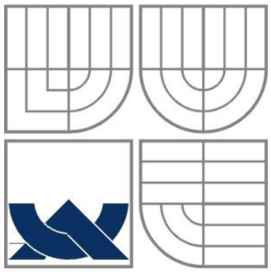
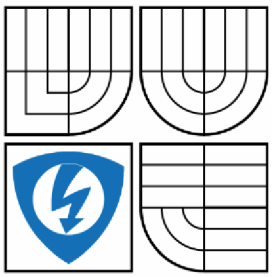


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A
KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ZABEZPEČENÍ OPERAČNÍHO SYSTÉMU APPLE IOS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

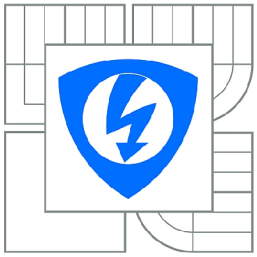
AUTOR PRÁCE
AUTHOR

Zdeněk Zahradníček

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Jan Hajný, Ph.D.

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Zdeněk Zahradníček ***ID:*** 134441
Ročník: 3 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Bezpečnost operačního systému Apple iOS

POKYNY PRO VYPRACOVÁNÍ:

Téma je zaměřeno na analýzu operačního systému Apple iOS na zařízeních iPhone z pohledu bezpečnosti. Cílem práce je nastudovat strukturu operačního systému iOS a analyzovat bezpečnostní mechanismy na této platformě. Výstupem projektu bude zhodnocení použitých mechanismů z pohledu kryptografie a demonstrace těchto mechanismů na reálné aplikaci. Důraz bude kladen na funkčnost aplikace a na vhodný výběr bezpečnostních mechanismů pro demonstraci zabezpečení iOS.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and Network Security: Principles and Practice (5th Edition). USA : Prentice Hall, 2010. 744 s. ISBN 0136097049.

[2] IOS Security. [online]. 2012 [cit. 2012-10-09]. Dostupné z: http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 5.6.2013

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

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

UPOZORNĚNÍ:

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

ABSTRAKT

Bakalářská práce se zabývá zabezpečením operačního systému iOS a vytvářením aplikací pro tento systém. Popisuje základní vlastnosti systému, použitá zabezpečení, šifrování a třídy pro bezpečné ukládání souborů. Další oblastí, kterou se práce zabývá je zabezpečení přístupu k internetu a ochranou odesílaných a přijímaných dat.

V práci je popsán programovací balíček Xcode, používaný vývojáři pro iOS platformu. Dále byla zkompilována aplikace pro ukázku práce s tímto programovacím balíčkem a její následné otestování na virtuálním zařízení v simulátoru, který je součástí tohoto balíčku. V poslední řadě byla vytvořena aplikace využívající velkých česel o hodnotě 1024bit a následné využití těchto čísel pro početní operace. Výstupem aplikace byla průměrná doba trvání výpočtu použitých matematických operací.

KLÍČOVÁ SLOVA

Apple, iOS, Xcode, iPhone, iPad, šifrování, zabezpečení,Xcode

ABSTRACT

Bachelor´s thesis addresses with the iOS operating system security and creating applications for this system. Describes the basic features of the system, security, encryption, and classes for secure file storage. Another area that deals with the work is that access to the Internet and the protection of transmitted and received data.

The work is described Xcode programming package, used by developers for iOS platform. Furthermore, the compiled application to examples of work with this programming package and its subsequent testing in a virtual device simulator which is part of this package. I did application that uses large numbers of 1024bit value and then use these numbers for calculations. The output of application was average time of measuring duration of calculating the mathematical operations.

KEYWORDS

Apple, iOS, Xcode, iPhone, iPad, Encryption, Security, Xcode

ZAHRADNÍČEK, Z. *Bezpečnost operačního systému Apple iOS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 51 s. Vedoucí bakalářské práce Ing. Jan Hajný, Ph.D.

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu této bakalářské práce panu Ing. Janu Hajnému Ph.D. za odborné vedení, cenné připomínky, konzultace, trpělivost a návrhy k práci.

Vysoké Učení Technické, Brno . . . .

(podpis autora)

PROHLÁŠENÍ

Prohlašuji, že svoji bakalářskou práci na téma „Zabezpečení operačního systému Apple iOS“ jsem vypracoval samostatně, pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujícího autorského zákona č. 121/2000 Sb.,včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne . . . . . .

. . . . . .
(podpis autora)

OBSAH

Úvod ... 10
1 Úvod do systému iOS ... 11
2 Architektura systému ... 12
2.1 Bezpečné načtení řetězce při startu ... 12
2.2 Softwarová perzonalizace systému ... 13
2.3 Kódový podpis aplikací ... 14
2.4 Vývoj aplikací pro iOS ... 14
2.5 Zabezpečení běžících procesů ... 15
3 Šifrování a ochrana dat ... 17
3.1 Bezpečnostní prvky hardwaru ... 17
3.2 Zabezpečení pomocí šifrování RSA ... 18
3.3 Ochrana souborových dat ... 19
3.3.1 Přehled architektury ... 19
3.4 Hesla ... 20
3.5 Třídy ... 21
3.6 Ochrana dat v řetězci klíčů ... 22
3.7 Keybags ... 23
4 Zabezpečení sítě ... 24
4.1 Wi-Fi, Bluetooth ... 25
5 Přístup k zařízení ... 26
5.1 Konfigurační profily ... 26
Praktická část ... 28
6 Vývoj aplikací v programu Xcode - úvod ... 28
6.1 Práce v programu Xcode ... 29
6.2 Realizace aplikace ... 34
6.2.1 Uživatelské prostředí ... 34
6.2.2 Knihovna GMP ... 37
6.2.3 Operace ... 40
6.3 Ovládání aplikace ... 44
6.4 Test rychlosti operací ... 44
Závěr ... 48
Literatura ... 49
Seznam příloh ... 50

Seznam obrázků

Obr. 1 iPhone 5

Obr. 2 iPad 2

Obr. 3 iPod Touch

Obr. 4 Connect to iTunes

Obr. 5 OTA Aktualizace iOS

Obr. 6 OTA Aktualizace iOS info

Obr. 7 D-U-N-S číslo organizace

Obr. 8 iCloud úložiště ve webovém rozhraní

Obr. 9 nastavení iCloud na zařízení

Obr. 10 Chip A6 – červeně označené GID

Obr. 11 Nastavení vymazaní zařízení po 10ti neuspěšných pokusech zadání kódu

Obr. 12 Nastavení vymazaní zařízení po 10ti neúspěšných pokusech zadání kódu pomocí aplikace MDM

Obr. 13 Nastavení připojení k VPN

Obr. 14 Ukázka práce aplikace iMessage na iPadu

Obr. 15 Nastavení jednoduchého kódového zámku

Obr. 16 Dlouhé heslo s využitím kombinace abecedy a čísel

Obr. 17 Založení nového projektu

Obr. 18 Rozložení pracovního okna

Obr. 19 Informace o vytvářeném projektu

Obr. 20 Základní nastavení vytvářené aplikace

Obr. 21 Základní zdrojový kód programu

Obr. 22 Uspěšný Build aplikace

Obr. 23 Simulace zkompilované aplikace na virtualním iPhone

Obr. 24 Ikona aplikace

Obr. 25 Záložka Single Test

Obr. 26 Záložka Measurement

Obr. 27 Zobrazení v řádku

Obr. 28 DetailView

Obr. 29 Výpis výsledku časového testu

Obr. 30 Výsledky měření

Obr. 31 Generovaná čísla a provedené operace
Obr. 32 Graf 10ti opakování generování náhodných čísel
Obr. 33 Graf 10ti opakování pro modulární násobení a*b mod c
Obr. 34 Graf 10ti opakování modulárního mocnění a^b mod c
Obr. 35 Graf 10ti opakování odečítání
Obr. 36 View
Obr. 37 Stubs (měření času)

ÚVOD

Operační systém iOS je systém navržený výhradně pro přenosné zařízení od firmy Apple. V roce 2007 byl původně určen jen pro mobilní telefony iPhone, později se však začal používat i na dalších mobilních zařízeních této firmy, jako jsou přenosné tablety iPad, iPad mini, hudební přehrávače iPod Touch všech generací a nově i na Apple TV. Tento systém je založen na stejném jádře a technologiích, jako operační systém pro osobní počítače a Macbooky od Applu – Mac OS X. Jedná se o systém UNIXového typu a jelikož je určen pro přenosná zařízení, neobsahuje veškeré funkce, které nabízí OS X. Naopak však přidává jiné možnosti, jako například podporu dotykového ovládání. Systém se větví do čtyř základních vrstev, díky kterým je zaručena jeho správná funkčnost, stabilita a poskytují vývojářům potřebné API a frameworky nezbytné k vývoji aplikací.

Apple vytváří platformu iOS se zabezpečením obsaženém v jeho jádru. Pro každého koncového uživatele, jak klasického uživatele, tak pro využití ve firmách, je jedním z nejdůležitějších faktorů při výběru zařízení jeho bezpečnost. Nejčastěji se jedná bezpečnost uložených osobních dat, jako jsou fotografie, e-maily, bankovní spojení, adresy, přístupy k různým účtům a mnoho dalších citlivých informací o uživateli. Zařízení iOS jsou navrženy tak, aby udržely nejvyšší ochranu bez kompromisu na uživatelské schopnosti, proto je mnoho klíčových funkcí, například šifrování, nenastavitelné, aby je nezkušený uživatel nemohl omylem vypnout. Tato zařízení poskytují přísné bezpečnostní technologie a funkce. Jsou zkonstruována tak, aby ochrana dat byla co nejspolehlivější, a přesto se zařízení snadno ovládala.

Zařízení mají implementovány nižší vrstvy zabezpečení, tím jsou ochráněny proti malwaru a virum, ve chvíli, kdy vyšší vrstva povolí přístup k osobním informacím a datům. Tyto funkce pomáhaj rozpoznání neautorizovaného přístupu a zmaření případného útoku.

V této práci se zaměřím na to, jak Apple pojímá zabezpečení svého operačního systému a jaké technologie využívá k tomu, aby bylo možné, bez jakýchkoli obav, zařízení neustále používat, jak při spouštění aplikací tří stran, synchronizování, či při přístupu k internetu různými metodami. V další části práce je vysvětlen princip kryptografie a ochrana dat pomocí tříd a klíčů. Další oblastí, kterou se práce zabývá, je ochrana dat uživatele, při ztrátě nebo odcizení zařízení, například vzdálené vymazání nebo vyhledání telefonu pomocí gps modulu. Poslední částí semestrální práce je popis vývojářského balíčku Xcode a kompilace aplikace pro mobilní zařízení.

1 ÚVOD DO SYSTÉMU IOS

iOS byl původně nazýván iPhone OS, ale v roce 2010 byl přejmenován na iOS z důvodů vyvíjející se podpory pro další Apple zařízení. V současné době je k dispozici verze systému 6.x.x, uvolněna společností po uvedení nového iPhone 5, v říjnu 2012. Poslední verze iOS od verze 5 zavádí řadu nových funkcí mobilního operačního systému. Nejvíce pozoruhodný je vestavěný klient po odesílání zpráv (iMessage), dále vestavěná podpora iCloud, tedy osobní Apple cloud úložiště a také aplikace pokročilého rozpoznávání hlasu Apple Siri. Starší zařízení jako je původní iPhone 2G a iPod touch již nejsou podporovány v systémech iOS 4, iOS 5 a iOS 6. iPhone 3G a iPod Touch druhé generace nelze aktualizovat od verze iOS 4.2.1., ve které již nepodporovali některé funkce, například multitasking – více spuštěných aplikací pracujících na pozadí zároveň, z důvodů nedostatečného výkonu těchto zařízení.

Největšími konkurenty Apple iOS v mobilním odvětví operačních systémů jsou Windows Phone 7, Google Android OS, a BlackBerry OS. Apple iOS má velký náskok v oddělení aplikací, jeho internetový obchod App Store již obsahuje téměř 1 000 000 aplikací třetích stran a může se pochlubit více než 50 miliardami stažených aplikací během 5 let od spustění App Store. Dalším významným krokem vpřed bylo spuštění multimedialního obchodu iTunes, kde si každý uživatel muže stáhnout požadovanou hudbu, filmy a podcasty během chvíle. Apple se také muže chlubit aplikací zvanou iBook Store, neboli elektronickou knihovnou, ve které si uživatelé mohou vybrat z 1.5 milionu titulů a ihned stáhnout do svého iOS zařízení.

Obr. 1 iPhone 5

Obr. 2 iPad

Obr. 3 iPod Touch

2 ARCHITEKTURA SYSTÉMU

Blízké spojení softwaru a hardwaru, umožňuje zařízením, běžícím na systému iOS, ověřování činností napříč všemi vrstvami. Od počátečního bootu až k aplikacím třetích stran je každý krok důkladně prověřen, zda jsou prováděné činnosti vždy důvěryhodné a používají řádná pravidla zabezpečení. Spuštěný systém využívá integrovanou bezpečnostní architekturu, která spočívá na spolehlivosti XNU, tedy jádru vytvořené společností NeXT a nyní vyvíjeném společností Apple a využívaném na operačních systémech Mac OS X a iOS. XNU má za úkol zařídit správnou funkci bezpečnostních prvků.

2.1 Bezpečné načtení řetězce při startu

Při startu systému obsahuje každý krok startovacího procesu části, které jsou kryptograficky podepsány Applem pro zajištění integrity a všechny tyto části musí projít tak zvaným řetězcem důvěry. Tento řetězec je spuštěn po zapnutí zařízení a postupně, jeden po druhém, ověřuje kódy všech důležitých souborů pro spuštění celého systému. Pokud jeden z kódů nelze ověřit, spouštění zařízení se zastaví. Do této skupiny patří : bootloaders, kernel, kernel extensions, a základní firmware. Aplikační procesor ihned po zapnutí iOS zařízení převede kód z Read-only Memory, ROM - paměť jen pro čtení. Tento kód byl uložen do pamětového čipu již při výrobě, ten je neměnný a tedy velmi důvěryhodný. Obsahuje : Apple Root CA public key, který je používán k ověřování low-level bootloaderů (LLB) zda mají podpis od Applu, ještě předtím, než jim dovolí spuštění. První krok je tedy založen na ověření podpisů. Po dokončení úlohy LLB se spustí další krok startu systému, ten obsahuje iBoot, který se znovu ověří a poté je spuštěn iOS Kernel. Tedy jádro operačního systému. Bezpečné načtení řetězců při startu, zaručí nejnižší úrovni softwaru, že s ním nebylo manipulovano a také to, aby nebylo možné zprovoznit iOS na jiném zařízení, než ověřené právě Applem. Když v jakémkoliv kroku bootovacího procesu nastane chyba a není možné ověření podpisů, proces se zastaví a na obrazovce zařízení se nám zobrazí nápis “ connect to iTunes “ – žádost o připojení zařízení k oficiálnímu programu iTunes, pro správu dat a aplikací od firmy Apple. Tento mód, žádost, se nazývá Recovery mode. Pokud není Boot ROM schopný získat ověření LLB, vstoupí zařízení do DFU “ Device firmware update “ – V tomto případě je nutné připojit zařízení USB kabelem k počítači a vrátit ho do továrního nastavení.

Obr. 4 Connect to iTunes

2.2 Softwarová perzonalizace systému

Firma Apple průběžně vydává aktualizace firmwaru, zaměřené zejména na bezpečnost systému a opravu drobných chyb v uživatelském prostředí. Tyto aktualizace jsou vydávány zároveň pro všechna zařízení běžící na iOS, uživatel obdrží upozornění na update přímo na zařízení, nebo samozřejmě po připojení zařízení k iTunes. Aktualizace iOS systému mohou být instalovány do zařízení přes oficiální software iTunes, nebo nově pomocí OTA (over the air – neboli bezdrátově). Tato technologie byla zavedena v iOS až od verze 5.0. Zařízení pomocí OTA podporují rychlé přijetí nejnovějších oprav zabezpečení, takovým způsobem, aby nebylo nutné stahovat celou kopii nejnovějšího OS, ale pouze části, které byli nějakým způsobem změněny. Při aktualizaci přes iTunes je celá kopie systému stažena do počítače a následně nainstalována do zařízení. Během instalace nebo updatu softwaru se zařízení připojí k autorizačnímu serveru Applu a žádá ověření každé části datového instalačního rámce. Mezi části obsažené v rámcích patří například LLB, iBoot, kernel a samotný iOS image.

Server zkontroluje a autorizuje všechny rámce pro verzi, pro kterou je instalace updatu povolena, aby se tak vyvarovalo downgradu zařízení, či nainstalování systému z jiných zdrojů, než oficiálních. Po ověření je do instalačních rámců vloženo unikátní ID zařízení ECID (Electronic Chip ID jiné pro každé zařízení). Po tomto kroku je kompletní sada dat podepsána serverem a následně odeslána do zařízení jako část instalace a může začít proces aktualizace. Přidáním ECID kódu se autorizace instalačního rámce přizpůsobí jen pro požadované zařízení. Autorizací a podpisem rámců dat, tedy server zařídí oficiální aktualizaci přímo od Applu. Zda podpis a kombinace rámců s ECID opravdu pochází od Applu, si zařízení samo ověří při bootování a průběhu již zmíněného řetězce důvěry. Tyto kroky zajistí, že autorizace je unikátní pro jedno určité zařízení a také zabrání staré verzi iOS k nainstalování, nebo zkopírování na zařízení jiné. Jedinečné kódy v tomto procesu brání útočníkům uložit odpovědi odesílané servery a použít je v budoucnu pro downgrade, tedy nainstalování starší verze iOS, než je aktuálně na zařízení nainstalovaná.

Na ochranu proti downgrade využívá iOS proces, kterému se říká softwarová perzonalizace systému. Tento proces je v systému obsažen kvůli nedostatku zabezpečení obsažených ve starších verzích, které již mohly být útočníky prolomeny. Jestliže by byl downgrade možný, potencionální utočník by tak získal výhodnou pozici k nainstalování starší verze systému a mohl by využít jeho zranitelnosti v místech, které jsou v nové verzi již dostatečně zabezpečeny.

Obr. 5 OTA Aktualizace iOS

Obr. 6 OTA aktualizace iOS info

2.3 Kódový podpis aplikací

Po spuštění iOS je jádrem kontrolován každý proces a všechny aplikace nainstalované uživatelem. K tomu, aby všechny aplikace pocházely ze známých, prověřených zdrojů a nebylo s nimi jakkoliv nepatřičně manipulováno, vyžaduje iOS spustitelný kód podepsaný Apple certifikátem. Aplikace dodávané s zařízením, jako Safari a mail jsou do systému implementovány a certifikátem podepsány již od výrobce. Samozřejmě aplikace třetích stran musí být tímto certifikátem podepsány také. Povinný kódový podpis rozšiřuje pojem řetězce důvěry od systémových částí až po aplikace a brání tak načítání nepodepsaných či neschválených aplikací třetích stran. Stejně tak brání i přístupu kódů, které dokáží samy sebe modifikovat.

2.4 Vývoj aplikací pro iOS

V případě, že jste vývojář pro Apple a chcete nainstalovat na zařízení vámi vytvořenou aplikaci, musíte být zaregistrováni do vývojového programu společnosti Apple. Každý vývojář musí vyplnit svojí identitu, ať už se jedná o jedince, či o firmu. Před obdržením certifikátu je každý žadatel ověřen a poté muže začít certifikát využívat. Díky němu mají vývojáři povoleno podepisovat vlastní aplikace a odesílat je, k dispozici pro ostatní uživatele, na App Store – oficiální internetový obchod aplikací pro Apple zařízení. Všechny aplikace, již téměř 1 milion, které App Store obsahuje, byly odeslány známým

uživatelem, či organizací. Tento způsob ověřování slouží jako odrazující příklad pro vytváření škodlivých aplikací.

Aplikací přímo od Applu, je minimum, vše nechávají na ostatních vývojářích a díky tomu mají velké zisky z aplikací odeslaných na App Store. Za každou placenou aplikaci, kterou vývojář na tento internetový obchod vloží, si Apple bere provizi 30% z ceny aplikace. V některých zemích, převážně evropských, jsou provize Applu 40%. I po odeslání aplikace z prověřeného zdroje, ověří Apple její správnou funkčnost popisovanou vývojářem a také zda neobsahuje skryté bugy, neboli programátorské chyby, které by mohly způsobit bezpečnostní, nebo jiné problémy. Celý tento proces zaručí zákazníkům důvěru v opravdovou kvalitu zakoupených aplikací.

Podniky mají výhodu v tom, že mohou aplikace naprogramované svými pracovníky používat i jejich ostatní zaměstnanci, bez potřeby oficiální cesty pomocí iTunes či App Store. Organizace i podniky využívají iOS developer Enterprise Program (iDEP) s D-U-N-S čísly –(Data Universal Numbering System) je unikátní identifikační devítimístný číselný znak, díky němuž lze najít jakoukoli firmu kdekoliv na světě a rozpoznat její vlastnické vazby. Ty schvaluje Apple všem žadatelům po ověření jejich způsobilosti. Když se organizace stane členem iDEP, získá profil, který umožňuje vytvořené aplikace spouštět na autorizovaných zařízeních, bez potřeby App Store. Díky tomu mohou uživatelé z této organizace spouštět aplikace na svých zařízeních. Na rozdíl od ostatních mobilních platforem (například android)nepovoluje iOS všem uživatelům nainstalovat potenciálně škodlivé nebo nepodepsané aplikace přímo z webových stránek nebo spustit kód, který není dostatečně ověřen. Během spouštění ověřuje kodový podpis, zda je spustitelná paměť stránek napsána tak, jak je načtena a tím pádem zajistí, aby aplikace nebyly modifikovány během instalace či posledního updatu.

Obr. 7 D-U-N-S číslo organizace

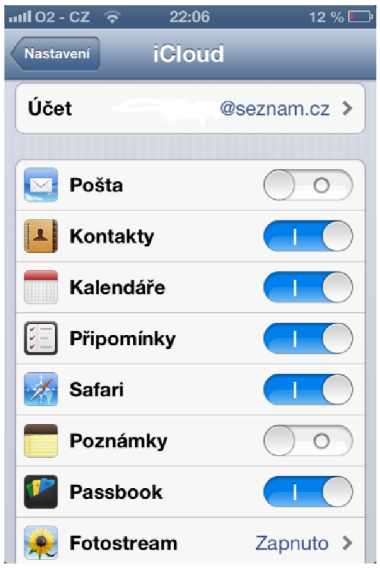
2.5 Zabezpečení běžících procesů

Všechny aplikace pocházející z jiných než oficiálních zdrojů jsou takzvaně "sandboxed", nebo-li jsou umístěny v určité části paměti a zabezpečeny tak, aby nemohly dělat změny v zařízení či neměly bez povolení přístup ke zbytku systému, uživatelským datům a k souborům vytvořených jinou aplikací. Výjimky jsou pouze pro aplikace, kterým uživatel dovolí přístup například k obrázkům (různé foto/video app), ke kontaktům (viber, facebook). I v tomto případě jsou však aplikace stále omezené a nemohou provádět žádné změny, před kterými by uživatele předem upozornily. Sandboxing tedy ochraňuje ostatní aplikace proti modifikaci informací uložených aplikacemi. Každá aplikace má všechna svá data uložena v domácí složce, vytvořené během instalace. Jedná se o stejný princip instalace jako na operačním systému Windows, s tím rozdílem, že přístroj sám zvolí, kam požadovanou aplikaci nainstalovat.

Pokud aplikace žádá přístup k jiným než vlastním datům, je to možné pomocí APIs nebo služeb poskytovaných systémem iOS – samozřejmě tato funkce musí být implementována v kódu již během programování. I tak, jak už bylo uvedeno výše, se však musí aplikace znovu zeptat uživatele, zda s vyžadovaným přístupem souhlasí. Ovšem toto neplatí jen pro aplikace ze zdrojů třetích stran, ale zajisté jsou před všemi nainstalovanými aplikacemi chráněny i systémové soubory. Celá tato část iOS je nastavena jen pro čtení. V systémovém softwaru však některé nezbytné nástroje, jako například vzdálené přihlášení, nejsou implementovány a APIs těmto aplikacím nedovolí stupňovat jejich vlastní privilegia k modifikování ostatních aplikací, ba dokonce samotného iOS.

Přístup aplikací třetích stran k uživatelským informacím a funkcím jako je iCloud (bezdrátová záloha dat uživatele, na servery applu, do vlastního počítače, či jiného iOS zařízení) je kontrolován pomocí stanovených nároků. Tyto nároky jsou zaměřeny na spárování klíč/hodnota, to je popsáno v kódu aplikace a povoluje tím autentifikaci mimo běh ostatních faktorů jako je unix user ID. Protože tyto nároky jsou digitálně podepsané, nemohou být změněny. Jsou využívány především systémovými aplikacemi k provedení zvláštních privilegovaných operací, které by jinak vyžadovaly spustit proces jako root, neboli s vysokou prioritou. Kromě toho mohou aplikace provádět běh na pozadí pomocí funkce system provided APIs, také zaváděnou již při psaní aplikace. Tato funkce umožnuje aplikacím pokračovat v chodu bez snížení výkonu zařízení nebo dramatického dopadu na životnost baterie. Aplikace nemohou navzájem přímo sdílet data, sdílení je implementováno pouze pomocí schémat URL (řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací), nebo prostřednictvím sdíleného přístupu do skupiny klíčů uložených, ve speciálním úložišti keychain.

Obr. 8 iCloud úložiště ve webovém rozhraní



Obr.9 Nastavení iCloud na zařízení

3 ŠIFROVÁNÍ A OCHRANA DAT

Podepisování kódu, zabezpečení spuštěných procesů a bezpečné načtení řetězců, nám zajistí spouštění jen důvěryhodných aplikací a kódů běžících na zařízení. Systém iOS obsahuje přidané prvky pro ochranu dat uživatele, a to i v případě, že ostatní části systémové infrastruktury budou v ohrožení (například na zařízení, které používá neautorizované úpravy – toho docílíme Jailbreakem zařízení. Podstata Jailbreaku je otevření jinak poměrně uzavřeného iOS, a poté do zařízení instalovat zdarma i placené aplikace z jiných zdrojů, než je App Store. Aplikace z těchto zdrojů nejsou samozřejmě podepsané Applem, tím pádem nedůvěryhodné a nebezpečné). Jak samotná systémová architektura, tak i šifrování a ochrana dat využívá vrstev integrovaných v hardwarových a softwarových technologiích.

3.1 Bezpečnostní prvky hardwaru

Na mobilních zařízeních hraje rychlost a úspora energie hlavní roli. Šifrovací operace jsou komplexní a mohou tak snižovat životnost baterie v případě, že nejsou správně navrženy. Všechna iOS zařízení využívají šifrování AES s pevně daným blokem stejného 256bitového klíče pro šifrování i dešifrování, zabudovaným do DMA cesty mezi flash úložištěm a hlavním systémovou pamětí. Technologie DMA využívá přímého propojení paměti a úložiště bez potřeby průchodu dat přes procesor, tím pádem je tato funkce rychlejší a díky tomu se stává šifrování vysoce efektivní. Společně s AES enginem je v hardwaru implementována kryptografická hashovací funkce SHA-1, snižující zatížení během šifrování. Unikátní ID (UID) zařízení a skupinové ID (GID) mají klíč AES 256-bit vložený přímo do aplikačního procesoru již při výrobě. Žádný software ani firmware z něj nemůže číst přímo. Mohou být zobrazeny jen výsledky po použití šifrovacích nebo dešifrovacích operací.

UID je unikátní pro každé zařízení a toto ID není Applem nebo některým z jeho dodavatelů zaznamenáno. GID je na rozdíl od unikátního ID stejné pro všechny procesory ve své třídě, tedy například pro chip procesorů A5 (iPhone4) je GID jiné, než pro chip z třídy A6(iPhone5) a využívá se jako přídavná úroveň zabezpečení při dodávání systémového softwaru během instalace a obnovení zařízení. Vypálení tohoto GID do křemíku, zabraňuje Apple jeho obejití či manipulaci a zaručuje, že může být přístupný pouze AES enginem. Pomocí UID se uživatelská data zašifrovaně svážou se zařízením. Příkladem toho je klíčová hierarchie, ta v sobě má implementované UID, pokud by byla pamět fyzicky přemístěna z jednoho zařízení do jiného, učinilo by to data na této paměti nepřístupná. V zařízení nesouvisí UID s žádným jiným identifikátorem. Na rozdíl od UID a GID jsou ostatní kryptografické klíče vytvořené systémovým generátorem náhodných čísel (RNG) využívající algoritmus založený na řebříčkové struktuře[1]. Bezpečné mazání dočasně uložených klíčů je stejně důležité jako jejich generování.

Zejména náročné je toto provádět na zařízeních s flash pamětí, kde opotřebení této paměti hraje klíčovou roli, a proto je nutné vytvářet více kopií dat, které je třeba poté vymazat. Chceme-li tomuto problému předejít, nabízí iOS funkci věnovanou bezpečnému vymazání dat ze specialního úložiště klíčů Effaceable storage, ve kterém se klíče uchovávají jen na potřebnou dobu. Tato funkce přistupuje k základní technologii ukládání přímo a řeší či vymaže malý počet bloků na velmi nízké úrovni.

Obr. 10 Chip A6 – červeně označené GID

3.2 Zabezpečení pomocí šifrování RSA

RSA je jedním z algoritmů používaných v tzv. infrastruktuře veřejných klíčů. RSA šifruje zprávy v blocích (blok za blokem).

Každodenně je nejčastěji využíván jen na šifrování podpisů a klíčů. To z důvodu, že jeho šífrování a následné dešifrování je pomalé a vyžaduje mnoho času. Obecně samozřejmě platí, že čím rychlejší hardware máme, tím rychleji šifrování probíhá. V mobilních telefonech nepotřebujeme pomocí RSA šifrovat filmy, muziku, či další data rozsáhlejší velikosti a proto se toto velmi učinné šifrování využívá jen na informace, které potřebujeme mít zabezpečené proti neoprávněnému zneužití a aby jsme při šifrování nebyli nijak omezeni zpomalením našeho mobilního zařízení.

Zabezpečení RSA je založené na problému faktorizace velkých čísel. Například přenos zprávy mezi dvěma uživateli – přijímací strana vygeneruje pár klíčů z nichž je jeden klíč věřejný a ten je je sdělen vysílacím kanálem straně odesílající zprávu. Odesílatel tímto klíčem zašifruje zprávu. Poté přichází na řadu druhý klíčí – privátní, který přijímací strana nikomu nesděluje, ale po přijetí zašifrované zprávy ji tiímto klíčem děšifruje. Aby to bylo možné, je nutné, aby oba klíče byli určitým způsobem propojeny. K bezpečnosti RSA šifrování musí být těžké na základě znalosti veřejného klíče a zašifrované zprávy zjistit klíč privátní. Toho RSA dosahuje tím, že spolehá na neschůdnost faktorizace velkých celých čísel.

Doba potřebná k nalezení hodnoty faktoru k proboření klíče je velmi dlouhá. Mnoho odborníků na šifrování usuzuje, že lineární a diferencialní útok na RSA je velmi komplikovaný. Proveditelným útokem je útok matematický, ale čas potřebný k použití

tohoto způsobů je dosti dlouhý a tím pádem i nepraktický. Možná v budoucnu až budou počítače o mnoho rychlejší, bude tento šifrovací algorytmus uznán za málo bezpečný a tudíž nepoužitelný.

Ovšem nesmíme zapomenout na to, že jsme schopni jednoduše změnit velikost klíče a tím zvýšit účinnost zabezpečení proti útoku. Nyní je nejvíce využívána velikost klíče 1024-bitů.

3.3 Ochrana souborových dat

K hardwarovým funkcím šifrování, zabudovaných do iOS zařízení navíc patří technologie Data protection – ochrana dat, která pomáhá další k ochraně dat uložených na flash paměti. Tato technologie je navržena pro mobilní zařízení s ohledem na to, že mohou být neustále zapnuty, připojeny k internetu, kdykoliv přijímat hovory, sms zprávy a e-maily. Data protection povoluje zařízení odpovídat na události jako jsou příchozí hovory, bez potřeby dešifrování citlivých dat a stahování nových informací, i když je zařízení uzamčeno. Toto chování je kontrolováno pomocí základního přiřazení třídy každému souboru, jak je popsáno níže. Data v každé třídě jsou chráněna v závislosti na tom, která data zrovna potřebují přístup. Přístup k datům je možný jen pokud byla určitá třída odemčena. Data protection je implementována při vytváření a spravování hierearchie klíčů a postavena na technologii hardwarového šifrování popsaného výše.

3.3.1 Přehled architektury

Vždy, když je na oddílu dat vytvořen soubor, vytvoří Data Protection nový 256-bit klíč a předá jej hardwarovému šifrovacímu AES enginu, ten využívá klíč k zašifrování souboru uloženého na flash paměti, tomuto klíčí se říka Per-file. Per-file klíč je zabalen jedním z několika tříd klíčů, v závislosti na okolnostech, za kterých by soubor měl být přístupný. Nově zabalené klíče se ukládají ve speciální složce, pro to určené, s názvem – file´s metadata. Po otevření uloženého souboru jsou jeho metadata dešifrovány “File system klíčem“ ,ten odhalí zabalený per-file klíč a také třídu, která ho ochraňuje.

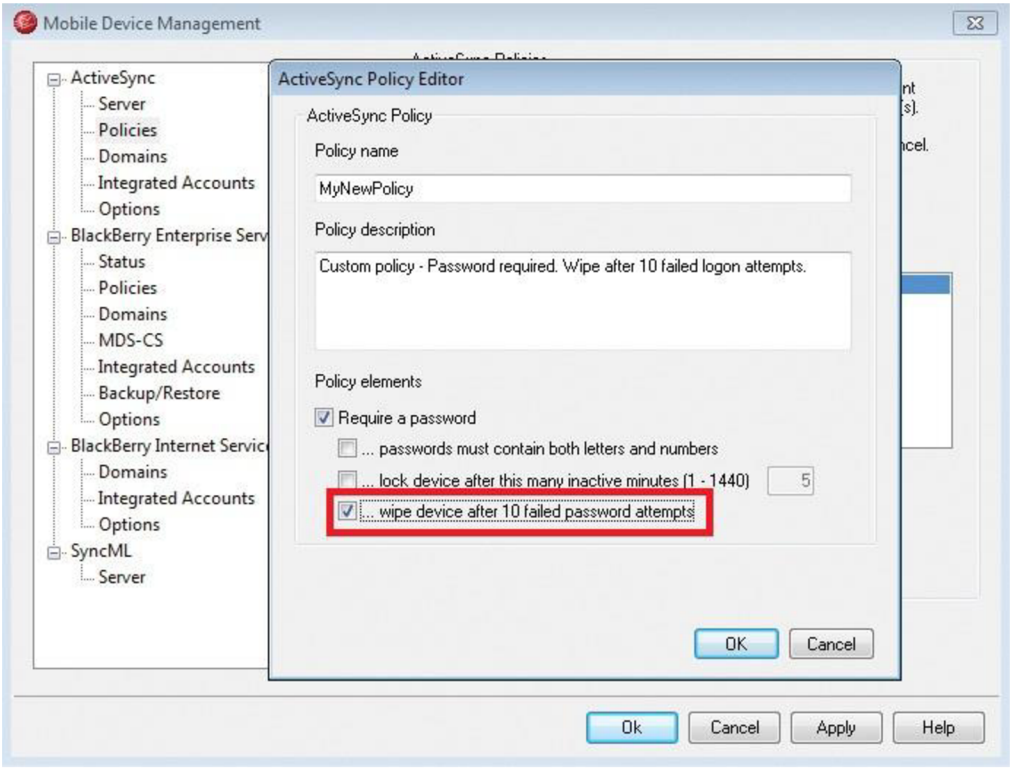
Poté je per-file klíč rozbalen pomocí klíče třídy, kterou získal v předchozím kroku, poskytnut AES enginu, který dešifruje soubor a ten poté muže být načten z flash paměti zařízení. Metadata všech souborů v tomto souborovém systému jsou šifrovány náhodným klíčem, který je vytvořen po nainstalování iOS, nebo pokud je zařízení vzdáleně vymazáno uživatelem. Klíč souborového systému je uložen v Effaceable storage. Od chvíle uložení souboru do zařízení není tento klíč použit, k udržení důvěryhodnosti dat. Místo toho je navržen k rychlému vymazání na žádost uživatele (použitím možnosti “Vymazat všechen obsah a nastavení“ , nebo vysláním žádosti o vzdálené vymazání, například přes iCloud). Po vymazání se stanou všechny soubory kryptograficky nepřístupné, a tím pádem zničené. Obsah souboru je zašifrován per-file klíčem, ten je zabalen klíčem třídy a uložen na flash paměti ve složce metadata,

zašifrovaná systémovým souborovým klíčem – kombinace uživatelského hesla a UID zařízení. Stejně tak jako klíč třídy, který je chráněn hardwarovým UID a v některých případech i heslem uživatele. Tato hierearchie poskytuje především potřebný výkon a flexibilitu. Například změnou souborové třídy je vyžadováno překombinování per-file klíče a změna hesla pro právě překombinovanou třídu.

3.4 Hesla

Během nastavování hesla zařízení spustí uživatel automaticky ochranu dat (data protection). Systém iOS podporuje čtyřčíselné a také jakkoliv dlouhé abecední hesla. Kromě odemykání zařízení zajišťují hesla entropii pro šifrování klíčů, které nejsou na zařízení uloženy. V případě, že má utočník cizí zařízení, nedostane přístup k některým chráněným třídám bez hesla. Tyto hesla jsou spleteny s UID. Každý pokus zadání hesla je nastaven tak, aby zabral v nejlepším případě přibližně minimálně 80 milisekund, pokud by se tedy útočníčník snažil náhodně zadat heslo, trvalo by mu to 5,5 let při použití šesti místného hesla v kombinaci s čísly a abecedními znaky, nebo 2,5 roku při použití hesla s devíti číslicemi. K odrazení dalších útoků poskytuje iOS stupňující se časové zpoždění po zadání špatného hesla v módu uzamčené obrazovky. Uživatel si může také zvolit, zda má být jeho zařízení po 10ti nesprávných pokusech automaticky vymazáno.

Obr. 11 Nastavení vymazaní zařízení po 10ti neuspěšných pokusech zadání kódu



Obr. 12 Nastavení vymazaní zařízení po 10ti neúspěšných pokusech zadání kódu pomocí aplikace MDM

3.5 Třídy

Jakmile uživatel iOS zařízení přidá nebo vytvoří nový soubor, je tomuto souboru automaticky přiřazena třída. Každá třída využívá při přístupu k uloženým údajům různé bezpečnostní podmínky.

První třída se nazývá kompletní ochrana. Při použití této ochrany je soubor dostupný jen pokud je zařízení odemčeno. Pokud uživatel zařízení uzamkne, je klíč této třídy vymazán a všechna právě otevřená data jsou nepřístupná až do opětovného zadání hesla a poté vytvořen nový klíč třídy. Tato třída je implementována například v emailové aplikaci nebo prohlížeči obrázků.

Druhou nejčastěji využívanou třídou je ochrana po otevření. Je prakticky stejná jako kompletní ochrana, s tím rozdílem, že pokud je soubor otevřen, je mu dovoleno pokračovat v zápisu i po uzamčení zařízení. Dobrým příkladem je přijímání emailů a jejich zapsání na úložiště při uzamčeném zařízení.

Další ochranou je ochrana po první ověření. Jakmile je zařízení zapnuto a uživatelem zadáno heslo, jsou všechny soubory chráněné touto třídou dostupné až do vypnutí zařízení. Tento princip je podobný jako u klasického stolního počítače. V této třídě nehraje uzamčení přístroje žádnou roli.

Čtvrtou třídou zabezpečenou jen unikátním ID zařízení je třída tzv. bez ochrany. Je využívána jako výchozí třída pro všechny nové soubory přiřazené do “Data protection“. Soubory jsou v případě této ochrany vždy dostupné.

3.6 Ochrana dat v řetězci klíčů

Řetězec klíčů je imlementován jako SQLite databáze uložená v souborovém systému ve třídě "bez ochrany". Položky řetězce klíčů mohou být sdíleny jen mezi aplikacemi od stejného vývojáře. To je řízeno vyžadováním přístupu aplikací třetích stran ke skupinám s prefixem přiděleným přes vývojářský program iOS. Ochrana dat řetězců je realizována pomocí struktury tříd, podobné té, využívané v Data protection. Tyto třídy mají tedy stejné chování jako třídy v Data protection, ale používají jiné klíče a jsou součástí jinak pojmenovaných APIs. Rozdílné názvy tříd jsou uvedeny v tabulce číslo 1.

Tabulka č. 1.

Přístup k datům	File Data Protection	Keychain Data Protection
Po odemčení	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
Po uzamčení	NSFileProtectionCompleteUnlessOpen	Není dostupné
Po prvním odemčení	NSFileProtectionCompleteUntilFirstUserAut-entification	kSecAttrAccessibleAfterFirstUnlock
Vždy	NSFileProtectionNone	kSecAttrAccessibleAlways

Každá třída řetězce klíčů, který je vždy chráněn univerzálním ID. Při kopírování ze zařízení během zálohy má protějšek jen pro zařízení ze kterého je kopírováno, a tím činí řetězce klíčů nepoužitelnými, pokud jsou obnoveny na jiném zařízení.

Apple má pečlivě vyvážené zabezpečení a jeho použitelnost vybíraní tříd zavisí na zjištění typu informace. Například certifikát VPN (virtual private network) musí být vždy dostupný, aby zařízení udrželo spojení (VPN je Virtuální Privátní Síť vznikající propojením několika počítačů připojených k internetu, nejčastěji v rámci firmy. Pokud ovšem uživatel zná přihlašovací údaje, může se do této sítě připojit odkudkoliv). VPN je klasifikován jako "non-migrary", což znamená, že nemůže být přemístěn do jiného zařízení.

Obr. 13 Nastavení připojení k VPN

Pro položky klíčového řetězce kódů obsažené v iOS se uplatnují následující třídy ochran: Tabulka č. 2

Funkce - položka	Přístup
Hesla k Wi-Fi	Po prvním odemčení
E-mail účty	Po prvním odemčení
Exchange účty	Po prvním odemčení
VPN certifikáty	Vždy, non-migrary
VPN hesla	Po prvním odemčení
LDAP,CalDAV,CardDAV	Po prvním odemčení
Záloha iTunes	Po odemčení, non-migrary
Hlasové zprávy	Vždy
Hesla v Safari	Po odemčení
Bluetooth klíče	Vždy, non-migrary
Servisní oznámení	Vždy, non-migrary
iCloud certifikáty a soukromé klíče	Vždy, non-migrary
Klíče k iMessage	Vždy, non-migrary
Certifikáty a soukromé klíče konfig. profilu	Vždy, non-migrary
PIN kód SIM karty	Vždy, non-migrary

3.7 Keybags

Klíče pro řetězce Data protection jsou shromážděny a řízeny v tzv. keybags. System iOS používá následující "keybagy" : systémový, zálohovací, escrow a zálohu iCloud. Systémový keybag je jako jediný uložen v zařízení v uložišti, nazývaném vymazatelné. To má velmi důležitý význam v ochraně dat, neboť klíč v tomto úložišti je vymazán a vytvořen znovu pokaždé, když uživatel změní heslo.

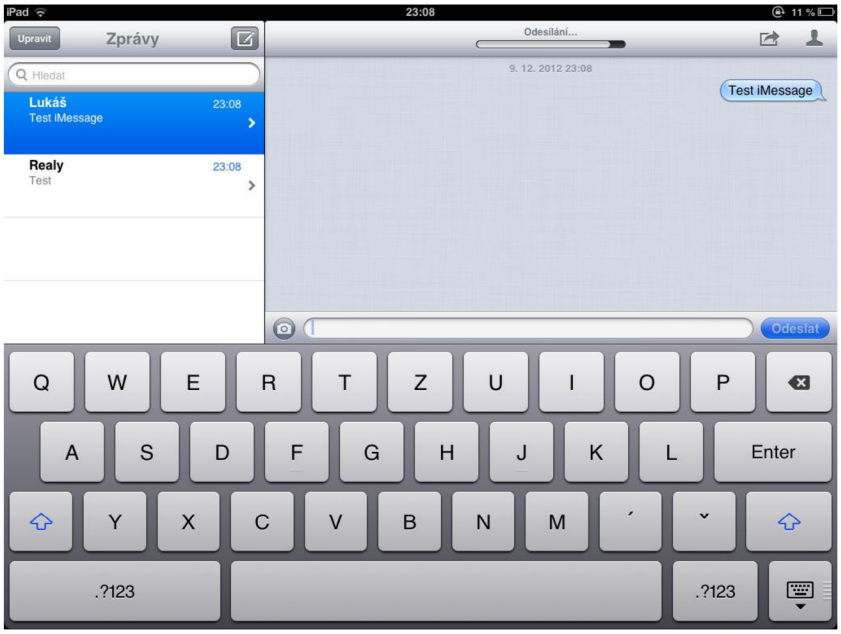
Keybag pro zálohování je narozdíl od systémového, vytvořen po prvním připojení zařízení k iTunes, poté je zašifrován a uložen na pevném disku počítače z kterého je prováděna záloha zařízení. Zajišťuje také, aby záloha jednoho zařízení nemohla být zkopírována do jiného. Escrow keybag také využívá iTunes a Mobile Device Management(MDM) – viz níže. Využívá se pro zálohu pomocí iTunes a synchronizaci zařízení bez potřeby zadání uživatelského hesla. Dále povoluje MDM serveru vzdáleně vymazat heslo uživatele. Tento keybag je také uložen na počítači z kterého je synchronizace prováděna. K rozbalení escrow keybagu je potřeba kódu, který je také vytvořen při prvním připojení k iTunes a nakopírován do zařízení a využívá ochranou třídu zvanou "ochrana po prvním otevření" popsanou výše.

Keybag zálohy iCloud je podobný klasickému zálohovacímu keybagu. Pro všechny třídy ochrany dat, mimo třídy bez ochrany, jsou čtena šifrovaná data ze zařízení a odesílána na iCloud a chráněna klíčem spadajícím do třídy, která byla odesílaným datům přidělena. Tato záloha je prováděna v pozadí bezdrátově přes Wi-Fi a uživatel o jejím průběhu není informován.

4 ZABEZPEČENÍ SÍTĚ

Kromě opatření která Apple využívá k ochraně dat uložených na iOS zařízeních, má ve svém systému implementováno také mnoho síťových bezpečnostních opatření k zabezpečení přijímaných či odesílaných dat z iOS zařízení. Uživatel musí mít přístup k různým sítím odkudkoliv na světě. Je tedy velmi důležité zajistit ochranu všech dat během přenosu. Systém iOS využívá osvědčené technologie a nejnovější standarty k dosažení nejlepšího stupně zabezpečení jak pro přístup přes Wi-Fi, tak pomocí datového síťového připojení, které nám poskytuje operátor. Na ostatních platformách (např. android) je vyžadováno, aby firewall ochraňoval mnoho otevřených komunikačních portů proti napadení. Apple tento problém, tedy omezení útoku, řeší limitováním odposlechu těchto portů a odstraněním nepotřebných síťových nástrojů, které firewall nepotřebují. Těmi jsou telnet, shells nebo web server. Komunikační nástroje (iMessage,FaceTime) vyvíjené Applem a vloženy do systému už od výroby, jsou plně zašifrovány a ověřeny. FaceTime i iMessage jsou aplikace sloužící pro komunikaci mezi Apple zařízeními iPhone a iPad. Pro svou funkci nevyžadují přístup k mobilní síti a veškerá výměna dat probíhá i přes Wi-Fi.

V případě, že není dostupný přístup k datové síti, mohou být odesílané zprávy odeslány jako klasické sms, ovšem jen v případě povolení této funkce uživatelem v nastavení, v opačném případě zpráva není odeslána. Vzhledem k tomu, že komunikace probíhá z iOS na iOS, jsou přenášená data velmi dobře chráněna a tedy neodposlechnutelná.



Obr. 14 Ukázka práce aplikace iMessage na iPadu

4.1 Wi-Fi, Bluetooth

Operační systém iOS samozřejmě podporuje standartní Wi-Fi protokoly se zabezpečením WPA2 (Wi-Fi Protected Access), pro poskytnutí oprávněného přístupu k bezdrátovým sítím. WPA2 využívá 128 bitové AES šifrování, a tím zaručuje uživatelům nejvyšší úroveň ochrany a zaručuje tak, že jejich bezdrátově přenášená data budou i nadále chráněna. Díky podpoře standartu 802.1X mohou být iOS zařízení připojena k síti takřka ve všech prostředích po celém světě. IEEE 802.1X je protokol umožňující jednoduché a efektivní zabezpečení fyzického přístupu do počítačové sítě.

Další bezdrátovou funkcí, kterou nám iOS nabízí, je Bluetooth. Na zařízeních od firmy Apple je realizováno Bluetooth takovým způsobem, aby bylo možné poskytnout její plnou funkci bez potřeby přístupu k osobním datům uživatele (při případném útoku na data uživatele přes bluetooth se útočník k datům nedostane, jelikož bluetooth využívá funkce mimo dátové úložiště). Toho je dosaženo využitím šifrování módu 3 a bezpečnostního módu 4 [2]. Již od počátku využívání Bluetooth v mobilních telefonech bylo možné spárovat jedno zařízení s druhým a poté si navzájem odesílat soubory jakéhokoliv typu. Toto u systému iOS možné není.

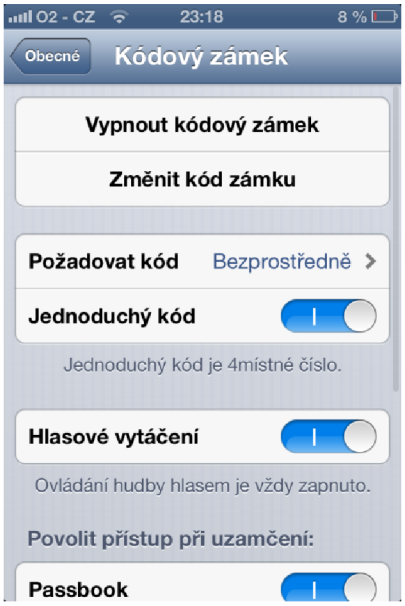
Apple omezil používání Bluetooth jen pro následující funkce:

- Hands-Free (například spárování s autorádiem pro uskutečnění hovorů během jízdy)
- Přístup ke kontaktům (souvislost s Hands-Free)
- Streamování audia (napřiklad do bezdrátových sluchatek)
- Vzdálené ovládání audia a videa (dálkové ovládání TV, Hi-fi)
- Vytvoření osobního Hot-Spotu (sdílené připojení k internetu)
- Podpora připojení klávesnicí či joysticků (nejvyužívanější na iPadu)
- Spojení dvou iOS zařízení pro hraní multiplayer her (hra více hráčů)

5 PŘÍSTUP K ZAŘÍZENÍ

Kromě poskytování kryptografických ochran uvedených výše, jsou hesla chráněna proti neautorizovanému přístupu do zařízení. Rozhraní iOS je nastaveno tak, aby při každém zadání špatného hesla dramaticky zvýšilo dobu potřebnou pro zadání hesla na uzamčené obrazovce. Uživatel si také může vybrat, zda nechá svá data po 10ti nesprávných pokusech vymazat jednoduchým nastavením v nastavení systému nebo pomocí MDM.

S využitím Mobile Device Managment (MDM) mohou IT oddělení organizací, které využívají Apple zařízení iPhone a iPad, bezdrátově nastavovat a updatovat nastavení zařízení. Mohou také sledovat dodržování firemních pravidel. Dokonce i vzdáleně vymazat zařízení, nebo uzamknout jeho ovládaní. Základní uživatelské heslo je nastaveno jako čtyřčíselný PIN, ovšem uživatel si může nastavit delší heslo, nebo heslo s využitím abecedních znaků v kombinaci s čísly. Delší hesla jsou poté samozřejmě težší k uhodnutí nebo k útoku a jsou doporučena nejen pro uživatele, ale hlavně pro firemní užití.



Obr. 15 Nastavení jednoduchého kódového zámku

Obr. 16 Dlouhé heslo s využitím kombinace abecedy a čísel

5.1 Konfigurační profily

Profil konfigurace je obsažen v XML souboru, který poskytuje administrátorovi konfigurační informace o iOS zařízení. Nastavení definované konfiguračním profilem nemůže být uživatelem změněno. Pokud uživatel profil vymaže, je kompletně veškeré nastavení, definované tímto profilem, vymazáno. V tom případě si může administrátor vynutit nastavení pravidel pro přístup. Například konfigurační profil poskytující nastavení emailu, stanoví pravidla pro vytvoření hesla. Uživatelé tedy nebudou mít přístup k mailu, pokud jejich hesla nesplňují požadavky administrátorů. Konfigurační profil iOS obsahuje množství nastavení, které mohou být nastaveny následovně :

- Povolení Siri při uzamčeném zařízení
- Povolení používání YouTube
- Povolení užití obchodu iTunes
- Povolení používání webového prohlížeče Safari
- Povolení automatického vyplňování v Safari
- Povolení JavaScriptu
- Blokování vyskakovacích oken
- Přijímaní cookies
- Povolení zálohy přes iCloud
- Povolení synchronizace dokumentů pomocí iCloud
- Povolení streamování fotek
- Povolení diagnostiky a následné odeslání na Apple server (pro Apple důležité informace aby mohli ve svěm systému například opravovat různé chyby či nedostatky)
- Povolení uživateli přijímat nedůvěryhodné TLS certifikáty

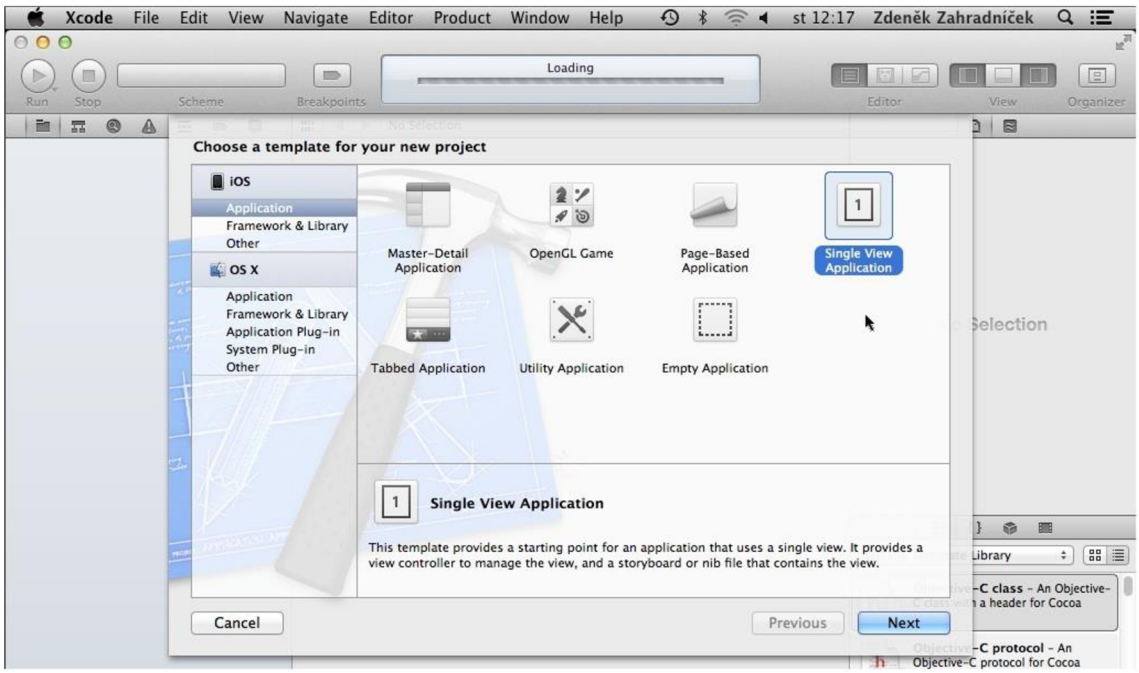
PRAKTICKÁ ČÁST

6 VÝVOJ APLIKACÍ V PROGRAMU XCODE - ÚVOD

Xcode je vývojářský balíček od firmy Apple podporující správu projektů, úpravy kódů, sestavení spouštěcích souborů, správu na úrovni zdrojového kódu, správu repozitářů, ladění výkonu a mnoho dalšího. Jádrem tohoto balíčku je samotná aplikace Xcode, která poskytuje základní prostředí pro vývoj zdrojového kódu. Toto vývojové prostředí poskytuje vývojářům vše potřebné pro vytváření aplikací na všechna zařízení běžící na iOS a Mac OS X, tedy pro MacBooky, iPhony, iPady a iPody Touch všech generací. Program umí analyzovat informace o projektu, identifikovat chyby jak v syntaxi tak v logice a dokonce může pomoci opravovat váš kód. Na webových stránkách Applu je k dispozi ke stažení zdarma, ale jen pro osobní užití. Pokud by uživatel měl zájem vyvíjet aplikace pro iOS nebo Mac OS X, musí být zaregistrován jako vývojář a Applem ověřen jak již bylo popsáno v kapitole 3.4.

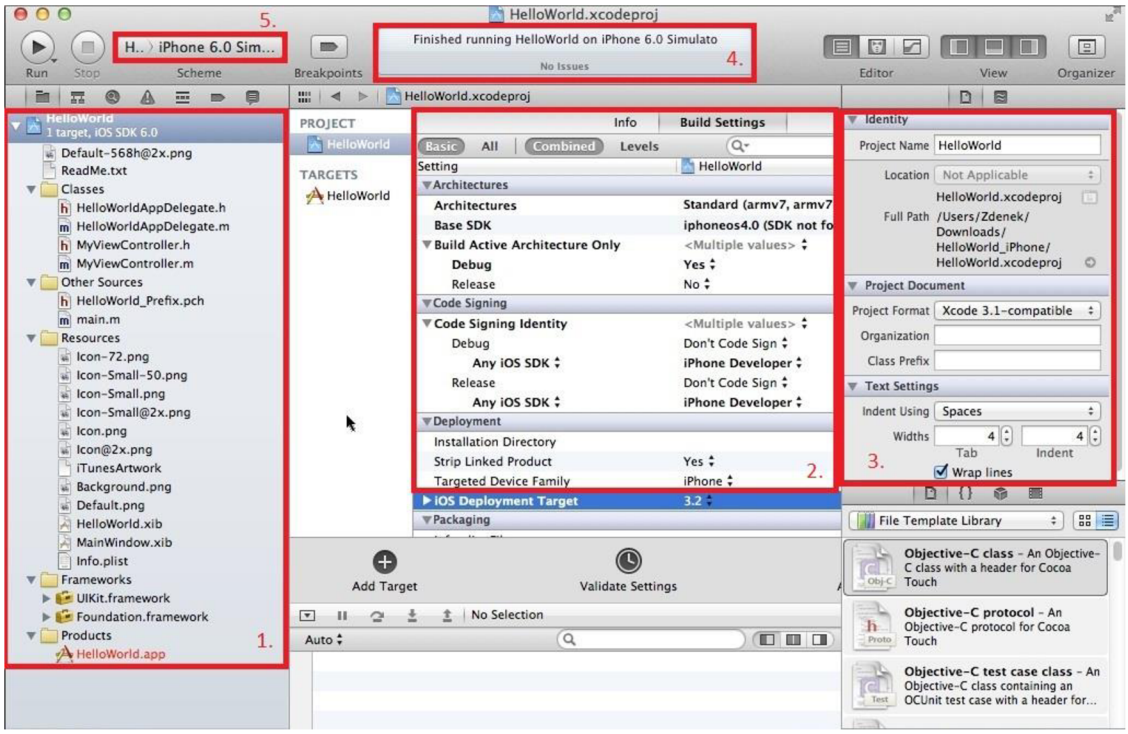
Při sestavování aplikace v Xcode si může vývojář zvolit pro jaké zařízení chce aplikaci vytvořit. Tím se také automaticky nastaví simulátor zvoleného zařízení. Simulátor poskytuje prostředí pro testování aplikace ve stejných podmínkách jako na reálném zařízení a tím zajistí, že se aplikace bude chovat tak, jak se od ní očekává po nainstalování na požadované zařízení. Když je vývojář s chováním své aplikace spokojen, může instruovat Xcode k jejímu sestavení a spustit ji na zařízení připojenému k počítači.

6.1 Práce v programu Xcode



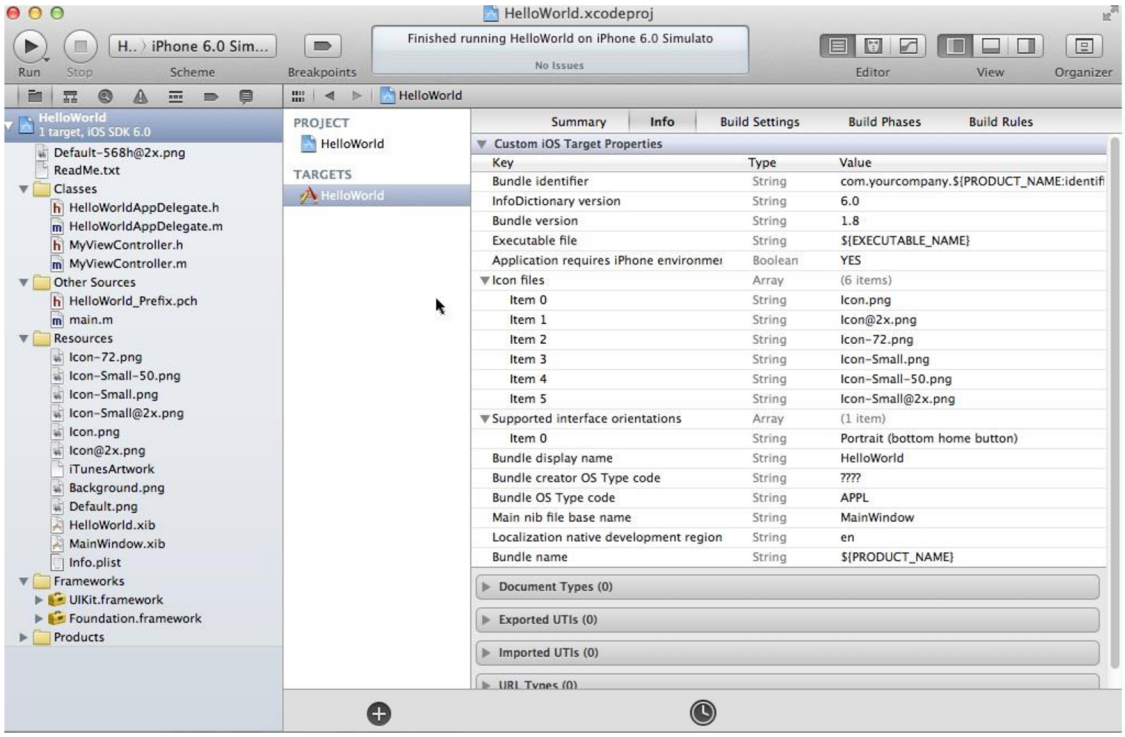
Obr. 17 Založení nového projektu

Po spuštění balíčku Xcode a následného vybrání nového projektu se zobrazí okno, ve kterém si uživatel může zvolit jaký typ aplikace chce vytvořit, viz Obr. 17 Na základě toho si program sám přizpůsobí parametry pro programování zvoleného typu požadované aplikace. Pro výběr je k dispozici Master-Detail Appliciation, OpenGL Game (vytvoří projekt pro programování herních aplikací), Page-based Application (například pro aplikace jako je kalendář. Vytvoří projekt obsahující aplikaci určenou k zobrazení stránek pro každý měsíc v roce, vytvoří tedy 12 stránek), Single view Application (Vytvoření jednoduché aplikace), Tabbed Application (vytvoření aplikace s možností užití více záložek), Utility Application (vytvoření aplikace umožnující uživateli dělat její pomocí drobné úpravy) a Empty Application (vytvoří prázdnou šablonu).



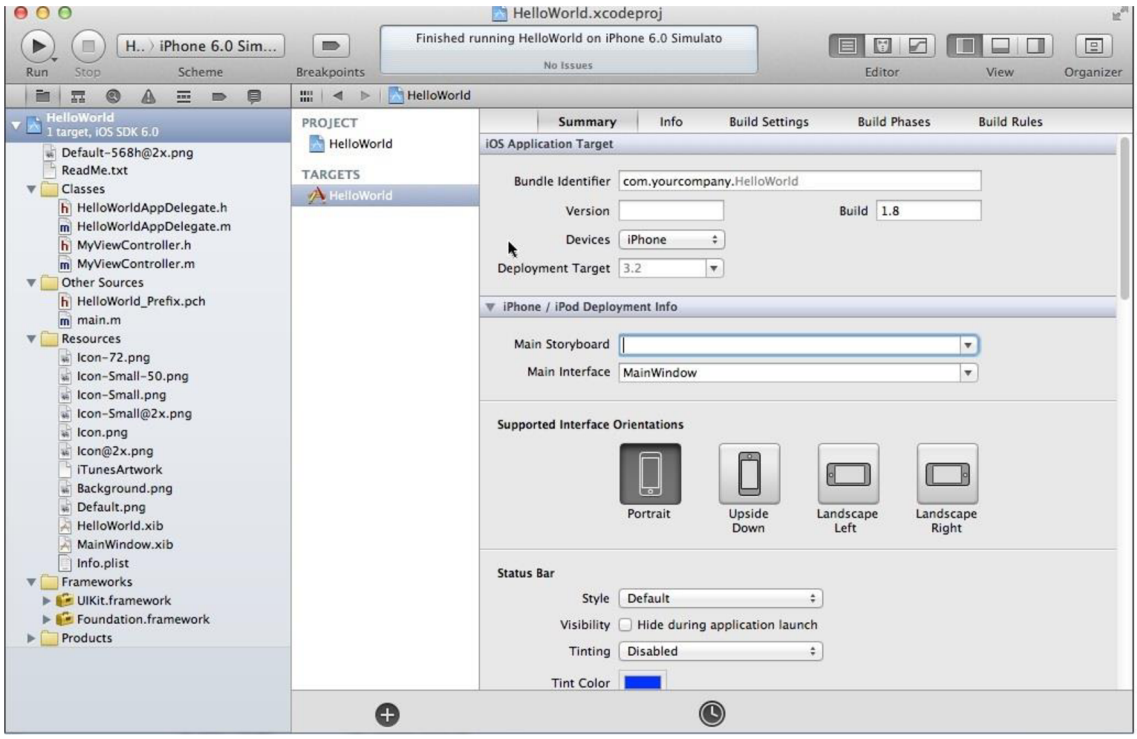
Obr. 18 Rozložení pracovního okna

1) Tato část okna slouží jako navigační. Obsahuje složky se soubory právě vytvářené aplikace a umožnuje uživateli v těchto souborech rychlé hledání a přepínání potřebných oken. Panel je možno skrýt tlačítkem v pravé horní části programu.
2) Uprostřed okna na nachází editační oblast, v té se otevírají soubory obsahující zdrojový kód aplikací, nastavují se zde také například cesty ke knihovnám, dostupnost pro zařízení, verze aplikace a vlastnosti aplikací.
3) Třetí oblast je takzvaná užitečná oblast, zde se nastavuje dokumentace projektu. Tento panel se dá skrýt tlačítkem v pravé horní části programu.
4) Informační display, informuje například o průběhu buildu, či varuje před chybami.
5) Rozbalovací nabídka, ve které si uživatel může zvolit v jakém zařízení bude probíhat simulace zkompilovaného projektu.



Obr. 19 Informace o vytvářeném projektu

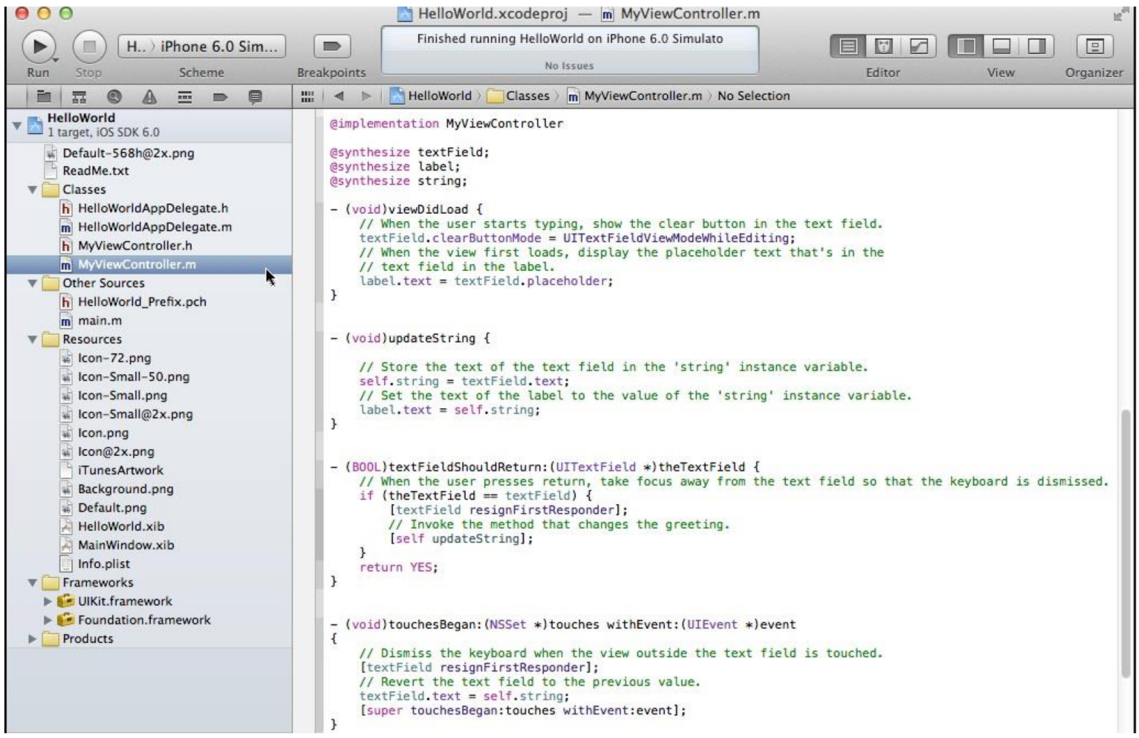
Na obrázku č. 19 je zobrazena stránka obsahující informace k výsledné aplikaci. Výrobce, verze, názvy obrázků pro ikony, povolená orientace otočení displaye atd.



Obr. 20 Základní nastavení vytvářené aplikace

Hned na začátku vytváření aplikace si uživatel může zvolit orientaci otáčení displeje, k dispozici je klasické zobrazení na výšku home buttonem dolů i nahoru a samozřejmě i zobrazení na šířku, pro jednu i druhou stranu. Natočení displeje telefon sám pozná pomocí integrovaného gyroscopu, nebo accelometru a podle toho otočí na požadovanou stranu.

Dále je zde popsána aplikace, verze aplikace a také pro jakou verzi systému bude aplikace dostupná, například můžeme zvolit starší verzi iOS 4.0 pro správnou funkci aplikace i na starém iPhonu 3G.

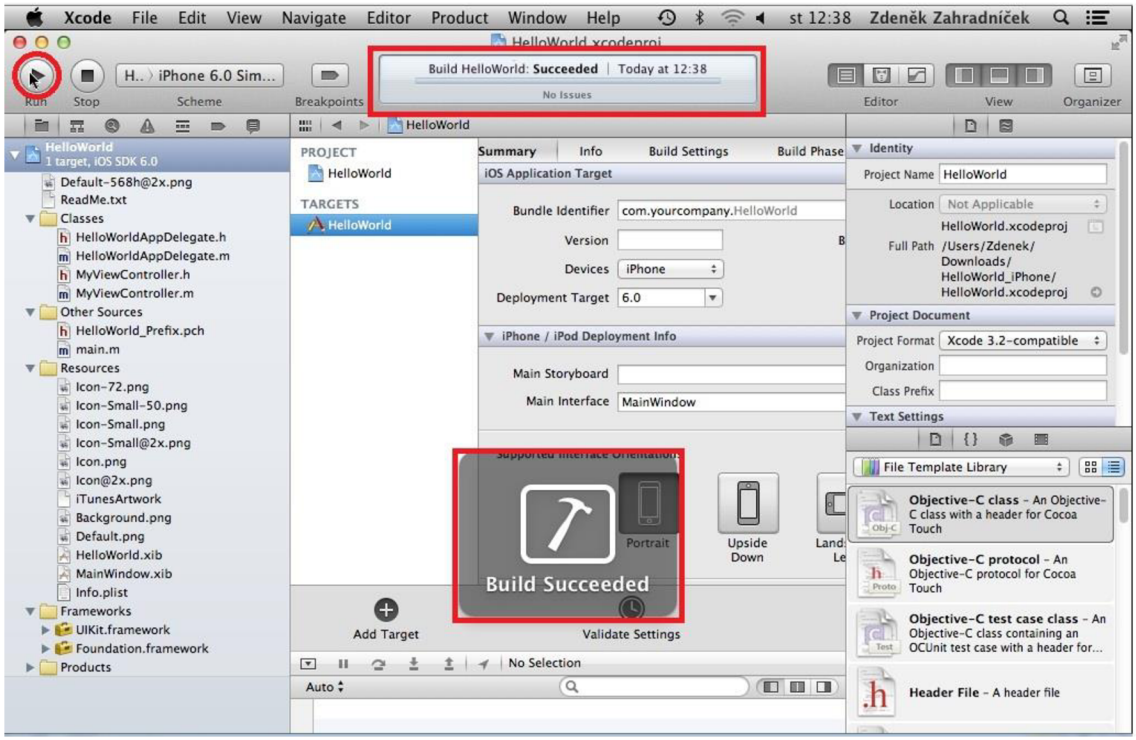


Obr. 21 Základní zdrojový kód programu

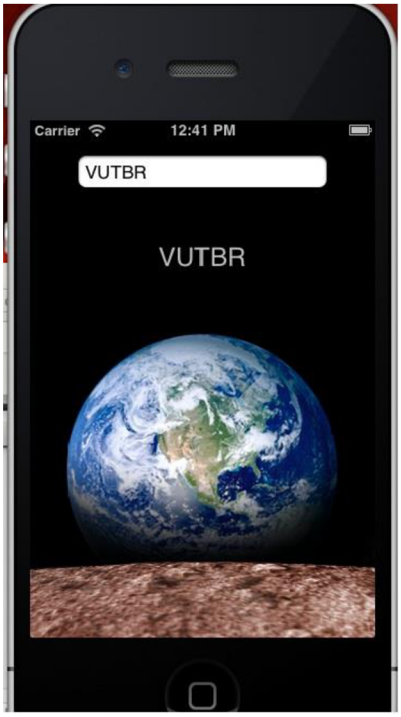
Již hotový projekt pro ukázku práce s Xcode jsem vybral a stáhl na stránkách Apple [3]. Aplikace nese název "HelloWorld" a je to jednoduchá aplikace, která obsahuje jedno základní okno, jeden label a jeden textbox. Po zadání textu do texboxu se nám text zobrazí v labelu.
Na obrázku 20. je zobrazen téměř celý kód pro správnou funkci této aplikace.

Po dokončení aplikace, je pro její funkci třeba udělat její build. Build spustíme v pravém horním rohu programu Xcode, kliknutím na tlačítko RUN. Proces trvá okolo 2-3 vteřin. Záleží však na velikosti vytvářené aplikace. Čím je projekt větší, tím delší dobu buildování zabere. Po dokončení nás program informuje v informační liště, zda proces proběhl v pořádku, popřípadě zobrazí chyby vzniklé například špatným napsáním části kódu.
Jakmile je projekt správně zkompilován, je možné ho otestovat simulací na příslušném zařízení.

“HelloWorld“ aplikaci jsem vytvořil pro zařízení iPhone. Na obrázku 21 je zobrazen její vzhled a funkce.



Obr. 22 Uspěšný Build aplikace



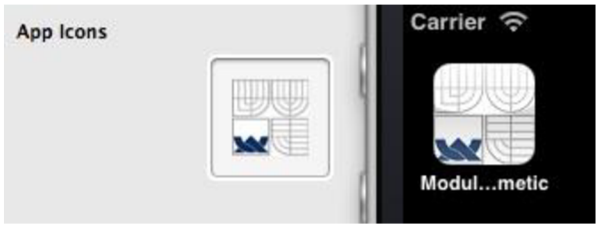
Obr. 23 Simulace zkompilované aplikace na virtualním iPhone

6.2 Realizace aplikace

Mým úkolem byla demontrace kryptografických mechanizmů na reálné aplikaci a zhodnocení rychlosti průběhu jednotlivých operací. Aplikace obsahuje generování velkých čísel o velikosti 1024 bitů , jejich následné využití pro matematické operace (modulární aritmetika), generování hashovacího algoritmu SHA1. Konečným výstupem bylo otestování použitých operací z hlediska jejich časové náročnosti.

6.2.1 Uživatelské prostředí

Jako ikonu aplikace jsem použil logo VUT. Pro možnost využití libovolného obrázku jako ikonu je nutno mít obrázek ve formátu Icon.png a ve velikosti 57x57 pixelů, pro retina display 2x větší, tedy 114x114 pixelů.



Obr. 24 Ikona aplikace

Pro realizaci aplikace jsem v novém projektu zvolil šablonu "Tabbed Application". Tuto šablonu jsem zvolil z důvodu jednoduchého, přehledného a nikterak náročného uživatelského prostředí. Šablona je základem pro aplikace, které využívají "Taby" neboli lišty karet s různými operacemi. V mé aplikaci jsou použity lišty dvě, jedna pro jednotlivé operace a druhá pro měření časové náročnosti operací. Každá lišta musí mít svůj controller, ve kterém má kromě funkcí nadefinované i uživatelské prostředí a způsob jakým se, v mém případě, řádky mají chovat po klepnutí na ně.

Obr. 25 Záložka Single Test

Obr. 26 Záložka Measurement

K vytvoření prostředí je možné použít Interface Builder, který býval dříve samostatnou aplikací, nyní je však integrován do Xcode a umožnuje vývojářům jednoduché propojení GUI vrstvy Cocoa Touch se zdrojovým kódem a také umožnuje vytváření průchodů jednotlivých obrazovek. S jeho využitím si tedy mužeme uživatelské prostředí připravit v grafickém rozhraní. Použití Interface Builderu, zvlášť pro jednoduché aplikace, není nutné, jelikož prostředí mužeme sepsat ve zdrojovém kódu, popřípadě mužeme Interface Builder se zdrojovým kódem kombinovat.

Část kódu pro lištu SingleTest obsažená v SingleTestController.m, je zde ukázka vytvoření lišty(obdobně i druhá lišta pro Measurement). Tato třída slouží pouze pro zobrazování dat, ostatní data a operace jsou obsažena v SingleTestBesiness.

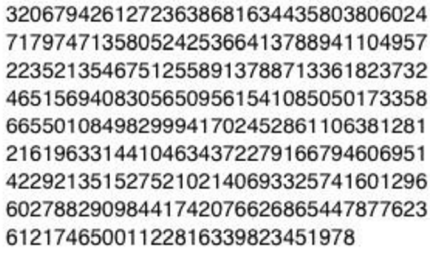
```
- (id)initWithNibName:(NSString *)nibNameOrNil
bundle:(NSBundle *)nibBundleOrNil
{
    self =
[super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
      self.title = NSLocalizedString(@"Single Test", @"Single
Test"); // Název záložky

      self.tabBarItem.image = [UIImage imageNamed:@"first"];
 // Obrázek záložky
        measurement = [[SingleTestBusiness alloc] init];
    }
```

SingleTestController obsahuje ještě jednu důležitou funkci a tou je detailní zobrazení řádků, to slouží hlavně k zobrazení delších textových nebo číselných řetezců delších, než dokaže zobrazit jeden řádek na displej zařízení. Příklad viz. obr. 27, jelikož vygenerované 1024b číslo obsahuje 309 číslic, není možné ho zobrazit do jednoho řádku a proto využijeme DetailViewController, díky němuž zobrazíme celé číslo. DetailViewController je generován Xcodem.

```
// Po klepnutí na buňku(řádek)
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (!self.detailViewController) {
        self.detailViewController =
[[DetailViewController alloc] initWithNibName:@"DetailViewControlle
r" bundle:nil];
    }
    [self.detailViewController setDetailItem:[self stringFromNumber
AccordingTitle:[measurement.titles objectAtIndex:indexPath.row]]];
            // Předání údajů, které mají být zobrazeny
    [self.navigationController pushViewController:self.detailViewCo
ntroller animated:YES];
    // Zobrazí pohled na detail
}
```

Obr. 27 Zobrazení v řádku



Obr. 28 DetailView

V liště Measurement je zobrazení výsledných časů generováno pomocí samostatného tlačítka, ale po kliknutí na řádek následovně:

```
// po kliknutí na řádek
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
[tableView deselectRowAtIndexPath:indexPath animated:YES]; //
Animace - modrá barva
    UITableViewCell *cell =
[tableView cellForRowAtIndexPath:indexPath];
    [[cell textLabel] setHidden:YES];    // Skryje popisek labelu
    [cell addSubview:loader]; // během průběhu operace se zobrazí
kruhový loader
   [loader setFrame:CGRectMake(cell.frame.size.width/2-
loader.frame.size.width/2, 4, loader.frame.size.width, loader.frame
.size.height)];
    [loader startAnimating]; // .. začne animace
    // spuštění testu
    [measurement runTest:indexPath.row];
}
```

Vysledný čas se zobrazí v náležejícím řádku v mili sekundách. Například Random Numbers s přesností na 18 desetinných míst, což se bezproblému v řádku zobrazí a není zde nutno využívat DetailView.

Obr. 29 Výpis výsledku časového testu

Záložka Measurement obsahuje i tlačítko, které po stisknutí spustí všechny testy najednou. Je definováno v MeasurementController takto:

```
// Run All Tests button
self.navigationItem.leftBarButtonItem =[UIBarButtonItem alloc
] initWithTitle:
NSLocalizedString(@"Run Tests", @"Run All Tests")
style:UIBarButtonItemStyleBordered target:self action:@select
or(runAllTests)];
```

Tlačítko odkazuje na runAllTests:

```
- (void)runAllTests
{
    currentTestIndex = 0; //index testu
    [self runNextTest];
}
- (void)runNextTest
{
    if (currentTestIndex < [measurement.numbers count]) {
        NSIndexPath *indexPath =
[NSIndexPath indexPathForRow:currentTestIndex inSection:0];
        [self tableView:self.tableView didSelectRowAtIndexPat
h:indexPath];
        currentTestIndex++;
    }
}
```

6.2.2 Knihovna GMP

Prvním úkolem bylo generování náhodných čísel o velikosti 1024bit. Jelikož xcode takto rozsáhlé datové typy neobsahuje, bylo nutno přidat knihovnu GMP.
GMP je volně šířitelná knihovna pro aritmetiku, operace pro racionální čísla, a čísla s plovoucí desetinnou čárkou. Neobsahuje prakticky žádný limit přesnosti, záleží jen na paměti zařízení na kterém je knihovna použita. GMP obsahuje mnoho funkcí a hlavním cílem aplikací, ve kterých se využívá, jsou kryptografické aplikace a výzkumné výpočetní aplikace.

Knihovna je velni pečlivě navržena tak, aby rychlost byla co největší pro všechny malé i velké operace. Rychlosti je dosaženo použitím tak zvaných fullwords jako základního aritmetického typu s využitím rychlého algoritmu s vysoce optimalizovaným kodem pro většinu procesorů a s velkým důrazem na rychlost. Hlavní cílové platformy pro využití GMP jsou systémy unixového typu jako GNU/Linux, Solaris, MAC OS X, BSD a samozřejmě také Windows ve 32bitovém i 64bitovém modů.

Knihovnu jsem zdarma stáhl na webových stránkách výrobce a pomocí terminálu nainstaloval do systému, knihovnu bylo nejprve třeba zkompilovat pro architekturu našeho operačního systému (MAC OS X). Poté bylo nutné nastavení viditelnosti knihovny pro xcode k jejímu následnému využití v projektu, a to tak, že jsem v built settings v záložce Search Patch nastavil celou cestu, kde se části knihovny nachází.
Pro využívání základních funkcí knihovny a lepší přehlednost jsem zvolil wrapper GMPint.

Pro generování velkých čísel o hodnotě 1024bit jsem do GMPint přidal příkazy pro generování, které jsem vyčetl v manuálu knihovny GMP:

```
}
+ (GMPInt *)randomObj {
    mpz_t rand_Num;
    unsigned long int seed;
    gmp_randstate_t r_state;

    seed = arc4random(); // standartní funkce pro generování
náhodných čísel

    gmp_randinit_default(r_state);
    gmp_randseed_ui(r_state, seed);

    mpz_init(rand_Num);

    mpz_urandomb(rand_Num,r_state,1024); // definování
velikosti generovaného čísla
    GMPInt *random = [[GMPInt alloc] initWithValue:rand_Num];

    gmp_randclear(r_state);
    mpz_clear(rand_Num);

    return random;
}
```

Dále je generování využíváno jen voláním funkce. Příklad volání v single testu pro první a druhé číslo :

```
- (void)singleTest
{
    GMPInt* firstNumber = [GMPInt randomObj];
    GMPInt* secondNumber = [GMPInt randomObj];
    GMPInt* thirdNumber = [GMPInt randomObj];
}
```

Po kliknutí na tlačítko "Generate" se vygenerují náhodná 1024b čísla v single testu a jejich zobrazení v řádku proběhne díky následujícím příkazům(přiklad pro jedno náhodně číslo):

```
    _titles = [NSArray arrayWithObjects:@"First Random a", //
zobrazené popisy řádků
            nil];
 _values =
[NSArray arrayWithObjects:[NSNumber numberWithInt:0],
//Hodnoty vygenerovaných čísel,počet řádků(každá operace jeden)
                  nil];
        _numbers =
[NSMutableDictionary dictionaryWithObjects:_values forKeys:_tit
les];  // Vše do dictionary pro lepší manipulaci
```

Generování – za použití knihovny GMP

```
GMPInt* firstNumber = [GMPInt randomObj];
GMPInt* secondNumber = [GMPInt randomObj];
GMPInt* thirdNumber = [GMPInt randomObj];
```

Po vygenerování čísel jsou provedeny početní operace, které jsou umístěny do řádku pod generovanými čísly, stejným způsobem jaký je popsaný výše. Operace jsou nadefinovány v BigNumberOperations a jejich volání probíhá v SingleTestBusiness. Opět je zde využívána knihovna GMP pomocí wraperu GMPint.

```
GMPInt* addition =
[BigNumberOperations additionFirst:firstNumber toSecond:secondN
umber]; //sčítání
GMPInt*substraction=[BigNumberOperations substractionSecond:sec
ondNumber fromFirst:firstNumber]; //odčítání
GMPInt* modMulti =
[BigNumberOperations modularMultiplicationWithA:firstNumber and
B:secondNumber andC:thirdNumber]; //Modulární násobení
GMPInt* modSquar =
[BigNumberOperations modularSquaringWithA:firstNumber andB:
secondNumber andC:thirdNumber]; //Modulární mocnění
NSString* firstSha1 =
[BigNumberOperations sha1From:firstNumber]; // Generování sha1
z prvního čísla(obdobně z druhého čísla)
```

6.2.3 Operace

Modulární násobení (generování čísel do proměnných a následné jejich užití k výpočtu):

```
    GMPInt *tmpA =
[[GMPInt alloc] initWithValue:*[a valPtr]];
    GMPInt *tmpB =
[[GMPInt alloc] initWithValue:*[b valPtr]];
    GMPInt *tmpC =
[[GMPInt alloc] initWithValue:*[c valPtr]];
    [tmpA multiplyWithGMP:tmpB];
    [tmpA modulusWithGMP:tmpC];
    return tmpA;}
```

Modulární mocnění:

```
{ GMPInt *tmpA = [[GMPInt alloc] initWithValue:*[a valPtr]];
    GMPInt *tmpB =
[[GMPInt alloc] initWithValue:*[b valPtr]];
    GMPInt *tmpC =
[[GMPInt alloc] initWithValue:*[c valPtr]];
[a powerWithGMP:b onModulo:c];
 return tmpA;
```

Sčítání:

```
    GMPInt *temp =
[[GMPInt alloc] initWithValue:*[firstNumber valPtr]];
    [temp addWithGMP:secondNumber];
    return temp;}
```

Odečítání:

```
 GMPInt *temp =
[[GMPInt alloc] initWithValue:*[firstNumber valPtr]];
    [temp subtractWithGMP:secondNumber];
    return temp;
```

Generování šifrovacího klíče SHA-1 (SHA-1 je nejrozšířenějším stávajících SHA hash funkcí. Nevyužívá knihovnu GMP, jen číslo pomocí ní vygenerované):

```
+ (NSString*)sha1From:(GMPInt*)number
{
    NSString *string = [number stringValue];
    return [self sha1:string];
}
+ (NSString*)sha1:(NSString*)input
{
    const char *cstr =
[input cStringUsingEncoding:NSUTF8StringEncoding];
    NSData *data = [NSData dataWithBytes:cstr length:input.leng
    uint8_t digest[CC_SHA1_DIGEST_LENGTH];
    CC_SHA1(data.bytes, data.length, digest);
 NSMutableString*output=
[NSMutableString stringWithCapacity:CC_SHA1_DIGEST_LENGTH * 2];

    for(int i = 0; i < CC_SHA1_DIGEST_LENGTH; i++)
        [output appendFormat:@"%02x", digest[i]];
    return output;
```

Záložka Measurement, jak již bylo řečeno výše, obsahuje časové testy operací. Testy je možné spustit jednotlivě (po kliknutí na řádek s určitou operací) nebo je spustit všechny najednou. Všechny příkazy pro záložku jsou definovány obdobně jako u záložky Single test a jsou obsazeny v MeasurementBusiness a MeasurementController. Operace prováděné v jednotlivých řádcích jsou vytvořeny jako potomci od TestStubParrent. Potomky jsou MultiplyTest, SquaringTest a SubstractionTest obsahují generování náhodných 1024bit čísel, proměnné a příslušná operace.
Testy jsou spouštěny ve vedlejším vlákně. Pro testy jsou definované pre, in a post order metody, čas je sbíraný pouze pro inOrder.

Test modulárního násobení (vygenerování čísel a výpočetní operace):

```
- (void)preOrder
{
    a = [GMPInt randomObj];
    b = [GMPInt randomObj];
    c = [GMPInt randomObj];
}
- (void)inOrder
{
    [a multiplyWithGMP:b];
    [a modulusWithGMP:c];
}
```

Test modulárního mocnění (vygenerování čísel a výpočetní operace):

```
- (void)preOrder
{
    a = [GMPInt randomObj];
    b = [GMPInt randomObj];
    c = [GMPInt randomObj];
}
- (void)inOrder
{
   [a powerWithGMP:b onModulo:c];
}
```

Volání a spouštění těchto testů podle indexu se nachází v MeasurementBusiness(příklad jen pro randomNumberTest):

```
 (void)runTest:(int)index
{
    id<TestStub> testStub;
    switch (index) {
        case 0:
            testStub =
[[RandomNumberTest alloc] initWithIndex:index];
            break;
                .
                .
            return;
    }
```

Provedení testů na pozadí :

```
}
[self performSelectorInBackground:@selector(measurementWithTe
st:) withObject:testStub];
}
```

Metoda pro provedení testu – čas je šcítám po provedení každého opakování operace podle času startu a času ukončení operace (metoda běží v jádru):

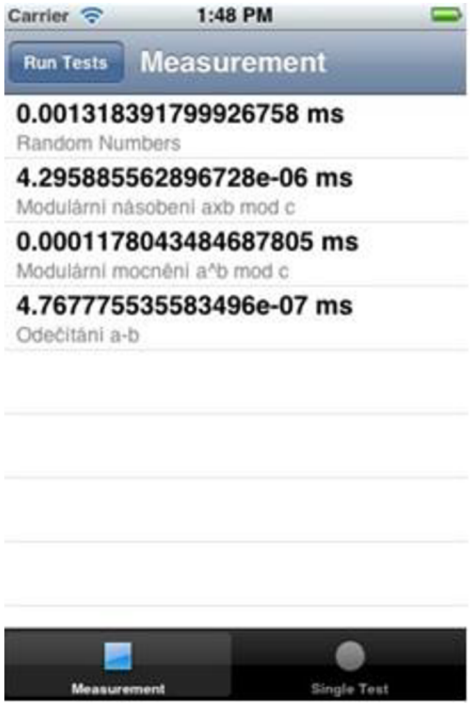
```
- (void)measurementWithTest:(TestStubParrent*)testStub
{
    double timeElapsedSum 0;  //Součet celkového času
    NSDate *methodStart, *methodFinish;

    if ([testStub respondsToSelector:@selector(preOrder)]) {
        [testStub preOrder];
    }
    for (uint i = 0; i < TEST_COUNT; i++) { //opakování
testů(po jednom)
        methodStart = [NSDate date];  //Začatek testu
        [testStub inOrder];
        methodFinish = [NSDate date]; //Konec testu
        NSTimeInterval executionTime =
[methodFinish timeIntervalSinceDate:methodStart];
        timeElapsedSum += executionTime;
```

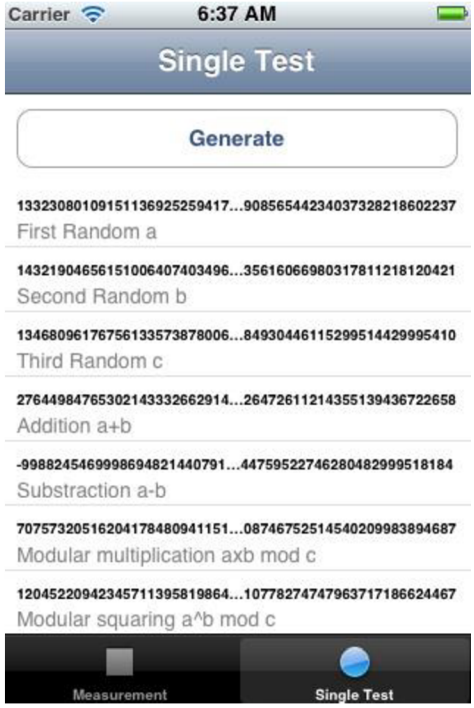
Metoda pro získání výsledného času :

```
NSNumber *timeElapsedAverage =
[NSNumber numberWithDouble:(timeElapsedSum
/ TEST_COUNT)];   // Vysledný průměrný čas pro jeden test
    [testStub setTimeElapsed:timeElapsedAverage];
    [self performSelectorOnMainThread:@selector(didTestDone:)
 withObject:testStub waitUntilDone:NO];    // Konec
s notifikací "main thread", že test je hotov
}
```

TEST_COUNT = počet kolikrát bude test opakován v jednom měření. Touto hodnotou bude vydělen výsledný čas, výsledkem je průměrný čas pro provedení žádané operace.



Obr. 30 Výsledky měření

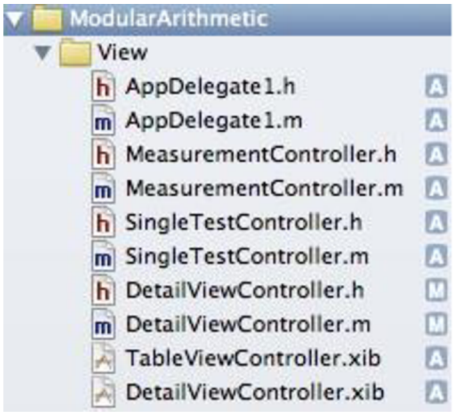


Obr. 31 Generovaná čísla a provedené operace

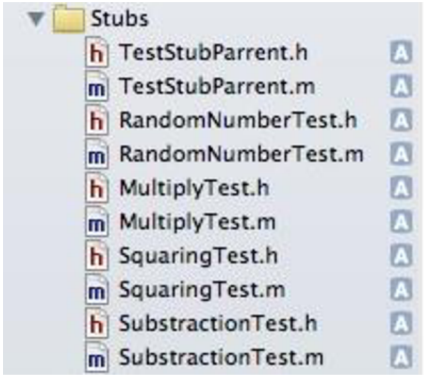
Projekt je rozčleněněn do třech složek pro lepší orientaci. První složkou je "view" v té jsou obsaženy především kontrolery jednotlivých záložek. "View" tedy slouží pro zobrazení prováděných akcí obsažených v ostatních složkách.

Druhou složkou s nadefinovanými početními operacemi je "Business". Je zde obsaženo nasezení do řádků a popis a operací, které mají být v jednotlivých složkách uskutečněny.

V poslední složce "Stubs" se nachází logika pro měření časů v záložce "Measurement" pro jednotlivé operace.



Obr. 36 View



Obr. 37 Stubs (měření času)

6.3 Ovládání aplikace

Aplikace obsahuje dvě záložky. První záložka "Measurement" obsahuje měření průměrných časů jednotlivých operací. Měření lze spustit po kliknutí na jednotlivý řádek, který chceme měřit, nebo tlačítkem "Run Tests" spustíme testy všechny.

V druhé záložce "Single Test" nalezneme výpis výsledků pro jednotlivé operace (Generovaná náhodná čísla, modulární mocnění a násobení, odčítání, sčítání, generování hash). Zde nelze spustit testy jednotlivě. Je nutné kliknout na tlačítko "Generate", které vygeneruje náhodná čísla. Vygenerovaná čísla poté slouží k ostatním prováděným operacím. Jelikož jsou výsledné hodnoty příliš velké a nevejdou se do řádků, je zde po kliknutí na řádek použit detailní náhled na jeho obsah.

6.4 Test rychlosti operací

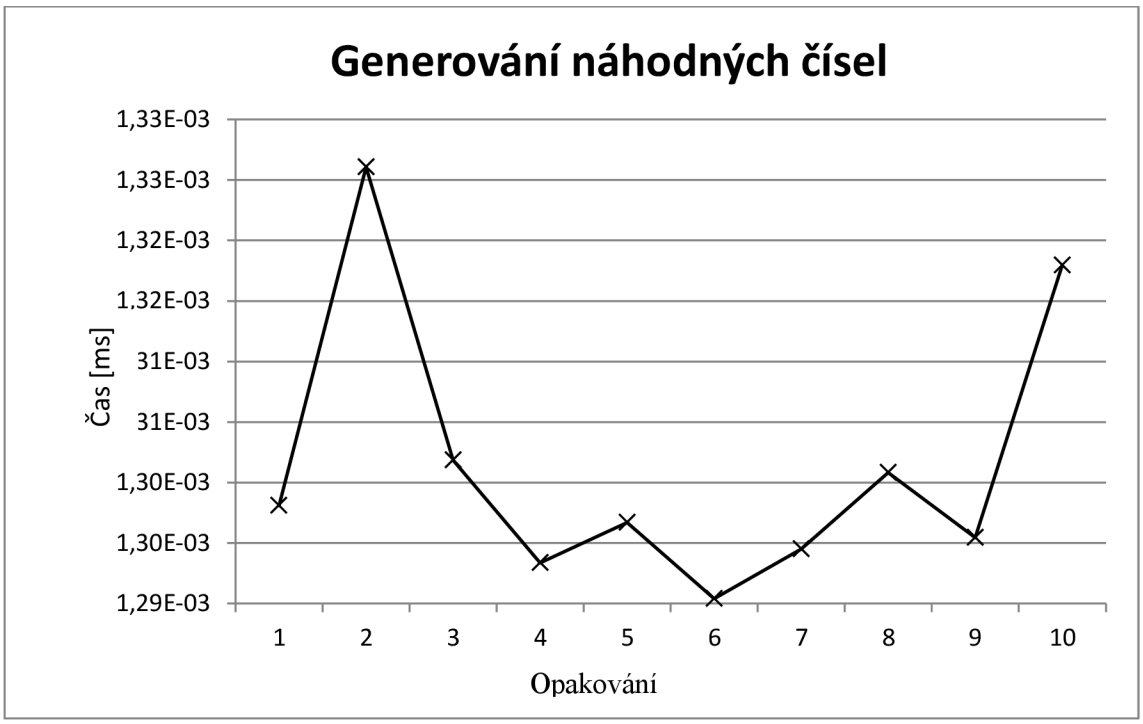
Měření rychlosti operací probíhá v záložce "Measurement". Nejprve jsou pro každou operaci vygenerována náhodná čísla, ty jsou uložena do proměnných A,B,C a s těmito čísly je příslušná operace 1000x opakována. Výsledné časy jednotlivých testů jsou k sobě přičítány a nakonci zprůměrovány(vyděleny počtem opakování).

Testy jsem nechal proběhnout 10x s 1000x opakováním a výsledné časy vynesl do grafů. Z průměrných výsledků zjistíme, že nejrychlejší operací je odečítání

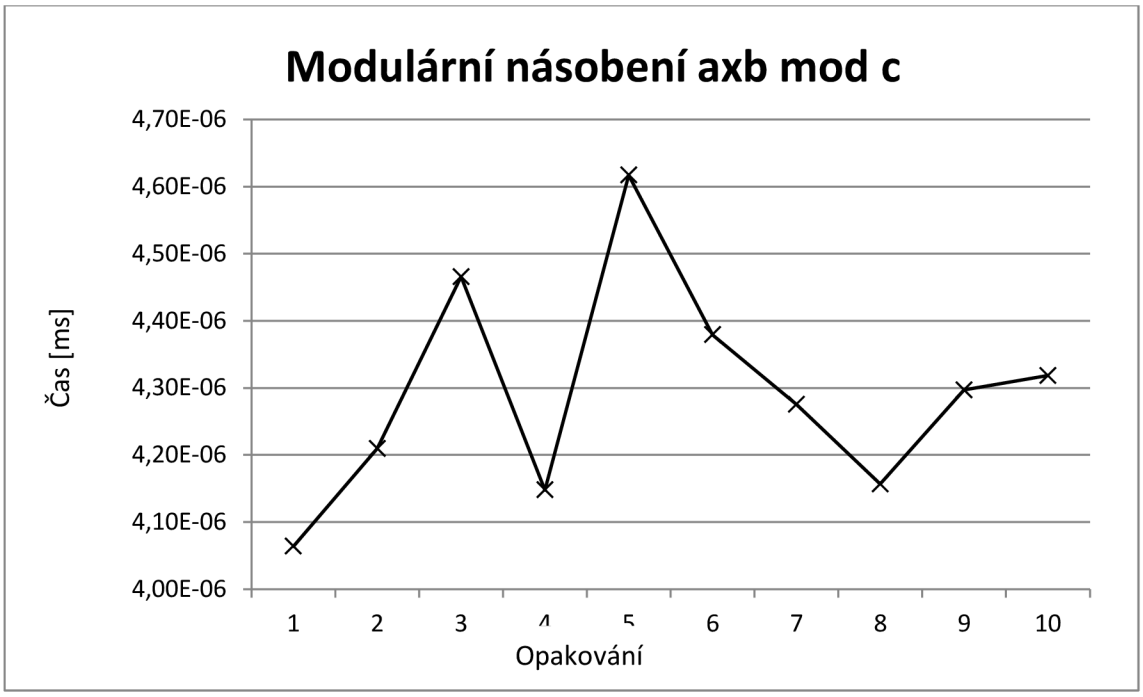
s průměrným časem provedení operace 4,5*10^-7 ms. Naopak nejpomalejší operací je modulární mocnění s průměrným časem 1,85*10^-3 ms. U modulárního mocnění je narozdíl od odečítaní pracováno se třemi náhodnými čísly a je s nimi nutno provést dvě operace, a to nejprve umocnění prvního čísla A druhým číslem B a následně umocněné číslo A modulo C.

Druhou nejrychlejší operací je modulární násobení (A * B modulo C) s průměrným časem jednoho výpočtu 4,3*10^-6 a třetí je generování náhodných čísel s průměrnou dobou trvnání 1,31*10^-3.

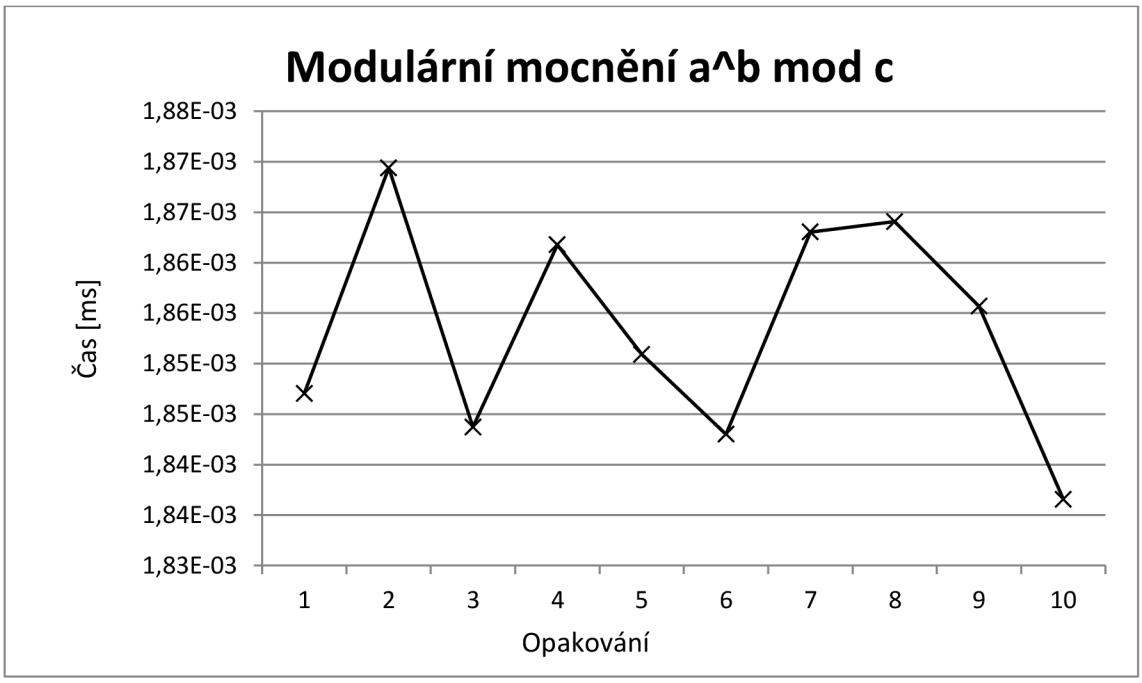
Z grafů je vidět, že operace jsou vždy prováděny se skoro stejným výsledkem. Například u generování náhodných čísel se liší maximálně o 4*10^-5 ms a mužeme tedy říct, že prováděné operace jsou stabilní.



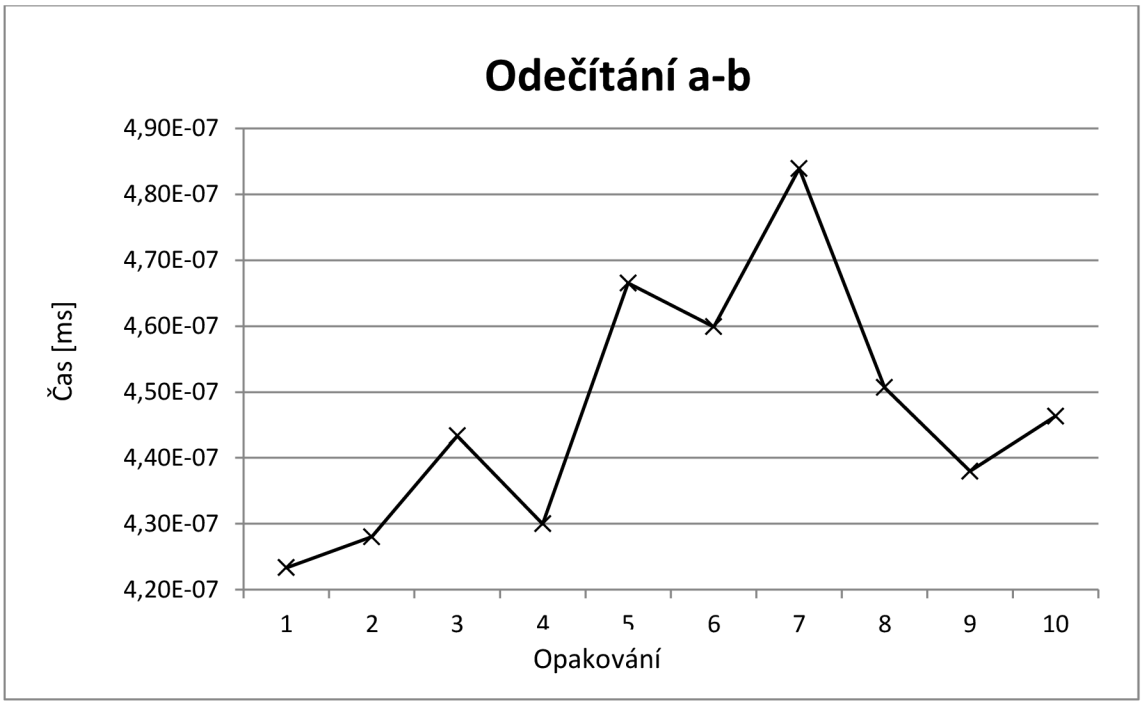
Obr. 32 Graf 10ti opakování generování náhodných čísel



Obr. 33 Graf 10ti opakování pro modulární násobení a*b mod c



Obr. 34 Graf 10ti opakování modulárního mocnění a^b mod



Obr. 35 Graf 10ti opakování odečítání

Aplikace byla testována pouze na simulátoru iPhone, který je součástí balíčku Xcode. K testů na realném hardwaru jsem se bohužel nedostal.

Simulátor plně nahrazuje reálné zařízení, ovšem v ideálních podmínkách, testy nejsou ovlivněny vytížením zařízení vedlejšími aplikacemi a podaplikacemi bežících na pozadí systému, jak by k tomu docházelo v reálných podmínkách. Z toho mužeme usoudit, že výsledné časy prováděných aplikací by byli na reálném hardwaru o něco vyšší.

ZÁVĚR

Bakalářská práce se zabývala způsoby zabezpečení operačního systému iOS od společnosti Apple. V první části práce byly popsány obecné vlastnosti systému, řešení přísných bezpečnostních kontrol ověřování, od startu systému až po odesílání aplikací vývojáři na Apple Store. Dále základní i pokročilé možnosti zabezpečení pomocí tříd, hesel a rozdělení úložiště na části s různou prioritou zabezpečení dle potřeby vyžadovaného bezpečnostními prvky.

Další částí, kterou se práce zabývala je zabezpečení sítě a síťových prvků. Jakým způsobem je řešena ochrana odesílaných a přijímaných dat při připojení iOS zařízení k sítím pomocí Wi-Fi, nebo mobilní datovou sítí poskytovanou operátory. V práci je popsána i funkce VPN a Bluetooth na iOS.

Poslední částí je část praktická. Ta je zaměřena na programování pro iOS platformu v programovacím balíčku Xcode. K práci v Xcode jsem se vzdáleně připojoval na školní Mac server. V práci je popsána funkce tohoto programu a následně v něm pro ukázku zkompilována jednoduchá aplikace s popisem základních částí Xcodu a jeho uživatelského prostředí. Po seznámení s tímto programovacím balíčkem jsem začal pracovat na samostatném programu, který demostruje využívané šifrování RSA na mobilních zařízeních a obsahuje také měření rychlosti prováděných operací s čísly, které se při šifrování využívají. Aplikace splnůje všechny zadáním požadované funkce – generování čísel o velikosti 1024bit, modulární aritmetiku, početní operace, generování hash sha1 a časové testy rychlosti operací.

Dle mého názoru má Apple svůj operační systém pro mobilní zařízení, po všech stránkách, velmi dobře zabezpečen. Seznámením s tímto tématem jsem se dozvěděl mnoho nových informací a do značné míry osvojil možnosti a funkce programu Xcode.

LITERATURA

[1] STALLINGS, William. Cryptography and Network Security: Principles and Practice (5th Edition). USA : Prentice Hall, 2010. 744s. ISBN 0136097049.

[2] IOS Security. APPLE. *IOS Security* [online]. 2012 [cit. 2013-06-05]. Dostupné z: http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf

[3] Xcode Description. *Itunes Apple* [online]. [cit. 2012-11-05]. Dostupné z: http://itunes.apple.com/us/app/xcode/id497799835

[4] RSA description. *RSA* [online]. [cit. 2012-11-30]. Dostupné z: http://www.karlin.mff.cuni.cz/~holub/soubory/qc/node23.html

[5] RSA description. *RSA* [online]. 2012 [cit. 2012-11-30]. Dostupné z: http://www.oocities.org/hmaxf_urlcr/rsa.htm

[6] SHA. [online]. 2013 [cit. 2013-03-15]. Dostupné z: http://csrc.nist.gov/groups/ST/toolkit/index.html

[7] Apple development [online] 2013 [cit. 2013-10-05]. Dostupné z: https://developer.apple.com/library/ios/navigation/#

[8] Bluetooth. *Pcword* [online]. 2012 [cit. 2012-11-15]. Dostupné z: http://pcworld.cz/hardware/Zaklady-technologie-Bluetooth-puvod-a-rozsah-_funkci-6635

[9] VPN. *Ondřej Průcha* [online]. 2005 [cit. 2012-11-15]. Dostupné z: http://home.zcu.cz/~ondrous/

[10] Apple. APPLE. [online]. 2013 [cit. 2013-5-30]. Dostupné z: http://www.apple.com

SEZNAM PŘÍLOH

CD obsahující bakalářskou práci v pdf a aplikaci vytvořenou v Xcode.