50VC s procesorem Intel Core 2 Duo P8400, 2 jádra, 2,2 GHz a pamětí RAM 4 GB, s připojenou čtečkou smart karet a smart

kartou Gemalto .NET Smart Card V2+. Při testování všech schémat bylo dosazeno za Identitu ID „UživatelTest“, za heslo PW „b347ec8395“ a za číslo b „5“.

Všechny výsledky jsou uvedeny v následující tabulce 7, kde je uvedena délka registrace, přihlášení, ověření a celková délka těchto úkonů v milisekundách a medián všech těchto hodnot pro každé autentizační schéma. V tabulce 8 se nachází shrnutí útoků, vůči kterým jsou daná autentizační schémata odolná.

Tabulka 7: délka trvání jednotlivých úkonů při autentiazací u jednotlivých schémat

	Počet pokusů a jejich medián	Délka registrace [ms]	Délka přihlášení [ms]	Délka ověření [ms]	Celková délka [ms]
Schéma podle Chiena, Jana a Tsenga	1.	812,047	926,053	577,033	2315,133
	2.	642,037	912,052	598,034	2152,123
	3.	492,028	940,054	581,033	2013,115
	4.	496,028	913,052	581,033	1990,113
	5.	575,033	914,052	588,034	2077,119
	Medián	575,033	914,052	581,033	2077,119
Schéma podle Chena, Hsianga a Shiha	1.	864,049	1757,101	648,037	3269,187
	2.	805,046	1740,010	634,036	3179,092
	3.	800,046	1784,102	598,034	3182,182
	4.	863,050	1976,113	617,035	3456,198
	5.	875,050	1775,102	586,034	3236,186
	Medián	863,050	1775,102	617,035	3236,186
Schéma podle Lee-Chiua	1.	2262,129	2001,114	2127,122	6390,365
	2.	2208,126	2082,119	1961,112	6251,357
	3.	2279,130	2085,119	2032,116	6396,365
	4.	2258,129	2252,128	2034,116	6544,373
	5.	2262,129	2076,119	2157,123	6495,371
	Medián	2262,129	2082,119	2034,116	6396,365
Schéma podle vlastního návrhu	1.	825,047	1722,099	1237,071	3784,217
	2.	777,044	1707,098	1232,075	3716,217
	3.	848,049	1769,101	1186,068	3803,218
	4.	851,049	1619,093	1227,070	3697,212
	5.	833,048	1731,099	1260,072	3824,219
	Medián	833,048	1722,099	1232,075	3784,217

Tabulka 8: porovnání bezpečnosti autentizačních schémat

	Odolnost vůči:
Schéma podle Chiena, Jana a Tsenga	opakovaným útokům, útokům zcizení ověření, útokům při zcizení karty
Schéma podle Chena, Hsianga a Shiha	opakovaným útokům, paralelním útokům, útokům zcizení ověření, útokům při zcizení karty a uhodnutým útokům
Schéma podle Lee-Chiua	opakovaným útokům, útokům založeným na padělání (pokud nedojde k odcizení karty)
Schéma podle vlastního návrhu	útokům zcizení ověření, útokům při zcizení karty, opakovaným útokům, paralelním útokům, uhodnutým útokům, útokům založeným na padělání

Nejkratší délku registrace vykazuje schéma podle Chiena, Jana a Tsenga s hodnotou mediánu 575,033 ms. Toto schéma má také nejkratší délku přihlášení s mediánem 914,052 ms a ověření s mediánem 581,033 ms. Naopak nejdelší délky dosahuje schéma podle Lee-Chiua u registrační fáze s hodnotou mediánu 2262,129 ms, také u přihlašovací fáze 2082,119 ms a i u ověřovací fáze 2034,116 ms.

Výsledky měření potvrdily některé teoretické předpoklady k jednotlivým schématům uvedené v 5. a 6. kapitole.

Z výsledků měření je patrné, že nejdelší dobu zpracování autentizace má schéma podle Lee-Chiua, hlavně z důvodu používání výpočetně náročných operací modulární aritmetiky, které u zbylých schémat nejsou použité. Samozřejmě využívá i operace XOR a hash funkce. Každá z fází trvá přes 2 sekundy. Toto schéma je tedy z časových důvodů nejméně efektivní.

Nejkratší délku celé autentizace splňuje schéma podle Chiena, Jana a Tsenga. Používá pouze operaci XOR a hash funkci. Ze všech čtyř schémat má nejmenší množství přenesených dat a nejmenší množství výpočetních operací. Z těchto důvodů je tedy nejefektivnější.

Schéma podle Chena, Hsianga a Shiha používá také pouze operaci XOR a hash funkci, ale obsahuje více výpočtů a přenosů dat než schéma podle Chiena, Jana a Tsenga, ale je více bezpečné. S celkovým časem 3 sekund je stále ještě efektivní.

Vlastní schéma bylo navrženo tak, aby bylo co nejvíce bezpečné, ale zároveň časově efektivní, což se projevilo ve výsledcích. Nejvíce je rozdíl patrný v ověřovací části, kde přibylo 5 procedur (hash, XOR) navíc oproti druhému schématu, a doba trvání této fáze se tím zvedla na dvojnásobek.

Velký podíl na délce trvání jednotlivých fází u všech schémat má také doba zápisu do paměti smart karty a čtení z paměti smart karty.

8 Závěr

V této diplomové práci byly uvedeny bezpečnostní služby, které kryptografie poskytuje, spolu s obecnými kryptografickými primitivy. Dále byla objasněna zařízení s nižším výpočetním výkonem, tj. smart karty, RFID tagy a mikrokontroléry. Dále jsou v práci blíže představeny smart karty Gemalto .NET Smart Card V2+ a SmartCafe Expert 144K. a mikrokontroléry ATmega128, Stellaris LM3S8962.

Byla naprogramována aplikace pro zjištění délky trvání šifrování na smart kartě Gemalto .NET Smart Card V2+. Z výsledků měření vyplývá, že pro šifrování a dešifrování prováděné kartou je optimální používat hodnoty klíčů 2048 b pro algoritmus RSA a 256 b pro algoritmus AES. Na přiloženém CD se nachází hotové Java applety se stejnou funkcí jako aplikace v případě Gemalto .NET Smart Card V2+ pro smart kartu SmartCafe Expert 144K sloužící pro jejich porovnání.

Dále byla představena autentizační schémata a jejich vnitřní postupy pro vzájemnou autentizaci uživatele a vzdáleného serveru pomocí smart karty. Po prostudování vnitřních postupů těchto schémat byl položen návrh autentizačního schématu, který sice zvyšuje bezpečnost vzájemné autentizace, ale na úkor efektivnosti.

Z důvodu praktického ověření byla provedena implementace všech autentizačních schémat do jedné aplikace na kartě Gemalto .NET Smart Card V2+, a jejich otestování a vzájemné porovnání z hlediska výkonnosti a efektivity. Autentizační schéma podle mého návrhu je nejbezpečnější ze všech čtyř schémat. Délka všech fází vzájemné autentizace u tohoto schématu byla podle naměřených hodnot do 4 sekund, což je přijatelné vzhledem ke zvýšené bezpečnosti i vzhledem k délce načítání operačního systému, pokud by se autentizace použila například při přihlašování do Microsoft Windows či jiného operačního systému nebo chráněných aplikací. Výsledkem této implementace byla aplikace, která dovoluje otestování chování, změnu parametrů a průběh jednotlivých fází u těchto 4 různých autentizačních schémat.

Všechna autentizační schémata by bylo možné použít v praktickém nasazení, jelikož časy nejsou nijak velké, ale z důvodů bezpečnosti je nejlepší použít z nich pro vzdálenou autentizaci schéma podle Chena, Hsianga a Shiha nebo bezpečnější autentizační schéma podle mého návrhu.

Seznam použité literatury

[1] MENEZES, J. Alfred, OORSCHOT, C. Paul, VANSTONE, A. Scott. *Handbook of applied cryptography*. CRC Press, 1996. 780 s. ISBN 0-8493-8523-7.

[2] BURDA, Karel. *Bezpečnost informačních systémů*. Brno : FEKT VUT Brno, 2005. 104 s.

[3] MAYES, E. Keith; MARKANTONAKIS, Konstantinos. *Smart Cards, Tokens, Security and Applications*. New York : Springer Science + Business Media, 2008. 392 s. Dostupné z WWW: <http://www.scribd.com/doc/44639348/Smart-Cards-Tokens-Security-and-Applications>. ISBN 978-0-387-72197-2.

[4] STALLINGS, William. *Cryptography and Network Security. 4th edition*. 2006. 592 s. ISBN 0131873164.

[5] COLE, Peter, RANASINGHE, Damith. *Networked RFID systems and lightweight cryptography: raising barriers to product counterfeiting*. 2008. 355 s. ISBN 3540716408.

[6] SPÁČIL, Viktor. *Užití čipové karty ke generování a správě šifrovacích klíčů*. Brno, 2009. 35 s. Bakalářská práce. Masarykova univerzita, fakulta informatiky.

[7] *Stellaris® LM3S8962 Microcontroller: Data sheet*. Austin: Texas Instruments Incorporated, 2011. 785 s.

[8] *Atmel ATmega128 : Data sheet*. San Jose, USA : Atmel Corporation, 2011. 386 s.

[9] DUDÁČEK, K.. *Mikrokontroléry*. 2001. 7 s. Dostupné z WWW: <http://home.zcu.cz/~dudacek/Pot/mikrokontrolery.pdf/>.

[10] *Gemalto .NET Smart Card: Integration Guide*. France : GEMALTO, B.P. 100, 13881 GEMENOS CEDEX, 2011. 179 s.

[11] *Sm@rtCafé® Expert: Java-Based, Open-Platform Card and Token Operating System*. Munich, Germany : Giesecke & Devrient GmbH, 2010. 4 s.

[12] *Smart Card Forum 2010* [online]. Verze 2010. [cit. 2011-13-12]. Dostupné z WWW: <http://www.smartcardforum.cz/>.

[13] *Smart Card Forum 2011* [online]. Verze 2011. [cit. 2011-13-12]. Dostupné z WWW: <http://www.smartcardforum.cz/>.

[14] *ORACLE: Java Card Technology* [online]. Oracle Corporation, 500 Oracle Parkway, Red Wood Shores: 2010 [cit. 2011-13-12]. Dostupné z WWW: <http://www.oracle.com/technetwork/java/javacard/overview/index.html>.

[15] POSCHMANN, York Axel. *LIGHTWEIGHT CRYPTOGRAPHY: Cryptographic Engineering for a Pervasive World*. Bochum, 2009. 179 s. Dizertační práce. Faculty of Electrical Engineering and Information Technology, Ruhr-University Bochum, Germany.

[16] VLÁŠEK, Petr. *Demonstrace šifrování na platformě Java Card*. Praha, 2008. 133 s. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická.

[17] ROSA, Tomáš; KLÍMA, Vlastimil. Http://crypto.hyperlink.cz [online]. únor 2007 [cit. 2011-13-12]. *Bezkontaktní karty MIFARE*. Dostupné z WWW: <http://crypto.hyperlink.cz/files/ST_2007_02_x_x.pdf>.

[18] ROSA, Tomáš; KLÍMA, Vlastimil. Http://crypto.hyperlink.cz [online]. leden 2008 [cit. 2011-13-12]. *Bezpečněji s MIFARE*. Dostupné z WWW: <http://crypto.hyperlink.cz/files/ST_2008_01_x_x.pdf>.

[19] *NXP* [online]. 2011 [cit. 2011-12-13]. The leading industry standard for contactless and dual interface smart card schemes. Dostupné z WWW: <http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/#description>

[20] MALINA, L.; HAJNÝ, J. *Accelerated Modular Arithmetic for Low-Performance Devices*. In 34th International Conference on Telecommunications and Signal Processing (TSP 2011). 2011. s. 1-5. ISBN: 978-1-4577-1409- 2.

[21] LABAK FI MUNI. *JavaCard applet development with NetBeans IDE* [online]. 2011 [cit. 2011-12-13]. Dostupné z WWW: <https://minotaur.fi.muni.cz:8443/~xsvenda/docuwiki/doku.php?id=public:smartcard:javacardcompilation>.

[22] FIALA, Martin. *Software pro simulaci řídících algoritmů s generováním C-kódu pro mikrokontrolér*. Brno, 2011. 66 s. Diplomová práce. Vysoké učení technické v Brně.

[23] ŠEVČÍK, Michal. *Moderní autentizace předmětem a znalostí*. Brno, 2011. 48 s. Bakalářská práce. Vysoké učení technické v Brně.

[24] *RFID-NFC* [online]. 2011 [cit. 2011-12-13]. What is NFC. Dostupné z WWW: <http://www.rfid-nfc.eu/what-is-nfc>.

[25] CHIEN, Hung-Yu, JAN, Jinn-Ke, TSENG, Yuh-Min. *An Efficient and Practical Solution to Remote Authentication: Smart Card.* Great Britain, 2002. 4 s. Computers & Security, Elsevier Science Ltd.

[26] CHEN, Tien-Ho, HSIANG, Han-Cheng, SHIH, Wei-Kuan. *Security Improvement on a Remote User Authentication Scheme Using Smart Cards.* Berlin, 2010. 8 s. Springer-Verlag.

[27] XU, Jing, ZHU, Wen-Tao, FENG, Deng-Guo. *An improved smart card based password authentication scheme with provable security.* Great Britain, 2008. 6 s. Computer Standards & Interfaces, Elsevier B. V.

Seznam zkratek

AES	Advanced Encryption Standard
CAN	Controller Area Network
CAP	Converted Applet
CIL	Common Intermediate Language
CLI	Common Language Infrastructure
CLR	Common Language Runtime
DES	Data Encryption Standard
DH	Diffie-Hellman
DMA	Direct Memory Access
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptography
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPROM	Erasable Programmable Read-Only Memory
FISH	Fibonacci Shrinking
IO	Input Output
JCAPI	Java Card Application Programming Interface
JCRE	Java Card Runtime Enviroment
JCVM	Java Card Virtual Machine
MD5	Message Digest 5 algorithm
NFC	Near Field Communication
OCV	Off-Card Verifier
PKI	Public Key Infrastructure
RAM	Random Access Memory
RC4	Alleged RC4
RFID	Radio Frequency Identification Systems
ROM	Read-Only Memory
RSA	Rivest Shamir Adleman
RTC	Real Time Clock
SFR	Special Function Registers
SFR	Special Function Registers
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module
SPI	Serial Peripheral Interface
URI	Uniform Resource Identifier
XOR	Exclusive Or

Seznam příloh

Příloha A: CD obsahující elektronickou verzi diplomové práce, zdrojové kódy vytvořených programů: dva projekty vytvořené v Microsoft Visual Studio 2008 a zdrojové kódy pro Java kartu.