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

ZABEZPEČENÍ KOMUNIKACE V SYSTÉMECH VYUŽÍVAJÍCÍ OMEZENÁ ZAŘÍZENÍ

COMMUNICATION SECURITY IN SYSTEMS USING LIMITED DEVICES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE Tomáš Michálek

AUTHOR

VEDOUCÍ PRÁCE Ing. Lukáš Malina

SUPERVISOR

BRNO 2014

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Tomáš Michálek ***ID:*** 134559
Ročník: 3 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Zabezpečení komunikace v systémech využívající omezená zařízení

POKYNY PRO VYPRACOVÁNÍ:

Analyzujte využitelnost kryptografických metod na mikrokontrolérech s omezenou výpočetní a paměťovou kapacitou. Implementujte na vybranou platformu mikrokontroléru vhodně vybrané kryptografické metody (symetrické šifry, hash funkce, digitální podpisy), změřte doby výpočtu a analyzujte paměťovou náročnost daných operací.

Výstupem bakalářské práce bude návrh a implementace komplexního a efektivního zabezpečení komunikačního záhlaví u komunikačního systému klient-server, kde klient využívá výpočetně a paměťově omezené 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.

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 4.6.2014

Vedoucí práce: Ing. Lukáš Malina
Konzultanti bakalářské práce:

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

UPOZORNĚNÍ:

ABSTRAKT

Tato práce pojednává o využitelnosti a analýze kryptografických metod na zařízeních s nízkým výpočetním výkonem a malou pamětí. Práce popisuje různé metody a uvádí příklady jejich použití. Podle určitých parametrů zabezpečovacích metod, tedy velikosti šifry a náročnosti na operační paměť, jsou vybrány a následně porovnány symetrické šifry a autentizační kódy a dále jsou rozebrány bezpečnostní protokoly a možné útoky. V následující části práce je následné porovnání šifer mezi sebou a vybraní kandidáti jsou otestováni na zapůjčeném mikrokontroléru. Poté je navržen a implementován zabezpečovací protokol ve dvou verzích, změřeny parametry a provedena analýza výsledného řešení.

KLÍČOVÁ SLOVA

Šifrování, mikrokontrolér, symetrická kryptografie, hash funkce, autentizace, blokové šifry, protokoly, útoky

ABSTRACT

This thesis presents the usability and analysis of cryptographic methods on devices with low computing power and low memory capacity. The thesis describes different methods and shows examples of their usability in practice. According to certain parameters of security methods, such as code size and memory size it is selected and compared symetric ciphers and authentication codes. There are also described some security protocols and possible attacks. The next part includes a comparison of ciphers and chosen candidates are tested on our microcontroller. In the end there are propose, implemented and analyzed security protocols.

KEYWORDS

Encryption, microcontroller, symmetric cryptography, hash function, authentication, block ciphers, protocols, attacks

MICHÁLEK, T. *Zabezpečení komunikace v systémech využívající omezená zařízení*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 52 s. Vedoucí bakalářské práce Ing. Lukáš Malina.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Zabezpečení komunikace v systémech využívající omezená zařízení 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 bakalářské 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 30. 5. 2014 .....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Lukáši Malinovi za odborné rady a motivující přístup. Dále děkuji firmě Honeywell za zapůjčení mikrokontroléru a firmě Texas Instruments za udělení licence pro vývojové studio Code Composer Studio a také firmě IAR systems za udělení licence pro IAR Embedded Workbench.

V Brně dne 30. 5. 2014 .....

(podpis autora)

OBSAH

ÚVOD

Tato práce si klade za cíl přiblížit možnosti zabezpečení a využitelnost kryptografických metod na mikrokontrolérech s omezenou výpočetní a paměťovou kapacitou. Nejprve se čtenář dozví základy kryptografie, které jsou nezbytné k pochopení celé problematiky. Právě tímto se práce zabývá v první kapitole. Čtenář zjistí, co to jsou bezpečnostní služby, jak fungují a jaké části obsahují. Dále práce uvádí symetrické šifry a jejich rozdělení na blokové šifry s popisem používaných módů a proudové šifry. Následně je popsán princip asymetrické kryptografie včetně příkladů nejznámějších zástupců a použití asymetrických šifer v praxi.

Druhá kapitola se zabývá bezpečnostními mechanismy protokolů, které se často používají v praxi, a každý jednotlivě popisuje. Následují vybrané kryptografické útoky, které útočník může využít k prolomení zabezpečení komunikace či k jejímu obejití. Dnešní kryptografické zabezpečovací protokoly jsou vůči takovým útokům méně či více odolné a právě tyto protokoly jsou popsány ve čtvrté kapitole práce. Kapitola se zaměřuje na zabezpečení komunikace na síťové a komunikační vrstvě a popisuje i prvky, které jsou těmito protokoly využívány. Protokoly řídicích systémů jsou popsány ve stejnojmenné kapitole. Zde je vysvětlen a popsán systém Scada a jeho zabezpečené variace. Následně je rozebrán protokol Modbus, jeho tři verze, jejich popis, zobrazení a struktura rámce. Na konci kapitoly je nastíněna bezpečnostní slabina tohoto protokolu.

V následující části práce se čtenář může dovědět informace o omezených zařízeních, konkrétních symetrických šifrách stejně jako o autentizačních kódech. Cílem této kapitoly je čtenáři přiblížit specifické omezení mikrokontrolerů a popsat určité symetrické šifry vybrané právě dle těchto omezení. Seznámíme se také s vybranou funkcí hash a autentizačním kódem MAC a dle porovnávaných parametrů vybereme 2 algoritmy pro testování na našem zařízení. V předposlední kapitole se budeme zabývat návrhem vlastního zabezpečovacího protokolu ve dvou verzích, kdy díky předchozím teoretickým informacím navrhneme první řešení pouze s autentizací a následně druhou verzi s autentizací i šifrováním. Tyto dva protokoly si následně popíšeme a ukážeme si úsek kódu v jazyce C, spustíme na našem mikrokontroléru a změříme sledované parametry. Nakonec protokoly analyzujeme a zjistíme jejich odolnost vůči popsaným útokům.

1 ÚVOD DO KRYPTOGRAFIE

Na úvod si ujasníme, co rozumíme pod pojmem kryptografie. Kryptografie je věda o různých způsobech utajení smyslu a obsahu předávaných zpráv mezi dvěma stranami. Toto utajování probíhá převodem zprávy do podoby, která je čitelná jen se speciální znalostí, jde tedy o znehodnocení zprávy pro všechny nezasvěcené. Pokud bychom chtěli zašifrovanou zprávu rozluštit bez této speciální znalosti, jednalo by se o kryptoanalýzu. Kryptografie a kryptoanalýza se dohromady nazývají kryptologie.[1]

1.1 Pojmy v kryptografii

1.1.1 Šifra/šifrování

Jedná se o kryptografický algoritmus, který upraví původní zprávu na její nečitelnou podobu. Vzniklou šifru musíme před přečtením dešifrovat.

1.1.2 Klíč

Tajná informace, která nám umožní šifrovanou zprávu dešifrovat. Metody jak prolomit šifru a dostat se k původní zprávě bez znalostí klíče popisuje kryptoanalýza.

1.2 Historie kryptografie

Dnešní potřeba kryptografie je daleko větší, než tomu kdy bylo v minulosti. Čím větší je technologická vyspělost, tím víc přibývá komunikací mezi jednotlivými entitami a potřeby tuto komunikaci zabezpečit za použití kryptografie. Samo slovo kryptografie pochází z řečtiny – kryptós (skrytý) a gráphein (psát). Jako první doložená zmínka o kryptografii se považuje použití nestandartních hieroglifů egyptskými písaři. Dalším příkladem použití je šifrovaný popis na hliněných deskách glazurování výrobků v keramických dílnách v Mezopotámii. První pomůcku ke kryptografii vynalezli Řekové, kteří používali zařízení nazvané „scytale“. Jednalo se o dřevěnou tyč, na které byl trochu našikmo namotán papyrový proužek, na který se kolmo psalo. Po odmotání se mohla zpráva přečíst pouze tehdy, pokud měl příjemce zprávy tyč o stejném průměru. Scytale se datuje do období 5 století př. n. l. Dalším průkopníkem v oblasti kryptografie byl Julius Caesar, který používal lehkou substituční šifru (Caeserova šifra). Za zmínku stojí i arabský matematik Abú Bakr Ahmad nebo encyklopedie s popisem kryptografických metod z roku 1412 nazvaná Subh al-á sha. Giovan Battista della Porta (1540-1610), který zavedl myšlenku použití hesla jako šifrovacího klíče a položil tak základy moderní kryptografie nebo např. Thomas Jefferson, který společně s matematikem Robert Pattersonem vynalezl kotoučovou šifru. A samozřejmě Enigma,

která je postavena na metodě Albertiho disku. Enigma byla ve své době považována za nejdokonalejší systém. Ovšem kvůli chybě dvojího opakování klíče byla Enigma 1. května 1940 rozluštěna. Po druhé světové válce se kryptografickými metodami stávají především matematické algoritmy, jelikož díky zvyšujícím se výpočetním možnostem začala komunikace probíhat v digitální podobě. Po rozšíření začal internet fungovat jako přenosové médium a kvůli široké dostupnosti se k datům mohla dostat větší skupina lidí. V polovině 70. let byla proto představena symetrická šifra DES, která sloužila bankám a jiným finančním institucím k zabezpečení dat. Tato šifra byla v roce 2001 nahrazena šifrou Rijndael, což byl vítězný kandidát na AES (Advanced Encryption Standard). Už v roce 1997 byl možný výpočetní výkon dostatečný na prolomení DES hrubou silou. Velkým problémem symetrických šifer byl způsob předání společného klíče protistraně. V roce 1976 pánové Whitfield Diffie a Martin Hellman zveřejnili metodu předání klíče. Od tohoto roku se taktéž začaly vyvíjet nové algoritmy a těmi byly asymetrické šifry, které mají veřejný a soukromý klíč a tudíž odpadá problém s předáním klíče. [2-6]

1.3 Bezpečnostní služby

Při přenosu informací je důležitá otázka důvěrnosti dat: jsou data, která jsem vyslal shodná s těmi, které obdrží příjemce? Na přenosovém kanálu může být útočník, který námi vyslanou zprávu může podvrhnout za falešnou, nebo jen zamění část zprávy. Tomu lze zabránit použitím vhodných šifer a jejich módů, např. blokovou šifru s použitím módu CBC, kdy dešifrování každého bloku závisí na bloku předchozím. Tím se zabraňuje nahrazení části zprávy. Pokud bychom chtěli ověřit, jestli není celá zpráva podvrh, tak použijeme digitální podpis, který nám zaručí, že zpráva je poslána opravdu od původního odesílatele. Aby se služba dala považovat za bezpečnou, musí splňovat důvěrnost, integritu dat a autentičnost. [1]

1.3.1 Důvěrnost

Zamezení přístupu neautorizovaných osob k citlivým a důvěrným informacím a udržení důvěrných dat v tajnosti. Toho lze dosáhnout buď kontrolou fyzického přístupu k datům nebo šifrováním důvěrných dat. [1]

1.3.2 Integrita dat

Jde o zamezení neoprávněné modifikace dat. Za modifikaci se považuje smazání části dat, vložení nových dat, nebo substituce části stávajících dat jinými daty. Nejedná se pouze o zamezení neoprávněné modifikace dat, ale také o schopnost tuto modifikaci detekovat. Zajištění integrity dat přenášených (a šifrovaných) dat je kupodivu velkým praktickým problémem mnoha moderních systémů a přetrvává do současnosti, stejně jako mýtus, že šifrování řeší všechny problémy bezpečnosti. [1]

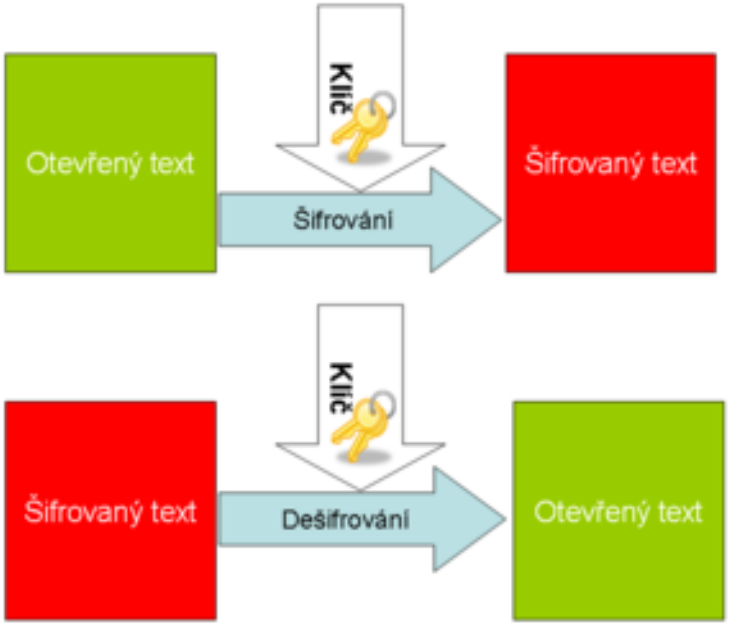
1.3.3 Autentičnost

K ověření autentičnosti, jinak řečeno hodnověrnosti zprávy, slouží digitální podpisy. Pomocí těchto pečetí, které jsou připojeny ke zprávě, kterou chceme poslat, se příjemce dozví, že zprávu jsme posílali opravdu my a zpráva je hodnověrná. [1]

K dalším bezpečnostním službám patří nepopíratelnost, která zajišťuje, aby daný subjekt nemohl později popřít to, co předtím vykonal. [1]

1.4 Symetrická kryptografie

Jedná se o šifrovací algoritmy, které používají ten samý klíč pro šifrování i dešifrování. Mezi výhody této metody patří nízká výpočetní náročnost a rychlost, z toho důvodu je její použití vhodné k šifrování většího množství dat. Na druhou stranu je zde velká nevýhoda v podobě nutnosti předat klíč příjemci zprávy zabezpečeným kanálem. Vzniká zde tedy problém, který se řeší např. zkombinováním symetrické a asymetrické kryptografie. Symetrické šifry mohou být blokové nebo proudové. [5]



Obrázek 1 - Princip symetrického šifrování

1.4.1 Blokové šifry

Symetrická šifra, která rozděluje původní text do bloků pevně stanovené délky, např. 256 bitů, se nazývá bloková šifra. Pokud jsou data delší, je třeba je rozdělit do více bloků. Avšak může dojít k tomu, že poslední blok bude kratší, než je pevně stanovená

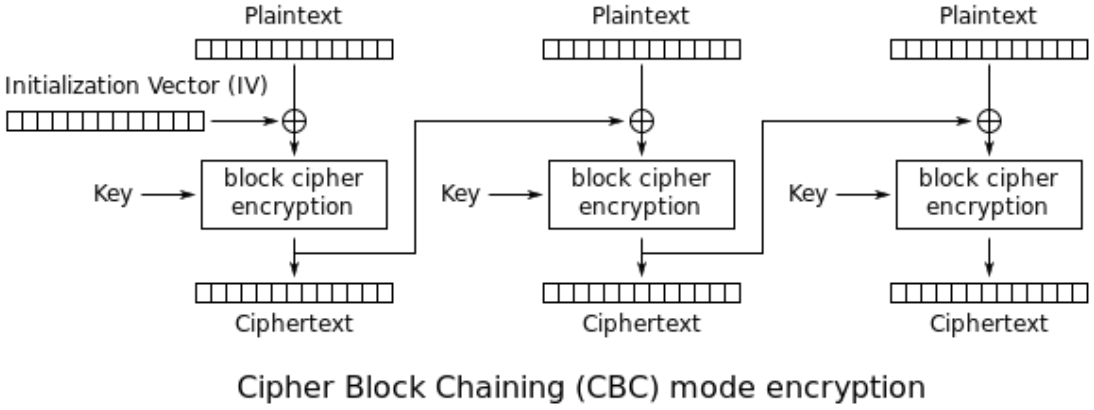
délka. Tento problém se řeší výplní (tzv. padding). K dispozici je několik algoritmů, které umožňují vložit výplň do bloku, např. vyplnění nulami. Složitější algoritmy nevkládají výplň jen do posledního bloku, ale do všech tak, aby byl text co nejvíce pozměněn. Příkladem jednoho z nejpoužívanějších algoritmů pro výplň je standard PKCS#7. U blokových šifer se dále používají různé druhy režimu provozu, které upravují fungování algoritmu. Takovým úpravám se říká módy. [5]

Mód ECB

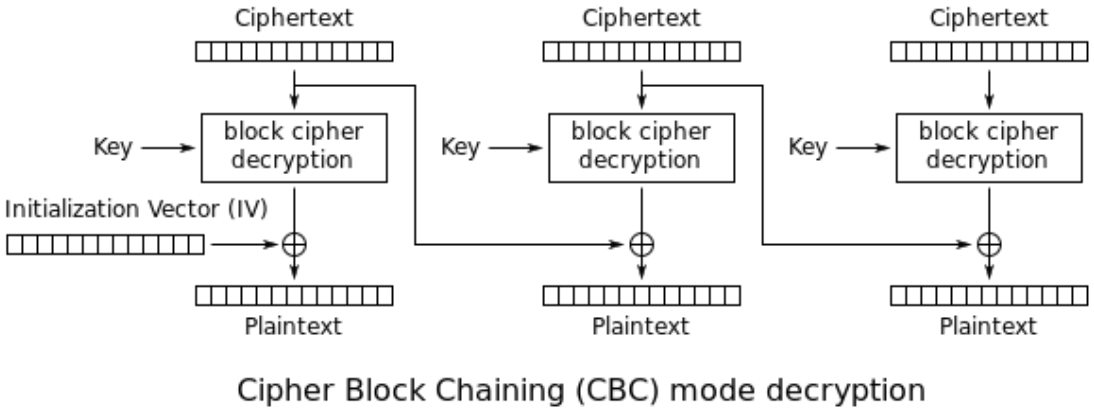
Tím úplně nejjednodušším módem je mód ECB. Při použití tohoto módu se všechny bloky budou šifrovat stejně. A protože bloky jsou poměrně malé, tak existuje možnost nalezení vzoru mezi zašifrovanými bloky a původní zprávou. Druhým problémem je nebezpečí porušení integrity zprávy, tzn. přidání nebo zaměnění bloku. Proto se použití ECB nedoporučuje. [5]

Mód CBC

Tento mód vytvořila firma IBM v roce 1976 a nahradil zranitelný ECB. Mód CBC ještě před šifrováním vezme blok původního textu (plaintext) a ten XORuje blokem předchozím. Tím jsou na sobě bloky naprosto závislé, a tudíž nemůže dojít k nahrazení jednoho bloku, protože by se musel nejprve dešifrovat předchozí blok. První blok se XORuje náhodně vygenerovaných blokem, který se přidá ke zprávě jako nultý blok, neboli inicializační vektor (IV). CBC je nejpoužívanějším módem blokových šifer. [5]



Obrázek 2 - Šifrování v CBC módu



Obrázek 3 - Dešifrování v CBC módu

Módy CFB a OFB

Jedná se operační módy, které převádí blokovou šifru na proudovou. Používá se náhodného inicializačního vektoru k nastavení odpovídajícího konečného automatu do náhodné polohy. Konečný automat pracuje tak, že vzniklé heslo (v módu OFB) nebo vzniklý šifrový text (v módu CFB) jsou vedeny na vstup blokové šifry a jejich zašifrováním je produkován následující blok hesla. OFB má vlastnost čisté (synchronní) proudové šifry, neboť heslo je generováno zcela autonomně bez vlivu otevřeného a šifrového textu. CFB je kombinací vlastností CBC a proudové šifry. [5]

1.4.2 Proudová šifra

Symetrická šifra, u které se vstupní datový tok zkombinuje s pseudonáhodným proudem bitů vytvořeným ze šifrovacího klíče a šifrovacího algoritmu. Výsledkem je zašifrovaný proud dat. Proudové šifry jsou rychlejší než blokové. Dělíme je na dvě základní skupiny. Synchronní proudové šifry a proudové šifry s vlastní synchronizací (asynchronní). Příkladem proudové šifry je např. RC4 nebo FISH. [1]

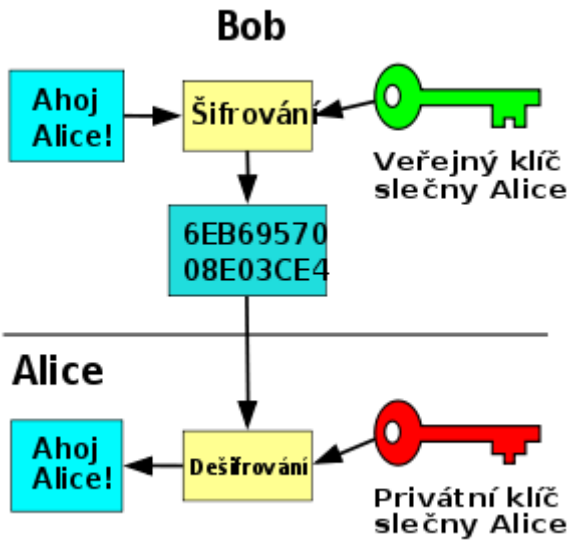
Synchronní šifra nevyužívá k šifrování text, data šifruje jen pomocí klíče a stavu, ve kterém se nachází. Pokud se při přenosu ztratí jediný bit, nebo nějaký přibude, dešifrování do původní podoby nebude možné.

Asynchronní šifry využívají k šifrování i dešifrování krom klíče i předchozích *n* bitů šifrovaného textu. Tím se zvyšuje bezpečnost, ale při ztrátě jednoho bitu se následujících *n* bitů zprávy dešifruje špatně. Po těchto *n* bitech se následující data dešifrují správně.

1.5 Asymetrická kryptografie

Jedná se o šifrovací algoritmus využívající pro šifrování i dešifrování jiný klíč. Algoritmus je založen na principu, že dvě čísla snadno vynásobím, ale rozklad součinu

na činitele je velmi složitý. Klíče jsou tedy dva a v páru, klíč veřejný, kterým se zpráva šifruje a klíč soukromý, kterým se zpráva dešifruje. V praxi si tedy příjemce (např. Alice) zprávy vygeneruje dvojici klíčů, veřejný předá odesílateli (např. Bob), ten tímto veřejným klíčem zašifruje zprávu a pošle ji Alici. Alice poté použije svůj soukromý klíč k dešifrování zprávy. Výhoda je zřejmá. Odpadá nebezpečí při posílání klíčů, jelikož se posílá pouze veřejný klíč, který si může kdokoliv zkopírovat a bez soukromého klíče je bezcenný. Zašifrovaná zpráva se dá dešifrovat pouze soukromým klíčem a ten se v komunikaci nevyskytuje. Nevýhodou je velká výpočetní náročnost, daleko větší než u blokových šifer. Asymetrické šifry jsou 10x až 100x pomalejší než šifry symetrické. [7]



Obrázek 4 - Princip asymetrického šifrování

RSA

Název RSA je složení prvních písmen jmen autorů Rivesta, Shamira a Adlemana. Jedná se o první šifrovací algoritmus asymetrické kryptografie a patří mezi nejpoužívanější šifry. Také je to první algoritmus vhodný k šifrování i podepisování. Pracuje s předpokladem, že faktorizace (rozklad čísla na součin prvočísel) je extrémně obtížná úloha a žádný algoritmus, který to dokáže v rozumném čase, zatím není znám. Oproti tomu vynásobení dvou prvočísel je elementární úloha. Algoritmus se využívá při výměně klíčů (hybridní kryptografie – kombinace symetrické a asymetrické kryptografie) a při digitálním podpisu. [8]

ElGamal

ElGamal systém je asymetrický kryptografický systém, založený na problému diskrétního logaritmu. Obsahuje algoritmus pro šifrování i algoritmus pro podepisování. Šifrovací algoritmus je podobný Diffieho-Helmanově výměně klíčů. Hlavní nevýhoda je ta, že šifrovaná data jsou 2x delší než původní text. Na druhou stranu je zase možnost mít vždy jiný šifrovaný text při stejných datech. Šifra se používá při výměně klíčů. [1]

DSA

Digital Signature Algorithm (algoritmus digitálního podpisu) byl v roce 1991 přijat Americkým úřadem pro standardizaci jako nový standard pro digitální podpis.

Algoritmus byl v dalších letech několikrát upravován. DSA je založen na problému výpočtu diskrétního algoritmu, stejně jako ElGamal. Využívá se stejně jako ostatní asymetrické šifry pro podepisování. Široké využití má v OpenSSL a GNUPG. [9]

Diffie-Hellman protokol

Byl vynalezen v roce 1976 Whitfieldem Diffiem a Martinem Hellmanem. Jedná se o kryptografický protokol umožňující skrze nezabezpečený kanál vytvořit šifrované spojení mezi komunikujícími stranami bez předchozího dohodnutí šifrovacího klíče. Využívá se zde jednocestných funkcí. Výsledkem algoritmu je vytvoření symetrického šifrovacího klíče, který může být následně použit pro šifrování další komunikace. Klíč konstruují všechny komunikující strany a nikdy se neposílá v otevřené formě. Avšak protokol neumožňuje autentizaci a tudíž je bezbranný proti útoku Man-in-the-Middle, ale je odolný proti odposlechu třetí nežádoucí strany. Počet účastníků není nikterak omezen. [10]

1.5.1 Digitální podpis

Digitální podpis je jednou ze stěžejních aplikačních oblastí kryptografie. Využívá asymetrické kryptografie k podpisu posílané zprávy. Odesílatel zprávy vytvoří pomocí hashe zprávy a svého soukromého klíče digitální podpis, který připojí k zašifrované zprávě a pošle všechno příjemci. Příjemce zprávy si dešifruje zprávu a následně na ni aplikuje stejný hashovací algoritmus. Potom vezme digitální podpis a rozluští jej za pomocí veřejného klíče odesílatele. Pokud se tato rozluštěná zpráva shoduje s hashem vytvořeným z dešifrované poslané zprávy, máme jistotu, že zprávu poslal opravdu její autor. [7] [11] [12]

1.5.2 Digitální certifikát

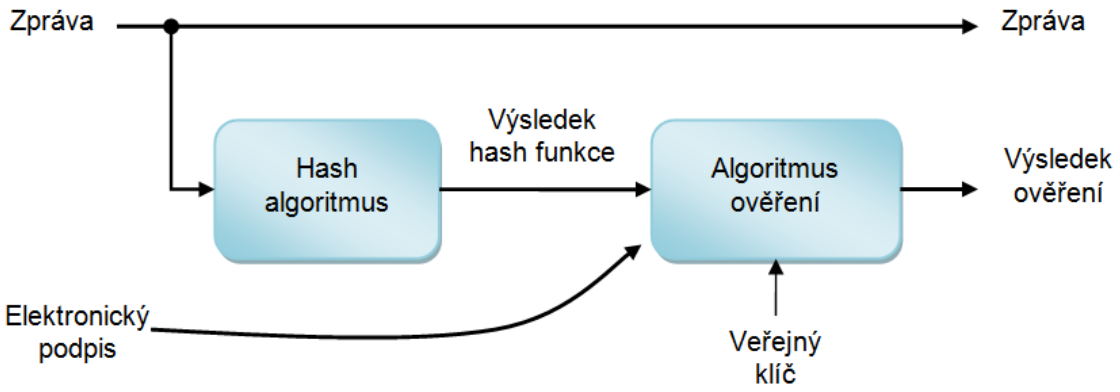
Pokud probíhá zabezpečená komunikace mezi dvěma stranami pomocí protokolu HTTPS či VPN, používá se digitální certifikát. Certifikát je elektronickým průkazem totožnosti komunikujícího subjektu a obsahuje: veřejný klíč, jméno a další údaje o osobě, datum počátku platnosti, datum ukončení platnosti, jméno certifikační autority, která certifikát vydala, sériové číslo a některé další informace. Pokud důvěřujeme certifikační autoritě (certifikační autorita řeší správu a distribuci veřejných klíčů, např. VeriSign, Comodo, apod.), důvěřujeme i majiteli veřejného klíče. (Na základě principu přenosu důvěry je možné důvěřovat neznámým certifikátům, které jsou podepsány důvěryhodnou certifikační autoritou). [11]

1.5.3 Autentizace dat

Autentizace dat přímo souvisí s ověřováním integrity. Obvykle je ověření integrity zprávy jedním z kroků, který je třeba udělat, abychom dokázali autentičnost dat či zprávy a tím určili autora nebo odesílatele. Používá se např. hashování zprávy, kdy se vypočítá hash zprávy a ten se pošle jiným kanálem. Příjemce si hash spočítá taky a porovná s číslem od odesílatele. V případě že by se neshodovaly, byla porušena integrita zprávy. [1]

1.6 Hash funkce

Hashovací funkce je algoritmus pro převod vstupních bitů na posloupnost pevné délky *n* bitů. Výstup hashovací funkce se označuje jako hash či otisk. Funkce se používá k zaručení integrity zprávy či autentizace a je jednosměrná. To znamená, že pro daný hash *m* je obtížné spočítat *x* takové, že $h(x)=m$. Na námi vytvořené heslo (např. v přihlašovacím systému) se použije hashovací funkce a výsledný otisk se uloží do databáze. Při přihlašování se vložené heslo opět prožene hashovací funkcí a pokud otisky souhlasí, s největší pravděpodobností souhlasí i samotná hesla a je umožněn přístup. Může se ale vyskytnout situace, kdy zadané heslo má stejný otisk jako heslo v databázi přičemž obě hesla jsou různá. Takové jevu se říká kolize, tedy nalezení dvojice vstupů (x, y), $x \neq y$ pro které platí $h(x)=h(y)$. Nalezení kolize pro hashovací funkci je principem útoku na tuto kryptografickou metodu a v dnešní době jsou známy způsoby prolomení některých starších funkcí jako je MD5. [1]



Obrázek 5 - Proces ověření elektronického podpisu

1.7 Autentizační kód zprávy

Autentizační kód zprávy (MAC, Message Authentication Code) je kód zajišťující integritu dat. Tento zabezpečovací kód autentizuje původ zprávy a zajišťuje obranu proti náhodným chybám při přenosu i úmyslným změnám. Pokud ke zprávě připojíme MAC, příjemce může ověřit, že data pochází od toho, kdo zná klíč. MAC je krátký kód, který dostaneme zpracováním zprávy s tajným autentizačním klíčem. MAC kód vypočítáme pomocí blokové šifry nebo pomocí hashovaní funkce a dělí se tedy na dva typy. [13]

1.7.1 HMAC

Typ autentizačního kódu zprávy počítané s použitím kryptografické hashovací funkce v kombinaci s tajným šifrovacím klíčem se nazývá HMAC (z angl. Keyed-hash Message Authentication Code). Šifra používá k šifrování hashovací algoritmus. HMAC je používán jednak na ověření datové integrity a jednak na autentizaci zprávy. Na výpočet je možné použít libovolnou kryptografickou hashovací funkci jako např. MD5

či SHA Kryptografická síla kódu HMAC přímo závisí na síle hashovací funkce, velikosti a kvalitě klíče a velikosti výstupu hashovací funkce v bitech. [14]

1.7.2 CMAC

Pokud se MAC kód spočítá pomocí symetrické blokové šifry, vzniká CBC-MAC. Tato verze kódu vznikla jako první, ale jelikož byl kód bezpečný pouze pro zprávy fixní délky, byl nahrazen novější verzí s názvem XCBC-MAC, který vytvořili pro organizaci NIST pánové Black and Rogaway. Tento kód využíval 3 autentizační klíče. Iwata a Kurosawa kód dále zdokonalili tak, že nadále využíval pouze jeden autentizační kód a tuto verzi ve své práci pojmenovali One-Key CBC-MAC neboli OMAC. Kód dále analyzovali a upravili do konečné podoby s názvem OMAC1, často uváděný jako CMAC. [13]

2 BEZPEČNOSTNÍ MECHANISMY PROTOKOLŮ

Zabezpečovací protokoly využívají spoustu algoritmů a mechanismů, které zvyšují bezpečnost při ustanovení spojení mezi dvěma stranami a následném přenosu dat. Bezesporu mezi tyto zabezpečení patří i ověření entit neboli autentizace, zajištění důvěrnosti dat šifrováním a integrita dat. Tyto zabezpečovací metody byly již popsány v kapitole 1.3 Bezpečnostní služby, a proto nyní nebudou znovu rozebírány.

2.1 Protokol ustanovení klíče

Key-agreement protokol neboli ustanovení klíčů je proces, při kterém dvě nebo více komunikujících stran dojdou vzájemnou dohodou ke společnému šifrovacímu klíči. Při správném provedení je vyloučeno, aby nežádoucí třetí strany klíč získaly. Taktéž by protokol neměl odposlouchávajícím stranám odhalit, že byl ustanoven šifrovací klíč. K tomuto je možné použít asymetrické šifrování, kde si nejprve obě strany ustanoví svůj privátní a veřejný klíč, veřejný klíč pošlou druhé straně a poté si už v šifrované podobě předají šifrovací klíč pro symetrickou šifru. Nebo je možné použít Diffie-Hellmanovu výměnu klíčů, která umožňuje dohodnutí klíče přímo pro symetrické šifrování. [10]

2.2 Pořadové číslo

Jedná se o číslování posílaných zpráv. Údaj je společný pro obě dvě strany a při každém odeslání zprávy v rámci jednoho spojení se k číslu přičte předem dohodnutá hodnota (např jednička). Pokud jedna strana obdrží zprávu s hodnotou nižší, než je očekávaná hodnota, jedná se zopakovanou starou zprávu. Při uzavření nového spojení nebo dosažení maximální hodnoty se hodnota resetuje a začne se zase od začátku. Tím se zabrání útočníkovi duplikovat a opětovně posílat pakety a tím provést tzv. Replay Attack. [15] [16]

2.3 Kontrolní součet

Kontrolní součet se přidává k posílané zprávě a jedná se o doplňující informaci, která slouží k ověření, zda jsou posílaná data úplná a nezměněna. Je to výsledek předem určené operace, například hashovací funkce nebo MAC kódu. Při posílání dat spočítá odesílatel kontrolní součet a tento přiloží ke zprávě. Příjemce taktéž provede kontrolní součet přijatých dat a při shodě je zaručeno, že data jsou úplná a nepozměněna. Nejedná se jen o ochranu před útokem, ale jedná se také o zjištění chyb na přenosovém kanále.

2.4 Síťový tunel

Samotné tunelování nezašifruje data, nýbrž vytvoří přímé síťové spojení mezi dvěma body. Tím poskytne zabezpečenou komunikaci skrze nezabezpečenou síť. Tunelování se provádí zapouzdřením datového prvku (PDU-protocol data unit) přenášeného protokolu do PDU jiného protokolu, který leží na stejné nebo vyšší síťové vrstvě. [17]

2.5 Nonce

Nonce je dlouhé náhodné číslo (obvykle více než 16 bajtů dlouhé), které jedna strana vždy přidává do svých zpráv k přenášeným datům. Pro každou zprávu musí být nonce znovu vygenerované a nesmí se opakovat. Tím se zajistí, že se nepošlou dvě stejné zprávy a tudíž se nevytvoří dva stejné MAC kódy. Druhá strana musí pokaždé zkontrolovat, zdali v minulosti neobdržela tutéž zprávu se shodnou nonce. Někdy se nonce vytváří ze dvou částí. Jedna část obsahuje náhodně vygenerované číslo a druhá část obsahuje čas a datum, který je samozřejmě neopakovatelný. Nonce se může vytvářet i jinak, například z kryptografického klíče odesílatele a příjemce apod. [16]

3 MOŽNOSTI PROLOMENÍ ZABEZPEČOVACÍCH PROTOKOLŮ

3.1 Útok hrubou silou

Útok hrubou silou je snaha o prolomení šifry bez znalosti dešifrovacího klíče a to systematickým zkoušením všech možných kombinací nebo omezené podmnožiny. Útok je poměrně rozšířený, avšak nalezení složitějšího hesla trvá velmi dlouho. U bezpečně dlouhého hesla, kde je použita diakritika, číslice a speciální znaky se dostáváme k době v řádech tisíců let. Při použití větší výpočetní kapacity se samozřejmě doba snižuje.

3.2 Slovníkový útok

Tento druh útoku je pravděpodobně tím nejjednodušším. Spočívá v prostém zkoušení hesel ze seznamu (slovníku) dokud není heslo nalezeno nebo se nedojde na konec seznamu. Pokud uživatel použije diakritiku či speciální znaky tak je podstatně snížena pravděpodobnost nalezení hesla. Pokud je účelem prolomit heslo v podobě hashe, je možné si slovník „přepočítat“ hashovaní funkcí a následně použít slovník hashů (viz Rainbow table).

3.3 Replay attack

Tento útok je formou síťového útoku, kdy útočník záměrně zpozdí paket nebo jej odchytí a následně pošle znovu. Útočníkem nemusí být jen třetí strana ale též odesílatel zprávy, který chce, aby protistrana vykonala určitý příkaz vícekrát. [16]

3.4 Narozeninový útok

Útok vychází z Narozeninového paradoxu z teorie pravděpodobnosti (je šance více než 50% že 2 lidi z nejméně 23 lidí budou mít ve stejný den narozeniny). Jedná se o snahu nalezení kolize v kryptografické hashovací funkci. Pro tento účel jsou v tomto útoku náhodně vybírány vstupní hodnoty a následně vypočítávána výstupní hodnota funkce. [18]

3.5 Útok Rainbow table

Stejně jako narozeninový útok je i Rainbow Table útokem používaným k prolomení hashe. Tento útok využívá předem vypočtené hodnoty určené k usnadnění prolomení hashovací funkce útočníkem. Útočník takto může získat hesla, která jsou typicky uložena v podobě hashe. Rainbox table jsou jakýmsi kompromisem mezi výpočetní

náročností útoku hrubou silou a prostorovou náročností předem vypočtených tabulek pro reverzní funkci k hashovací funkci (jelikož ty zabírají velké množství paměti). [19]

3.6 Lineární kryptoanalýza

Lineární kryptografie se nejčastěji používá k prolomení blokových šifer. Využívá vysokou pravděpodobnost výskytu lineárních vyjádření zahrnujících bity otevřeného textu, bity šifrovaného textu a bity podklíčů pro danou rundu. Algoritmus hledá lineární závislosti mezi vstupy a výstupy s-boxů. [20]

3.7 Diferenciální kryptoanalýza

Diferenciální kryptoanalýza taktéž využívá vysokou pravděpodobnost, ale na rozdíl od lineární kryptoanalýzy hledá rozdíl mezi otevřeným textem a poslední rundou šifry. [20]

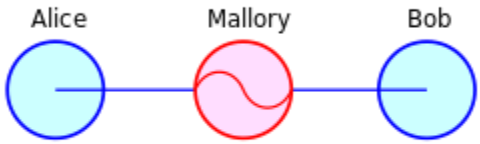
3.8 Útok postranním kanálem

Tento druh útoku nevyužívá prolomení šifry přímo, ale zaměřuje se na fyzické projevy kryptografického procesu. Zvýšení teploty procesoru, odběr proudu, trvání algoritmu a jiné parametry lze využít ke zjištění použitého klíče, pinu nebo jaký algoritmus byl v prvé řadě použit. Útok se dělí na několik analýz podle druhu fyzického projevu: časová, odběrová, elektromagnetická. Dalším způsobem je útok zaváděním chyb. Při tomto útoku se snaží útočník zavést do průběhu výpočtu chyby a získat tak nějaké informace o systému. Pokud například bleskem skrze mikroskop ozáří určitou část RAM paměti, může dojít při instrukcích podmíněného větvení k chybě a na terminál se zašle i tajný klíč a útočník jej takto získá. [21]

3.9 Útok Man-in-the-middle

Pokud spolu chtějí 2 strany (např Alice a Bob) komunikovat v šifrované podobě, vědí, že si mohou snadno vyměnit klíče díky asymetrickým šifrám nebo DH výměně klíčů. Ale nemohou vědět, že komunikují jeden s druhým, jelikož mezi nimi může být útočník a odchytávat komunikaci.

1) Alice pošle svůj veřejný klíč.
2) Mallory ho zachytí a nahradí ho svým veřejným klíčem.
3) Bob pošle svůj veřejný klíč.
4) Mallory ho zachytí a nahradí ho svým veřejným klíčem.



Obrázek 6 - Zobrazení útoku Prostředník

Nyní si Alice i Bob myslí, že mají veřejný klíč toho druhého. Místo toho ale oba mají klíč pocházející od Mallory. Kdykoliv něco šifrovaného pošlou, Mallory to zachytí, dešifruje svým tajným klíčem, přečte si to (popř. zamění) zašifruje veřejným klíčem pro druhou stranu a odešle. Příjemce (Bob) nic nepozná, dokonce ani v případě, že by zpráva byla Alicí digitálně podepsána. Mallory to může udělat totiž stejně tak.

Pokud se Alice a Bob rozhodnou použít symetrickou šifru a k dohodě šifrovacího klíče DH výměnu klíčů, Mallory může opět odchytávat komunikaci a jejich dohodnutý klíč získat. Poté by jen odchytával komunikaci a pomocí klíče ji snadno dešifroval anebo zprávy pozměnil k vlastnímu prospěchu. [22]

4 ZABEZPEČOVACÍ PROTOKOLY

Pokud chceme použít šifrovací algoritmy popsané v předchozí části, musíme určit, jakým způsobem se algoritmy budou používat. Těmto pravidlům se říká protokoly, a pokud mluvíme o zabezpečení výměny informací, jedná se o zabezpečovací či kryptografické protokoly. Tyto protokoly se používají pro bezpečný přenos dat na aplikační vrstvě (TLS/SSL) anebo už na vrstvě síťové (IPSec). Protokol by měl umožňovat získávání dat či provádět různé operace jen v předem určených mezích a jen tomu, komu to je určeno. Také pokud kdokoliv bude chtít získat data dešifrováním, mělo by to být co nejsložitější a mělo by existovat co nejméně záznamů o původním (otevřeném) textu. Kryptografické protokoly využívají několik zabezpečovacích prvků.

4.1 Protokol IPSec

IPSec rozšiřuje standardní IP protokol o bezpečnostní algoritmus, který je založen na autentizaci a šifrování každého IP paketu. Jelikož se jedná o nástavbu IP protokolu, který je na síťové vrstvě síťového modelu ISO/OSI, je toto zabezpečení transparentní pro síťové aplikace (na rozdíl od vyšších bezpečnostních protokolů jako jsou TLS a SSH). IPSec obsahuje protokoly pro vytvoření vzájemné autentizace mezi dvěma agenty na začátku spojení a dohodnutí šifrovacího klíče mezi stranami. Komunikace může probíhat mezi dvěma hosty, mezi dvěma sítěmi (jejich branami) nebo mezi bránou a hostem. [17]

4.2 Protokol SSL

Tento protokol se nejčastěji využívá pro zabezpečení komunikace s webovými servery pomocí HTTPS. SSL (spolu s TLS) se také používá k vytvoření zabezpečeného spojení při stahování emailů, ke zpracování citlivých osobních údajů či k on-line obchodům, kde se vkládají údaje platebních karet. Toho se docílí vytvořením autentizovaného a šifrovaného spojení mezi serverem a klientem. Secure Sockets Layer se dělí na 2 podprotokoly, které leží na různých síťových vrstvách a mají různé funkce.

- SSL Record protocol je zodpovědný za převzetí dat z výše postavených protokolů, šifrování, kompresi a autentizaci. Leží pod aplikační vrstvou.
- SSL Handshake protokol leží nad transportní vrstvou a má za úkol určit parametry přenosu: šifrovací klíče, mód šifry (blokový, proudový), kompresní algoritmus a hashovací funkci která bude použita. K ustanovení spojení je možné použít více druhů algoritmů. Pro výměnu klíčů Diffie-Hellman nebo některou asymetrickou šifru (RSA, DSA), pro symetrické šifrování IDEA, triple DES, AES nebo jiné a pro jednocestné hashovací funkce je možné použít MD5 nebo bezpečnější SHA-1.

Původní verze SSL 1.0 měla vážné nedostatky, chyběly zabezpečovací mechanismy jako kontrolní součet a pořadové číslo a protokol nezajišťoval autentizaci. Nyní se používá verze 3.0, která původní verzi dalece předčí. Následovníkem SSL je protokol Transport Layer Security (TLS). [23]

4.3 Protokol TLS

Cílem protokolu Transport Layer Socket je zajišťovat důvěrnost a integritu mezi komunikujícími aplikacemi. Je nástupcem SSL a rozdíly mezi TLS 1.0 a SSL 3.0 jsou nepatrné. Protokol zajišťuje ochranu proti útoku Man-in-the-middle a pomocí MAC kódů a principu certifikačních autorit dokáže spolehlivě provést autentizaci. Jednotlivé zprávy jsou opatřeny pořadovým číslem a to je přidáno do MAC kódu což zvyšuje zabezpečení. V různých aplikacích se většinou setkáme se zabezpečením v podobě TLS/SSL. [24] [25]

4.4 Protokol SSH

Dalším často používaným protokolem pro zabezpečení komunikace skrze nedůvěryhodnou síť je Secure Shell. Používá se v sítích s modelem TCP/IP. SSH je náhrada za telnet a jiné nezabezpečené vzdálené shelly, které jsou náchylné na odposlech a následné zneužití jelikož heslo posílají v nezabezpečené podobě. SSH neposkytuje síťové tunelování, ale umožňuje zabezpečené zprostředkování přístupu k příkazovému řádku a taktéž kopírování souborů. Stejně jako SSL/TSL poskytuje autentizaci, ale na rozdíl od nich poskytuje autentizaci obou komunikujících stran. Rovněž zajišťuje integritu a transparentní šifrování přenášených dat a volitelnou bezeztrátovou kompresi. [26]

4.5 Protokol Kerberos

Tento protokol byl vyvinut na univerzitě MIT (Massachusetts Institute of Technology) za účelem ochrany síťových služeb v rámci projektu. Kerberos je síťovým autentizačním protokolem, který umožní komukoliv, kdo komunikuje v nezabezpečené síti bezpečně prokázat svou identitu jiné straně. Taktéž zabraňuje odposlechu nebo zopakování takovéhle komunikace a zaručuje integritu dat. Autentizace, kterou poskytuje je vzájemná, tedy i klient i server se prokážou tomu druhému. Kerberos používá symetrickou kryptografii a potřebuje důvěryhodnou třetí stranu. Jednou z nevýhod kerberosu je nutnost přesné synchronizace času mezi klientem a serverem (s odchylkou 5min). K tomu se používá NTP (Network Time Protocol) démon. [27]

5 PROTOKOLY ŘÍDICÍCH SYSTÉMŮ

5.1 Systémy Scada

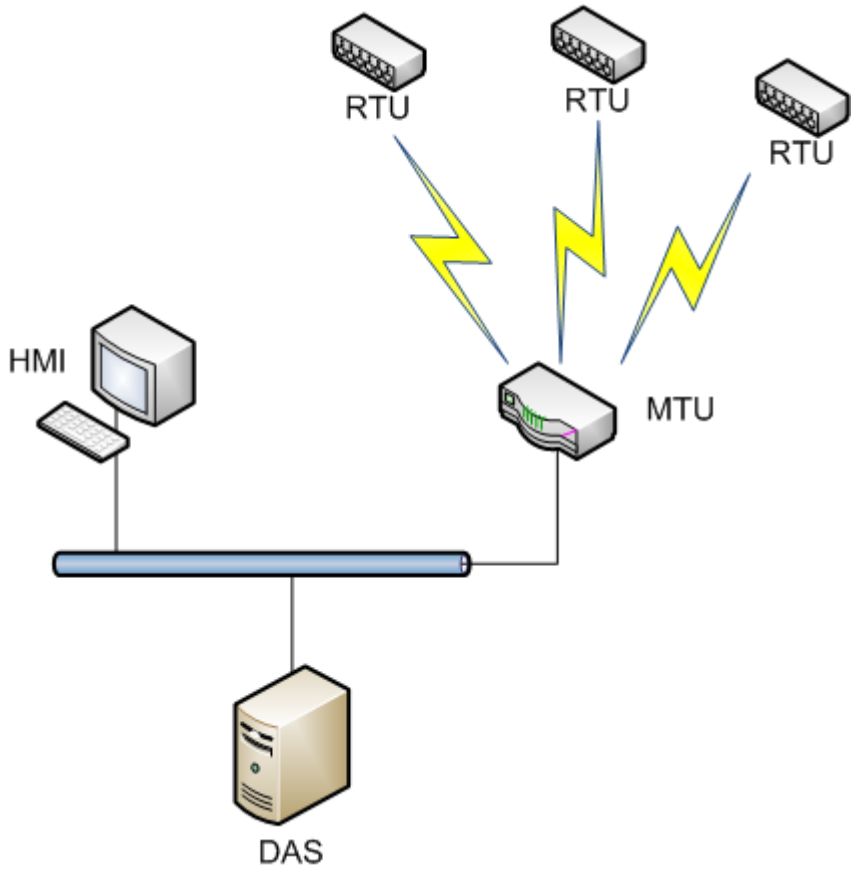
Název SCADA vychází z anglického supervisory control and data acquisition, tedy dispečerské řízení a sběr dat v reálném čase. Jedná se o centralizovaný systém. Funkce, které může vykonávat host (tedy koncové zařízení) jsou omezeny pouze na ty nejzákladnější. SCADA systémy se používají k monitoringu, sběru dat a poskytnutí rozhraní pro kontrolu ovládání zařízení v průmyslu. Systém dokáže mnoho funkcí, od shromažďování informací a jejich transport na centrálu přes reakci na různé události až po spuštění alarmu a zaslání zprávy. SCADA může být jednoduchá, například ke sledování počasí nebo životního prostředí, anebo velmi složitá a komplexní k použití v jaderných elektrárnách nebo k ovládání vodních systémů. Systém je vhodný pro použití v komplexech, které jsou rozprostřeny do velké oblasti. Používá se především ve výrobních zařízeních, jelikož tam je velmi důležitá regulace celého procesu a různí se dle zaměření podnikání. Dále v energetice či potravinářském průmyslu.

Remote Terminal Unit (RTU): jsou jednotky přímo připojené k senzorům a převádí jejich signály na digitální data. Zařízení shromažďuje data ze senzorů, dokud si je MTU nevyžádá. RTU je definováno jako vzdálené komunikační zařízení uvnitř systému.

Master Terminal Unit (MTU): MTU je srdcem SCADA systému, je umístěno v hlavním monitorovacím centru a zahajuje komunikaci se vzdálenými jednotkami a rozhraními.

Data Acquisition System (DAS): DAS sbírá informace od MTU, generuje a ukládá alerty, které mohou mít dopad na systém a tudíž vyžadují pozornost od operátora.

Human Machine Interface (HMI): HMI je rozhraní, přes které se operátor přihlašuje a může pak monitorovat události a informace nasbírané systémem DAS. [28]



Obrázek 7 - Architektura systému SCADA

5.1.1 Protokol SSCP

Secure SCADA Communications Protocol (SSCP) je navržen k zabezpečení sériového ovládacího komunikačního systému. Protokol zapouzdřuje zprávu i s hlavičkou. K zajištění autentizace při ustanovení spojení jsou použity dva symetrické klíče. Kryptografický klíč původní zprávy je použit k vytvoření nonce, což zajišťuje unikátní hodnotu paketu. Kryptografický klíč příjemce zprávy je použitý kryptografickou funkcí hash k vytvoření HMAC založené na hlavičce, nonce a původní zprávě. Pro zajištění co nejlepší bezpečnosti se klíče vytváří mezi komunikujícími stranami metodou ustanovení klíčů Diffie-Hellman. [29]

5.1.2 Implementace ScadaSafe

ScadaSafe je implementace navrhovaných protokolů k šifrování a zajištění integrity zpráv systému SCADA na sériových linkách. Americká plynová asociace vyvinula a publikovala standard vhodný k použití v plynovém průmyslu ve Spojených státech. Tento standard doplnil starší verzi systému SCADA, je určen pro sériové linky s modulační rychlostí min 1200 baudů a poskytuje šifrovací algoritmy pro ochranu systému. Lze jej tedy aplikovat na systémy SCADA používány i v jiných průmyslových odvětvích. V průběhu let vycházely novější verze systému, které jej dále doplňovaly.

Nyní obsahuje protokol Modbus Rtu i Ascii, DNP3, podporuje Java implementaci na operách systémech Linux i Windows, funguje na síťovém modelu point-to-point a na sériové lince i při modulační rychlosti 9600 baudů. Pro zabezpečení přenosu zpráv se mohou použít protokoly SHA1, MAC s blokovou šifrou AES/CTR nebo bezpečnější SHA256 MAC a časové razítka. [30]

5.2 Protokol Modbus

Protokol Modbus pochází z roku 1979, kdy byl poprvé představen firmou Modicon. Je to jeden z prvních komunikačních protokolů určených pro výměnu dat mezi stranami Server-Klient. Jedná se o model počítačové komunikace Master-Slave, kde Server (Master) posílá dotazy Klientu (Slave) a ten na ně odpovídá a reaguje. Tento protokol byl navržen tak, aby byl nezávislý na fyzické vrstvě a dokázal pracovat v síťovém prostředí. Jeden server tak může komunikovat až s 255 klienty. Modbus je velice rozšířeným protokolem jelikož je jednoduchý, transparentní a dá se snadno implementovat. [31]

5.2.1 Struktura protokolu

Struktura zprávy v protokolu Modbus je definována nezávisle na typu komunikační vrstvy. Podle typu sítě, na které se používá, se PDU (Protocol Data Unit) rozšíří o další dodatečné části a vytvoří tak ADU (Application Data Unit) – zprávu na aplikační úrovni. Při komunikaci se používá kód funkce, který udává, jaký druh operace má klient vykonat. Používají se kódy v rozsahu 1 až 255 přičemž 128 až 255 slouží pro oznámení chyby. Některé kódy funkcí mají obsažen i kód podfunkce, který upřesňuje požadovanou operaci. Obsah datové části zprávy poslané serverem může sloužit klientovi k uskutečnění operace určené kódem funkce. Obsahem může být například adresa a počet vstupů, které má klient přečíst nebo hodnota registrů, které má klient zapsat. Datovou část používá také klient, kde např. vyplní požadované informace. Některé funkce nepotřebují k provedení další dodatečná data a v tom případě může datová část ve zprávě úplně chybět. Modbus se nejčastěji používá na sériové lince. Zde běží ve formátu RTU nebo ASCII a tedy rámec tvoří buďto bity nebo znaky. Po vynalezení Ethernetu vznikl Modbus TCP a ten může fungovat i skrze síť a na jiných fyzických linkách než jen sériové. [31]

MODBUS RTU

První verze Modbusu, která je i nejčastěji využívaná. Vysílání zprávy musí být souvislé a mezery mezi znaky nesmí být delší než 1,5 znaku. Začátek i konec zprávy se identifikuje pomlkou o délce nejméně 3,5 znaku. Protokol je rychlý a spolehlivý s nízkými nároky na připojená zařízení.

Start 28b	Address 8b	Function 8b	Data n*8b	Checksum 16b	End 28b

Obrázek 8 - Rámec protokolu Modbus RTU

MODBUS ASCII

Tento režim je podobný jako RTU, ale rámce přenáší znaky místo bitů. Každý byte se posílá jako dvojice ASCII znaků. Je tedy oproti režimu RTU pomalejší, ale umožňuje vysílat znaky s mezerami 1s. Začátek zprávy se indikuje znakem „:“ a konec zprávy dvojicí řídících znaků CR LF. Tato verze je málo využívaná, i když je čitelnější.

Start 1char	Address 2char	Function 2char	Data n*char	Checksum 2char	End 2char

Obrázek 9 – rámec protokolu Modbus ASCII

MODBUS TCP

Část rámce s názvem Unit Identifier je používaná zařízeními ke komunikaci mezi Modbus TCP a bránou Modbusu RTU. V takových případech obsahuje Unit Identifier adresu Klienta za bránou. Zařízení pracující v režimu Modbus TCP obvykle tento parametr ignorují.

Transaction Identifier 2B	Protocol Identifier 2B	Length Field 2B	Unit Identifier 1B	Function code 1B	Data bytes n*B

Obrázek 10 - rámec protokolu Modbus TCP

Start – alespoň 31/2 délky znaku ticha. Systém takto pozná začátek nového rámce.

Address – adresa klienta, se kterým chceme komunikovat.

Function – kód funkce, kterou chceme, aby klient vykonal. Jedná se například o čtení určitého registru od určitého místa, změnu nebo čtení I/O portu apod.

Data – datová část rámce je zaplněna dle funkce - buďto požadavek na klienta nebo poté jeho odpověď serveru.

Checksum – kontrolní součet slouží k ověření integrity.

End – stejný jako Start, označuje konec rámce.

Transaction Identifier – pro synchronizaci mezi zprávami na serveru a klienta

Protocol Identifier – 0 pro Modbus TCP

Length Field – počet zbývajících bytů v tomto rámci.

Unit Identifier – adresa slave (255 pokud není použit)

[31]

5.2.2 Bezpečnostní slabiny protokolu Modbus

Modbus je otevřený protokol, každý ví, jak vypadají jeho rámce a v základním řešení nezajišťuje šifrování nebo jiné zabezpečení. Verze RTU i ASCII obsahují kontrolní

součet, což je zajištění integrity proti záměrným nebo náhodným chybám. Neobsahuje ale šifrování dat nebo autentizaci. Námi navržený zabezpečovací protokol bude vycházet ze struktury Modbusu RTU ale bude řešit i otázku bezpečnosti a použijeme výše popsané zabezpečovací metody, aby byl protokol odolný vůči útokům, které byly taktéž popsány. [31]

6 DETEKCE VHODNÝCH SYMETRICKÝCH ŠIFER A KÓDŮ MAC PRO OMEZENÁ ZAŘÍZENÍ

6.1 Omezená zařízení

Pojmem omezená zařízení budeme rozumět mikrokontroléry s omezenou pamětí Flash i RAM a s nízkým výpočetním výkonem. Výkon je omezen nízkou proudovou spotřebou a cenou, čímž se tyto zařízení vyznačují. Tato zařízení jsou používána například v senzorových systémech, kde je nutné komunikaci šifrovat a zároveň chceme co nejmenší výrobek a tedy malý mikrokontrolér. Takový nebude mít velký výpočetní výkon ani rozsáhlou paměť. Zde je uvedeno několik často používaných mikrokontrolérů:

Výrobce	Označení	Velikost Flash paměti [kB]	Velikost RAM paměti [kB]	Procesor [MHz]
Atmel [32]	AVR ATtiny 85	8	0.5	20
Atmel [33]	AVR ATmega128L	128	8	16
Freescale Semiconductor [34]	MC9s08ac128	128	8	40
Texas Instruments [35]	MSP430F2272	32	1	16
Texas Instruments [36]	MSP430F1610	32	5	8

Tabulka 1 - Přehled mikrokontrolerů

6.2 Vybrané symetrické šifry

6.2.1 AES

V roce 2001 schválil Americký úřad pro standardizaci novou šifru, která měla nahradit již zastaralou šifru DES. Název tohoto algoritmu byl Rijndael, podle prvních písmen

autorů Rijmen a Daemen. Šifra Rijndael se tak stala novým standardem (Advanced Encryption Standard - AES). Šifra má blok o velikosti 128 bitů, ale klíč může být velikosti 128, 196 nebo 256 bitů. Útok hrubou silou v dnešní době není možný, šifra je také odolná proti lineárním a diferenciálním metodám kryptoanalýzy. Kromě bezpečnosti má i další výhody. Je velice rychlá, to ji činí vhodnou pro šifrování velkého objemu dat. Není náročná, má malé nároky na operační paměť i velikost kódu, proto se dá vhodně programovat na různých typech procesorů. [37]

6.2.2 RC5

Šifra RC5 (Ronald's Cipher) je pozoruhodná pro svou jednoduchost. Byla navržena Ronaldem Rivestem roku 1994. Šifra má variabilní délku bloku (32, 64 nebo 128 bitů), délku klíče (0 až 2048 bitů) a počet rund (0 až 255). Původní předpoklad volby parametrů byl 64 bitový blok, 128 bitový klíč a 12 rund. [38]

6.2.3 RC6

Další z kandidátů na AES vychází z RC5 a byla vytvořena stejnými autory. Šifra nepoužívá s-boxy ale cyklickou rotaci závislou na datech a klíč se volí o velikosti 16, 32 nebo 64 bitů. Uvádí se, že je šifra bezpečná pro počet 20 rund při průchodu Feistelovou sítí. Šifra je patentovaná organizací RSA. [38]

6.2.4 IDEA

International Data Encryption Algorithm („Mezinárodní algoritmus pro šifrování dat") je symetrická bloková šifra, která měla původně nahradit šifru DES. Pracuje po 64bitových blocích za použití 128bitového klíče a odvozuje velkou část své bezpečnosti ze střídání operací z různých grup – modulární sčítání a násobení a bitové nonekvivalence (XOR) – které jsou v jistém smyslu algebraicky neslučitelné. Šifra je bezpečná při použití 8,5 rund a při tomto nastavení nebylo zatím ohlášeno prolomení šifry. [37] [40]

6.2.5 XTEA

Šifra XTEA (eXtended TEA) je bloková šifra, za jejímž vytvořením stojí autoři David Wheeler a Rogera Needham a byla publikována v roce 1997. Šifra je určena pro použití v mikrokontrolérech s omezenými prostředky. Jedná se o přepracovanou a zdokonalenou původní verzi šifry TEA (Tiny Encryption Algorithm a stejně jako TEA používá i XTEA standardně 64 bitové bloky, šifrovací klíč o velikosti 128 bitů a 64 rund při průchodu Feistelovou sítí. Malé velikosti zdrojového kódu je dosaženo především díky použití základních instrukcí pro sčítání, posuvy a funkce XOR, namísto tabulkové substituce a bitové permutace známé z algoritmů DES, AES apod. [39] [41]

6.3 Analýza symetrických šifer

Zkoumáme tedy vybrané šifry z hlediska paměti a náročnosti na procesor. Je důležité také zohlednit délku klíče, bloku a počet rund neboť tyto parametry mají vliv na velikost paměti a náročnost procesoru a také samozřejmě bezpečnost. Některé šifry jsou bezpečné až při určitých parametrech a studie, ze kterých jsem přejal tyto výsledky, vždy používaly nastavení, pro které jsou šifry bezpečné. Následuje tabulka, která udává počáteční parametry testování symetrických šifer.

šifra	Délka klíče [Byte]	Počet rund [-]
AES [38]	16	10
RC5 [38]	16	18
RC6 [38]	16	20
IDEA [42]	16	-
XTEA [39]	16	32

Tabulka 2 - Počáteční parametry symetrických šifer

Vždy je uváděn počet taktů nutných k provedení zašifrování, velikost, kterou šifra zabírá na paměti flash a velikost kódu na paměti RAM, kterou šifra potřebuje k provedení. Pokud bychom chtěli použít rychlejší šifry, musíme počítat s větší paměťovou náročností, než jaké jsou skutečné parametry mikrokontrolerů, které uvažujeme a pro které testujeme zabezpečení komunikace. Téměř všechny symetrické šifry byly analyzovány na mikrokontroléru o taktovací frekvenci 8MHz, jen XTEA proběhla na 16 MHz mikrokontroléru. Šifry AES, RC5 a RC6 proběhly v CFB módu.

Šifra	Velikost bloku[B]	Počet taktů [-]	Trvání šifry[µs]	Velikost šifry[B]	Velikost na RAM [B]
AES [38]	16	3717	464,625	14716	92
RC5 [38]	8	8991	1123,875	1746	64
RC6 [38]	16	17671	2208,875	6312	100
IDEA [42]	64	6800	850	1992	208
XTEA [39]	64	6718	419,875	1160	-

Tabulka 3 - Výsledné hodnoty po spuštění symetrických šifer

Pro šifru RC5 bylo použito 18 rund kvůli bezpečnostním důvodům. Ostatní šifry používají standardní počet rund. U šifry IDEA nebyl uveden počet rund, předpokládáme tedy, že bylo dodrženo minimálně 8,5 rund pro splnění podmínky bezpečnosti šifry.

V prezentovaných studiích byly uvedeny pouze počty cyklů procesoru. Pro informativní účely je uvedena i doba trvání těchto šifer dle vzorce

$$T = \frac{1}{f} \text{ [s]}$$

6.4 Analýza hash funkcí a autentizačních kódů

Nyní se zaměříme na funkce Hash a MAC kódy. Jak už bylo popsáno výše, MAC kódy využívají buďto hash funkci a pak se nazývají HMAC anebo využívají blokové kódy a pak jim říkáme CMAC. Ze skupiny Hash funkcí byla vybrána šifra SHA256 a pro zástupce autentizacích kódů šifra CMAC-Present.

6.4.1 SHA

Secure Hash Algorithm je rozšířená hashovací funkce. Vytváří ze vstupních dat výstup (otisk) fixní délky a jeho hlavní vlastností je, že malá změna na vstupu vede k velké změně na výstupu, tj. k vytvoření zásadně odlišného otisku. SHA je soubor pěti podobných algoritmů: SHA-1, SHA-224, SHA-256, SHA-384 a SHA-512. Poslední čtyři varianty se souhrnně uvádějí jako SHA-2 a jejich čísla značí délku výstupního hashe v bitech. SHA-1 vytvoří obraz zprávy dlouhý 160 bitů. SHA se používá u několika různých protokolů a aplikací, včetně TLS a SSL, PGP, SSH a IPsec, ale i pro kontrolu integrity souborů nebo ukládání hesel. Je považována za nástupce hashovací funkce MD5. Následují výsledky simulace algoritmu SHA256 na 10MHz mikrokontroléru. [11] [43]

šifra	Velikost otisku [bit]	Velikost kódu [Byte]	Velikost na RAM [Byte]	Délka zprávy [Byte]	Počet cyklů [-]	Doba trvání [ms]
SHA256 [43]	256	1090	64	8	33600	3,35

Tabulka 4 - Výsledné hodnoty po spuštění SHA256

6.4.2 Present MAC

Present je experimentální šifra vyvinutá Orange Labs (Francie) za účelem implementace v zařízeních s malou pamětí. Pracuje se šifrou AES, používá 4 rundy a je menší než samotná AES. Výstup Present MAC má velikost 16 bajtů. Výsledky pocházejí ze studie Shangaiského týmu testovali CBC-MAC kód založený na šifře Present za účelem srovnání s algoritmem AES-MAC. Present byla spuštěna na 16 MHz mikrokontroléru. [44]

šifra	Délka klíče [bit]	Velikost kódu [bit]	Velikost na RAM [bit]	Délka zprávy [Byte]	Počet cyklů [-]	Doba trvání [ms]
CBC-MAC Present [44]	80	1926	1040	8	104160	6.51ms

Tabulka 5 - Výsledné hodnoty po spuštění CBC-MAC

6.5 Implementace vybraných šifer

K otestování byly vybrány šifry XTEA a RC6 a obě dvě byly spuštěny na našem zařízení. Účelem bylo především odzkoušení, zdali se šifry podaří na mikrokontroléru spustit, jelikož kód mohl vyžadovat větší kapacitu paměti, než máme k dispozici. Jedná se o mikrokontrolér MSP430/F2274 firmy Texas Instruments. Zdrojový kód obou šifer pochází z knihovny LibtomCrypt 1.17 a je napsán v jazyce C. Šifry byly implementovány do zařízení ve vývojovém studiu Code Composer Studio.

```
    typedef struct Plaintext1 {
        unsigned char info[32];        //deklarace struktury plaintextu a
encrypted textu
      }Plaintext1 ;
      typedef struct Enctext1 {
          unsigned char encinfo[32];
        }Enctext1 ;
    void fillPlaintext1(Plaintext1 * plaintext1)
        {
             zeromem(plaintext1->info, 32);
             unsigned char info[32] = {0,1,2,3};
             memcpy(plaintext1->info, info, sizeof(info));
        }
    int encryptPlaintext1(Plaintext1 * plaintext1, unsigned char * key,
unsigned char * iv, Enctext1 * output)
        {
          int status = my_cbc_encrypt(CIPHER, key, 16, iv, (unsigned char *)
plaintext1, 32, output->encinfo);

          return status;
        }

    void encryptiononly()
    {
      unsigned char key[16] = {0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0};
      unsigned char iv[16] = {0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0};
      Plaintext1 plaintext1;
      Enctext1 enctext1;
        fillPlaintext1(&plaintext1);
    encryptPlaintext1(&plaintext1, key, iv, &enctext1);
    }
```

Nejprve jsme vytvořili struktury *Plaintext1* a *Enctext1*, *Plaintext1* naplnili daty, zašifrovali a výstup vložili do *encinfo*. To samé poté i s druhou šifrou. Obě šifry byly v módu CBC a inicializační vektor i šifrovací klíč byly nastaveny fixně. Parametry šifer jsou patrné ze zdrojového kódu programu, který je kompletní uveden na přiloženém

DVD. Velikost šifer na flash paměti a další údaje jsou uvedeny v následující tabulce.

šifra	Velikost šifry [B]	Potřebné místo na RAM [B]	Délka klíče [B]	Velikost bloku [B]	Počet rund [-]
XTEA	3420	176	16	8	32
RC6	5636	176	16	16	20

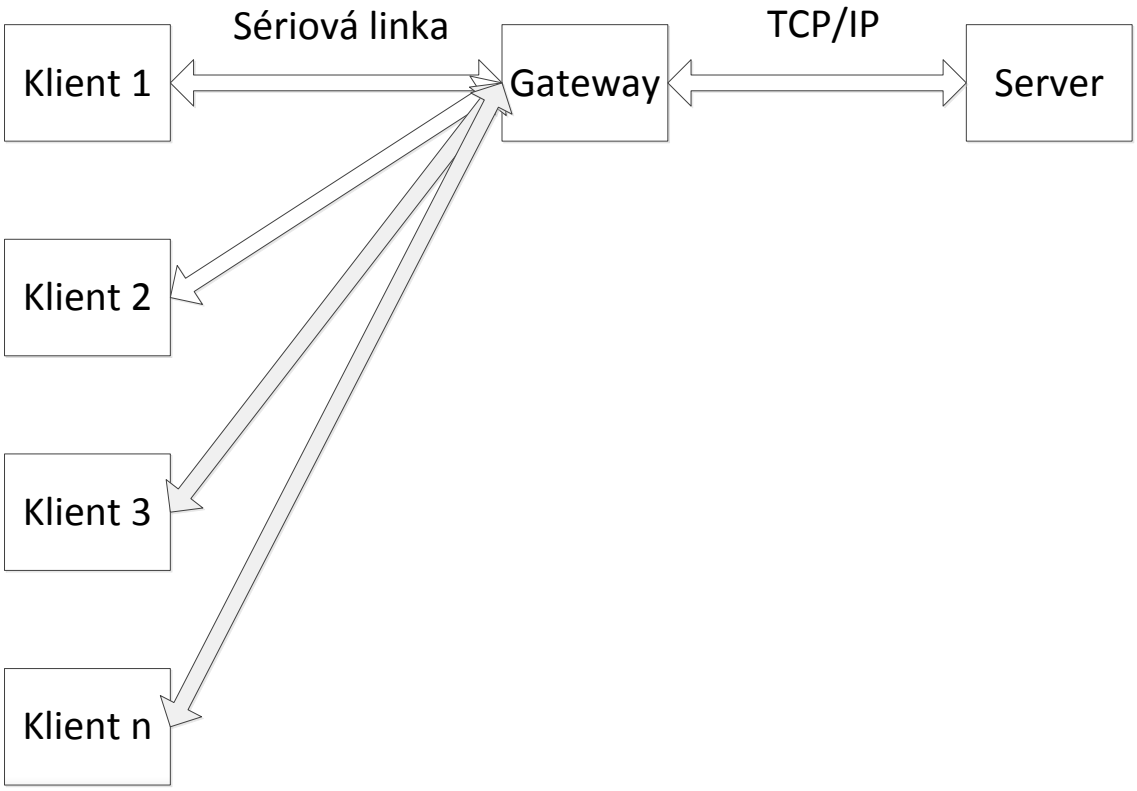
Tabulka 6 - Výsledky získané z vlastní implementace

Počet rund i délka klíče je dostatečná, aby šifra byla bezpečná a šifry se podařilo na mikrokontroléru spustit, jsou tedy vhodné k zabezpečení komunikace. Obě šifry zabíraly na RAM paměti stejnou velikost, na flash paměti se podstatně lišily. XTEA dle očekávání zabírala méně.

7 NÁVRH A IMPLEMENTACE ZABEZPEČOVACÍHO PROTOKOLU

Při návrhu našeho protokolu budeme vycházet ze struktury Modbus RTU. Ponecháme Start i End a budeme uvažovat použití na sériových linkách nicméně při návrhu a implementaci nebudeme brát v potaz fyzickou vrstvu ani případné další informace této vrstvy. Adresu i funkci potřebujeme k identifikaci klienta a posílání příkazů, takže tyto parametry jsou nezbytně nutné. Budeme je přenášet v otevřené, nezašifrované formě. Pokud bychom totiž adresu zašifrovali, útočník by sice nevěděl, se kterým klientem komunikujeme, ale zprávu bychom museli posílat jako broadcast a každý klient by adresu musel dešifrovat a pak posoudit, zdali je zpráva pro něj a má ji zpracovat anebo je pro jiného klienta a má zprávu zahodit. To by bylo zbytečné zpoždění, přičemž informace s kým server zrovna komunikuje, není pro útočníka příliš významná.

Na obrázku níže je znázorněna síť, ve které probíhá komunikace. Od serveru k bráně (Gateway) se zprávy mohou přenášet přes TCP/IP se zabezpečením SSH nebo TSL/SSL nebo s použitím IPSec. Brána zpracuje požadavek serveru, zjistí, s jakým klientem má komunikovat a jakou funkci server žádá a tyto informace zabalí do hlavičky, přiloží data poslaná serverem, zabezpečí rámec a odešle příslušnému klientu.



Obrázek 11 – schéma uvažované sítě

7.1 První verze – autentizační protokol

Tato verze řeší zajištění integrity a autentizace. Použijeme MAC kód s blokovou šifrou, který přiložíme na konec každé zprávy. MAC kód zpracuje celý rámec, tedy i pořadové číslo a datovou část. Když si klient ověří MAC kód, tak je zaručeno, že data pocházejí opravdu od serveru a zpráva nebyla znovu poslána útočníkem, jelikož klient ví, jaké pořadové číslo zprávy očekává. Také bude vědět, že zpráva nebyla pozměněna nebo se neobjevily chyby při přenosu.

Adresa	Funkce	Pořadové číslo	Data	CMAC
1B	1B	2B	12B	16B

Obrázek 12 - rámec 1. verze protokolu

V praxi pak tento návrh vypadá následovně: Vytvoříme rámec jako strukturu a naplníme jej. Adresa a Funkce mají obě po jednom bajtu, pořadové číslo má velikost 2 bajty a výstup kódu CMAC (nazývaný tag) má velikost 16 bajtů. To je dohromady 20 bajtů, takže pro Data zbývá 12 bajtů tak, aby byl rámec velký 32 bajtů. Tato velikost není nezbytná, avšak pokud by byl rámec jiné velikosti, než jsou násobky osmi, tak by při zpracování CMAC bylo nutné jednotlivé bloky vyplnit výplní, což by znamenalo větší kód. Níže následuje kus kódu zmíněného rámce, konkrétně jeho vytváření:

```
// Vytvoří strukturu rámce
typedef struct Frame {
unsigned char address;
unsigned char function;
unsigned short sn;
unsigned char data[12];
unsigned char cmac[16];
} Frame;
```

```
// Naplní paket základními daty
void fillFrame(Frame * frame)
{
  frame->address = 1;
  frame->function = 12;
  frame->sn = 0;
  zeromem(frame->data, 12);
  unsigned char data[12] = {0,1,2,3};
  memcpy(frame->data, data, sizeof(data));
}
```

Při vyplňování rámce se do proměnné *cmac* doplní nuly. Po spočítání CMAC se jeho výstup vloží do proměnné *cmac* a celý rámec se odešle. Příjemce rámec obdrží a taktéž spočítá CMAC z prvních 16 bajtů rámce, což odpovídá Adrese, Funkci, Pořadovému číslu a datové části.

```
    // Spočítá CMAC pro prvních 16 bajtů daného rámce
    int computeCmac(Frame * frame, unsigned char * key, unsigned char *output,
unsigned long * outputlength)
    {
      int status = omac(CIPHER, key, 16, (unsigned char *) frame, 16, output,
outputlength);

      return status;
    }
```

Pokud se spočítaný CMAC shoduje s druhou polovinou rámce, tedy s obdrženým kódem CMAC, je zaručena autentizace. Níže následuje popsaný kód, který nejprve vyplní vytvořený rámec, poté definuje délku výstupní proměnné pro CMAC, spočítá kód a výstup vloží do rámce. Nakonec je simulováno přijetí rámce příjemcem, který zkontroluje shodu výstupů a v případě shody se rozsvítí LED a rámec je autentizován.

```
    // Autentizace
    void cmacOnly(){
      unsigned char key[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
      // Odesilatel
      Frame frame;
      fillFrame(&frame);

      unsigned char outputCmac[16];
      unsigned long outputCmacLength = 16;
      computeCmac(&frame, key, outputCmac, &outputCmacLength);
      memcpy(frame.cmac, outputCmac, 16);
      // Příjemce
      int match = isAuthentic(&frame, key);
      if (match == 0) {
        P1OUT = 0x01;}
       else {
      }
```

Kód CMAC pracuje spolu se symetrickou šifrou. Zvolil jsem šifru AES, která se v tomto módu často používá a šifru XTEA, která je méně náročná (dle Tab. 3) a bude sloužit jako srovnání s AES. Celý kód pracuje v CBC módu a klíč byl zvolen fixně. Změřený počet cyklů je pro všechny operace, tedy od vytvoření rámce až po jeho přijetí a autentizaci. Jelikož používáme symetrické šifry a i CMAC se vytváří a ověřuje stejným způsobem, doba potřebná jen pro vytvoření rámce by byla poloviční.

šifra	Velikost bloku [Byte]	Velikost klíče [Byte]	Počet rund [-]	Velikost šifry [Byte]	Potřebné místo na RAM [Byte]	Počet cyklů [-]	Doba trvání [ms]
AES	16	16,24,32	10,12,14	11520	176	58601	3,66
XTEA	8	16	32	4054	176	35561	2,22

Tabulka 7 - Výsledné hodnoty 1.verze protokolu

7.2 Druhá verze – šifrovaný autentizační protokol

V první verzi mohl útočník odchytit rámec a snadno tak zjistit o jakou funkci se jedná, pořadové číslo a v případě komunikace ze směru klient-server i naměřená data. Druhá verze tedy přidává zabezpečení dat šifrováním symetrickou šifrou. Použijeme scénář, kdy nejprve zprávu zabezpečíme CMAC kódem a poté celý rámec zašifrujeme symetrickou šifrou. Také existuje druhá možnost – nejprve použít symetrickou šifru a poté rámec zabezpečit autentizačním kódem, ale tento způsob je méně používaný. [45] Nakonec k zašifrovanému rámci přiložíme adresu v otevřené podobě. Předpokládáme, že klientovi byl nějakým způsobem předán společný klíč, např při výrobě nebo instalaci zařízení nebo některým algoritmem pro ustanovení klíče.

Adresa 1B	Zašifrovaný rámec (Adresa, Funkce, Pořadové číslo, Data) 16B	CMAC 16B

Obrázek 13 - rámec 2. verze protokolu

Kód je stejný jako v prvním případě, jen obsahuje některé dodatečné funkce. Používáme opět šifry AES nebo XTEA se stejnými parametry jako v první verzi ale tentokrát jsou obě šifry použity nejen k výpočtu CMAC kódu ale i pro zašifrování a dešifrování celého rámce.

```
    // Zašifruje rámec
    int encryptFrame(Frame * frame, unsigned char * key, unsigned char * iv,
FrameEncrypted * output)
    {
      int status = my_cbc_encrypt(CIPHER, key, 16, iv, (unsigned char *)
frame, 32, output->encrypted);

      return status;
    }

    // Dešifruje rámec
    int decryptFrame(FrameEncrypted * encryptedFrame, unsigned char * key,
Frame * decryptedFrame)
    {
      int status = my_cbc_decrypt(CIPHER, key, 16, encryptedFrame->iv,
encryptedFrame->encrypted, 32, (unsigned char *) decryptedFrame);

      return status;
    }
```

Příjemce rámec dešifruje a postupuje stejně jako v první verzi. Z prvních 16 bajtů spočítá CMAC a pokud je shodný s přijatým CMAC kódem, který se nalézá v druhé polovině rámce, tak je ověřena autentičnost a rozsvítí se LED dioda. Inicializační vektor i šifrovací klíč byly nastaveny fixně.

```
// Autentizace spolu se šifrováním
void cmacWithEncryption()
{
  unsigned char key[16] = {0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0};
  unsigned char iv[16] = {0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0};

  // Odesílatel
  Frame frame;
  fillFrame(&frame);

  unsigned char outputCmac[16];
  unsigned long outputCmacLength = 16;
  computeCmac(&frame, key, outputCmac, &outputCmacLength);
  memcpy(frame.cmac, outputCmac, 16);

  FrameEncrypted frameEncrypted;
  frameEncrypted.address = frame.address;
  encryptFrame(&frame, key, iv, &frameEncrypted);
  memcpy(frameEncrypted.iv, iv, sizeof(iv));

  // Příjemce
  Frame decryptedFrame;
  decryptFrame(&frameEncrypted, key, &decryptedFrame);

  int match = isAuthentic(&decryptedFrame, key);
  if (match == 0) {
    P1OUT = 0x01;
  } else {
        }
}
```

Kód zabírá větší množství paměti a také je potřeba více cyklů procesoru k jeho provedení právě díky přidanému šifrování.

šifra	Velikost bloku [Byte]	Velikost klíče [Byte]	Počet rund [-]	Velikost šifry [Byte]	Potřebné místo na RAM [Byte]	Počet cyklů [-]	Doba trvání [ms]
AES	16	16,24,32	10,12,14	13130	176	111830	6,98
XTEA	8	16	32	5664	176	75196	4,69

Tabulka 8 - Výsledné hodnoty 2.verze protokolu

8 ANALÝZA VÝSLEDNÉHO ŘEŠENÍ

Navržené a implementované verze zabezpečovacího protokolu se podařilo spustit a to se šiframi AES i XTEA. AES je samozřejmě náročnější, jelikož využívá s-boxy a XTEA pouze modulární operace. Velikost kódu s AES šifrou byla 11520 bajtů respektive 13130 bajtů v druhé verzi, zatímco velikost kódu se šifrou XTEA byla méně než poloviční – 4054 bajtů a v druhé verzi 5664 bajtů. Všechny kódy zabíraly na RAM paměti stejnou velikost 176 bajtů. Nejnáročnější řešení je druhá verze protokolu s AES šifrou a tento kód zabíral na flash paměti 40%. Zbývající prostor je však dostatečně velký i pro dodatečné informace protokolů nižších vrstev a případné další úpravy kódu. Vlivem dodatečného zašifrování se doba potřebná k odeslání a autentizaci zvýšila takřka dvojnásobně.

Obě varianty protokolu jsou odolné vůči Útoku hrubou silou, jelikož používáme dostatečnou délku klíče (128 bitů) a testování všech možných kombinací, kterých jednotlivé bajty mohou nabývat (2^{128}) je mimo dnešní výpočetní možnosti.

Také slovníkový útok by neobstál, jelikož ten spočívá v systematickém testování předem připravených hesel, což jsou slova často se vyskytující v daném jazyce. My ovšem nemáme šifrovací ani autentizační klíč v podobě slova ale je to náhodně generovaná posloupnost bitů, tudíž jsou obě varianty protokolu odolné vůči slovníkovému útoku.

Použití útoku Replay attack je taktéž znemožněno. Pokud by útočník odchytil rámec a chtěl jej později znovu poslat, rámec by měl zastaralé pořadové číslo a příjemce by věděl, že se jedná o starý rámec a zahodil by jej. Zabezpečení by šlo dále zvýšit použitím časového razítka.

Obě verze jsou taktéž odolné vůči lineární a diferenciální kryptoanalýze, jelikož zatím nebyl proveden úspěšný útok při použití standardního počtu rund a délky klíče.

Při útoku Man in the middle by útočník musel znát autentizační klíč, pokud by útočil na první verzi protokolu, jelikož příjemce by ověřením MAC kódu zjistil, že rámec byl pozměněn a zahodil by jej. U druhé verze protokolu by útočník musel nejprve rámec dešifrovat, poté jej pozměnit, přiložit MAC spočítaný pravým autentizačním klíčem a rámec znovu zašifrovat. Toto by vyžadovalo znalost obou klíčů, které útočník nemá jak zjistit. Taktéž pokud by útočník chtěl poslat svůj vlastní vytvořený rámec, tak by musel u první verze znát autentizační klíč a u druhé verze protokolu navíc i šifrovací klíč. Tento druh útoku by taktéž neobstál proti první či druhé verzi.

Největší slabinou implementovaného kódu je fixně nastavený šifrovací klíč pro šifry, autentizační klíč pro MAC a inicializační vektor. Rozdíl mezi šifrovacím a autentizačním klíčem je v použití. Šifrovací klíč používají šifry AES a XTEA k zašifrování celého rámce, zatímco autentizační klíč se používá k vypočtení MAC kódu, k čemuž se používá i inicializační vektor. V reálném nasazení by tato slabina byla ale samozřejmě odstraněna. Nám šlo o otestování programu, a proto nebylo důležité, zdali se klíče a vektor generovaly náhodně nebo ne. Při skutečném provozu by se klíče i vektor tedy generovaly náhodně nebo vložily přímo do zařízení při výrobě.

ZÁVĚR

Cílem této práce bylo seznámit čtenáře s možnostmi zabezpečení komunikace prostřednictvím různých kryptografických metod a následný rozbor analyzovaných šifer pro možnost použití na zařízeních s nízkým výpočetním výkonem a malou paměťovou kapacitou. V první části práce se čtenář seznámil se základy kryptografie, s pojmy a rozdělením kryptografických metod. Čtenář získal poznatky o bezpečnostních službách a o tom jak fungují, o symetrické kryptografii a jejich formách a o asymetrické kryptografii a využitelnosti hash funkcí při autentizaci. Dále byly popsány nejznámější asymetrické šifry s uvedením jejich dnešního použití a čtenář se dověděl, co to je funkce hash, jak pracuje a k čemu slouží. Následně autor popsal princip autentizačního kódu a jeho použití a jeho verze s blokovou šifrou či hash funkcí.

V další kapitole se rozebíraly a popisovaly bezpečnostní mechanismy často používaných protokolů a následně byly vybrány nejznámější kryptografické útoky, které se v současnosti objevují. Na tuto kapitolu navazuje popis dnešních zabezpečovacích protokolů, které pracují na síťové či aplikační vrstvě. Následují řídící protokoly, které se používají v průmyslové výrobě či jiných rozsáhlých systémech. Byl popsán systém Scada a bylo znázorněno schéma sítě a popsány jednotlivé hardwarové součásti systému. Na tuto problematiku navazuje protokol Modbus, který byl rozebrán a popsán, včetně nákresu tří verzí rámců a vysvětlení jednotlivých částí rámce.

V následující části práce se čtenář mohl dovědět informace o omezených zařízeních, konkrétních symetrických šifrách stejně jako o autentizačních kódech. Tato kapitola čtenáři přiblížila specifické omezení mikrokontrolerů a popsala symetrické šifry, které se hodí k testování na tomto zařízení. Také byly vypsány parametry těchto šifer a algoritmů a dva vybrané algoritmy byly otestovány na našem zařízení. Nakonec byl navržen a vytvořen kryptografický zabezpečovací protokol ve dvou verzích. Na mikrokontroléru se jej podařilo spustit a změřit jeho parametry. Největší kód zabíral na flash paměti 13130 bajtů, na RAM 176 bajtů a bylo třeba 111830 cyklů na jeho provedení což odpovídalo době 6,98ms. Na závěr práce byla provedena analýza obou verzí protokolu s ohledem na útoky popsané ve třetí kapitole.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

LITERATURA

[1] Menezes, Alfred J. Handbook of applied cryptography. Vyd. 1. Boca Raton: CRC Press, 1997, 780 s. ISBN 08-493-8523-7.

[2] A Brief History of Cryptography. Cypher Research Laboratories Pty. Ltd. [online]. 2006, 17. 4. 2013 [cit. 2014-06-02]. Dostupné z: http://www.cypher.com.au/crypto_history.htm

[3] Sale, Tony. The Colossus computer, 1943-1996: and how it helped to break the German Lorenz cipher in WWII. Cleobury Mortimer, Shropshire: M, 1998, 17 p. ISBN 09-477-1236-4.

[4] Frequently Asked Questions (FAQ) About the Electronic Frontier Foundation's "DES Cracker" Machine. [online]. [cit. 2014-06-01]. Dostupné z: https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

[5] Stallings, William. Cryptography and Network Security. 4th edition. [s.l.] : [s.n.], 2006. 592 s. ISBN 0131873164.

[6] Till, Michal. Pro pamětníky : Enigma. Krypta.cz: Magazín o informační bezpečnosti.[online].3.11.2001[cit.2014-06-01].Dostupné z: http://www.krypta.cz/articles.php?ID=56

[7] Chaum, Edited by G.R. Blakley and David. Advances in cryptology: proceedings of CRYPTO 84. Berlin: Springer-Verlag, 1985. ISBN 03-871-5658-5.

[8] Voců, Michal. Šifrování a šifrovací systémy. Ikaros [online]. 1997, roč. 1, č. 6 [online]. [cit. 2014-05-24]. Dostupné z: http://ikaros.cz/node/84

[9] RFC 2792. DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System. Pennsylvania: Network Working Group, 2000. Dostupné z: http://tools.ietf.org/html/rfc2792

[10] Diffie, Whitfield; Hellman, Martin. New Directions in Cryptography. IEEE Transactions on Information Theory. listopad 1976, roč. 22, čís. 6, s. 644 - 654. [online]. [cit. 2014-05-24]. Dostupné z: http://www-ee.stanford.edu/~hellman/publications/24.pdf

[11] Hernady, Robert. Zavedení hash algoritmů SHA-2 v prostředí OS Microsoft Windows. In: Lupa.cz: Server o českém internetu [online]. 6.11.2009. 2009 [cit. 2013-12-30]. Dostupné z: http://www.lupa.cz/clanky/zavedeni-hash-algoritmu-sha-2-v-prostredi-ms-win/

[12] Doležal, Dušan. Jak funguje digitální podpis. Interval.cz [online]. 23. 10. 2002 [cit. 2013-12-30]. Dostupné z: http://interval.cz/clanky/jak-funguje-digitalni-podpis/

[13] Dworkin, Morris. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication [online]. Gaithersburg, 2005 [cit. 2014-06-01].

Dostupné z: NIST Special Publication 800-38B. Computer Security Division Information Technology Laboratory National Institute of Standards and Technology

[14] Denis, Tom St. LibTomCrypt: Developer Manual. Ottawa, Ontario Canada. Dostupné z: http://libtom.org/?page=features

[15] Replay Attacks. Microsoft. MSDN Library [online]. [cit. 2014-06-02]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa738652%28v=vs.110%29.aspx

[16] RFC 4418. UMAC: Message Authentication Code using Universal Hashing. CSU Sacramento: Network Working Group, 2006. Dostupné z: https://tools.ietf.org/html/rfc4418

[17] RFC 6071. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. NIST: Internet Engineering Task Force (IETF), 2011. Dostupné z: http://tools.ietf.org/html/rfc6071

[18] Bellare, Mihir a Tadayoshi KOHNO. Hash Function Balance and Its Impact on Birthday Attacks. Lecture Notes in Computer Science [online]. s. 401 [cit. 2014-06-02]. DOI: 10.1007/978-3-540-24676-3_24. Dostupné z: http://link.springer.com/10.1007/978-3-540-24676-3_24

[19] Hellman, M. A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory [online]. 1980, vol. 26, issue 4, s. 401-406 [cit. 2014-06-01]. DOI: 10.1109/TIT.1980.1056220. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1056220

[20] Lórencz, R. Pokročilá kryptografie, přednáška Lineární a diferenciální kryptoanalýza, ČVUT v Praze, Praha. [online]. [cit. 2014-06-01]. Dostupné z: https://edux.fit.cvut.cz/oppa/MI-KRY/prednasky/prednaska7_10.pdf

[21] Quisquater, Jean-Jacques. Side channel attacks. State-of-the-art [online]. 2002 [cit. 2014-06-01]. Dostupné z:http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf

[22] Till, Michal. Man in the middle attack. Krypta.cz: Magazín o informační bezpečnosti. [online]. 16.12.2001 [cit. 2014-06-01]. Dostupné z: http://www.krypta.cz/articles.php?ID=94

[23] RFC 6101. The Secure Sockets Layer (SSL) Protocol Version 3.0. Netscape Communications: Internet Engineering Task Force (IETF), 2011. Dostupné z: http://tools.ietf.org/html/rfc6101

[24] RFC 4346. The Transport Layer Security (TLS) Protocol: Version 1.1. RTFM, Inc.: Network Working Group, 2006. Dostupné z: http://www.ietf.org/rfc/rfc4346.txt

[25] RFC5246. The Transport Layer Security (TLS) Protocol: Version 1.2. Network Working Group, 2008. Dostupné z: http://tools.ietf.org/html/rfc5246

[26] RFC 4253. The Secure Shell (SSH) Transport Layer Protocol. Network Working Group, 2006. Dostupné z: http://tools.ietf.org/html/rfc4253

[27] RFC 3961. Encryption and Checksum Specifications for Kerberos 5. MIT:

Network Working Group, 2005. Dostupné z: https://tools.ietf.org/html/rfc3961

[28] Pelalez, Manuel. SCADA: A big challenge for information security professionals [online]. 2010, 23.8.2010 [cit. 2014-06-02]. Dostupné z: https://isc.sans.edu//diary/SCADA:+A+big+challenge+for+information+security+professionals/9436

[29] Control System Authentication: Secure SCADA Communications Protocol. In: HARDLEY, Mark. Pacific Northwest National Laboratory [online]. 2010 [cit. 2014-06-02].

[30] Wright, Andrew. SOURCEFORGE. ScadaSafe [online]. [cit. 2014-06-02]. Dostupné z: http://scadasafe.sourceforge.net/

[31] Modbus Organization, Inc. Modbus Application Protocol Specification: V1.1b 3 [online]. 2012 [cit. 2014-06-02]. Dostupné z: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[32] Atmel: ATtiny25/ATtiny45/ATtiny85 datasheet summary. [online]. [cit. 2014-05-24]. Dostupné z: http://www.atmel.com/Images/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet-Summary.pdf

[33] Atmel: AVR ATmega128 datasheet. [online]. [cit. 2014-05-24]. Dostupné z: http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[34] Freescale semiconductor: MC9s08ac128 datasheet. [online]. [cit. 2014-05-24]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/fact_sheet/9S08AC128FS.pdf

[35] TI MSP430F22x2, MSP430F22x4 datasheet. [online]. [cit. 2014-05-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/msp430f2274.pdf

[36] TI MSP430F15x, MSP430F16x, MSP430F161x, Mixed Signal Microcontroller datasheet. [online]. [cit. 2014-05-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/msp430f1610.pdf

[37] Cazorla, Mickaël, Kevin Marquet, and Marine Minier. "Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks⋆." IDEA 64.128: 34

[38] Law, Yee Wei, Jeroen Doumen, and Pieter Hartel. "Benchmarking block ciphers for wireless sensor networks." Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on. IEEE, 2004. Dostupné z: http://saluc.engr.uconn.edu/refs/secsensor/law_benchmarkingBlockCiphers.pdf

[39] Rinne, Sören, Thomas Eisenbarth a Christof Paar. Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers. SPEED: Software Performance Enhancement for Encryption and Decryption [online]. 2007, s. 33-43 [cit. 2014-05-24]. Dostupné z: http://www.ei.ruhr-uni-bochum.de/media/crypto/veroeffentlichungen/2011/01/29/lw_speed2007.pdf

[40] Khovratovich, Dmitry, Gaetan Leurent a Christian Rechberger. Cryptoanalysis of Full IDEA. [online]. [cit. 2013-12-30]. Dostupné z: http://www.cs.bris.ac.uk/eurocrypt2012/Program/Tues/Rechberger.pdf

[41] Implementace šifry XTEA. [online]. 10.1.2009. 2009 [cit. 2013-12-30]. Dostupné z: http://pandatron.cz/?1083&implementace_sifry_xtea

[42] Liu, Wei, Rong Luo, and Huazhong Yang. "Cryptography overhead evaluation and analysis for wireless sensor networks." Communications and Mobile Computing, 2009. [online] Dostupné z:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4797303

[43] Implementations of hash functions in Atmel AVR devices. [online]. [cit. 2013-12-30]. Dostupné z: http://perso.uclouvain.be/fstandae/source_codes/hash_atmel/

[44] Gong, Zheng, et al. "Towards secure and practical MACs for body sensor networks." Progress in Cryptology-INDOCRYPT 2009. Springer Berlin Heidelberg, 2009. [online] 182-198. Dostupné z: http://dl.acm.org/citation.cfm?id=1697047.

[45] Burda, K.: Bezpečnost informačních systémů. Elektronická skripta VUT v Brně, Brno 2013.

ABECEDNÍ PŘEHLED POUŽITÝCH ZKRATEK, VELIČIN A SYMBOLŮ

ADU	Application Data Unit
AES	Advanced Encryption Standard
AES/CTR	AES v módu Counter
ASCII	American Standard Code for Information Interchange
CBC	Cipher-block chaining
CBCMAC	cipher block chaining message authentication code
CFB	Cipher feedback
CMAC	Cipher-based MAC
DAS	Data Acquisition System
DES	Data Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECB	Electronic codebook
FISH	Fibonacci Shrinking
GNUPG	GNU Privacy Guard
HMAC	Keyed-hash Message Authentication Code
HMI	Human Machine Interface
HTTPS	Hypertext Transfer Protocol Secure
IBM	International Business Machines Corporation
IDEA	International Data Encryption Algorithm
IPSEC	IP security
ISO/OSI	International Standards Organization / Open Systen Interconnection
IV	Initialization vector
LED	Light-Emitting Diode
MAC	Message Authentication Code
MD5	Message-digest algorithm
MIT	Massachusetts Institute of Technology
MTU	Master Terminal Unit
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OFB	Output feedback

OMAC	One-key message authentication code
OPENSSL	-
PDU	Protocol data unit
PGP	Pretty Good Privacy
PKCS#7	Public Key Cryptographic Standards
RAM	Random-access memory
RC4	Ronald’s Cipher 4
RC5	Ronald’s Cipher 5
RC6	Ronald’s Cipher 6
RIJNDAEL	Rijmen+ Daemen
RSA	Rivest+Shamir+Adleman
RTU	Remote Terminal Unit
SCADA	Control and data acquisition
SHA	Secure Hash Algorithm
SSCP	Secure SCADA Communications Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
VPN	Virtual private network
XTEA	Extended Tiny Encryption Algorithm

Baud	modulační rychlost
Bit	jednotka informace
Byte	jednotka množství dat
F [Hz]	frekvence (jednotka Hertz)
T [s]	časová perioda (jednotka sekunda)

SEZNAM PŘÍLOH

Zdrojové kódy programů jako projekty CCS studia na přiloženém nosiči:

project_rc6_xtea_EncOnly

project_cmac_final