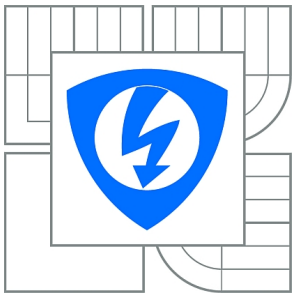
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ZABEZPEČENÝ PŘÍSTUP K INFORMAČNÍMU PANELU

SECURE ACCESS TO INFORMATION PANEL

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

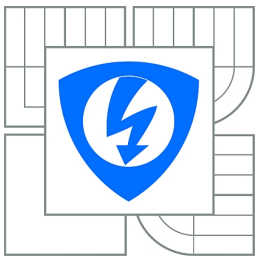
AUTOR PRÁCE — JAROSLAV PROCHÁZKA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SOBOTKA

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Jaroslav Procházka	***ID:***	106740
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Zabezpečený přístup k informačnímu panelu

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte metody komunikace informačních panelů s řídící jednotkou. Zaměřte se především na komunikační protokoly a metody zabezpečení přenosu zobrazovaných informací. Navrhněte koncepci modulu pro zabezpečenou komunikaci mezi zobrazovacím panelem a terminálem.

DOPORUČENÁ LITERATURA:

[1] SCHNEIER, Bruce. Applied cryptography. 2nd edition. [s.l.] : John Wiley & Sons, 1996. 784 s. ISBN 0-471-11709-9 .
[2] Stallings, W.: Cryptography and Network Security: Principles and Practice. Prentice Hall, Englewood Cliffs 2003.

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: Ing. Jiří Sobotka
Konzultanti bakalářské práce:

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

UPOZORNĚNÍ:

ANOTACE

Bakalářská práce se zabývá zabezpečeným přístupem k informačnímu panelu, která řeší metody komunikace informačních panelů s řídící jednotkou. Jsou zde popsána komunikační rozhraní. Práce je zaměřena především na komunikační protokoly a metody zabezpečení přenosu zobrazovaných informací. Práce se zabývá protokolem Diffie-Hellman, který je podrobně rozebrán. Je zde vysvětlena podstata útoku „Man in the middle“ na protokol Diffie-Hellman. Způsob obrany proti útoku „Man in the middle“ použitím digitálního podpisu v protokolu Diffie-Hellman po nezabezpečeném komunikačním kanálu. V závěru je navrhnuta koncepce modulu pro zabezpečenou komunikaci mezi zobrazovacím panelem a řídícím terminálem. Podrobně jsou popsány tři navržené simulace v programu.

Klíčová slova: komunikační protokol, kryptografie, protokol Diffie-Hellman, digitální podpis, útok „Man in the middle“.

ABSTRACT

My Thesis deals with secure access to the information panel, which deals with methods of communication of information panels with control unit. There are described the communication interface. The Thesis is focused mainly on communication protocols and security methods of transmission of information. The Thesis deals with the Diffie-Hellman Protocol, which is detailed discussed. There is explained the nature of the attack, the „Man in the middle“ on the Diffie-Hellman Protocol. A method of defense against attack by the „Man in the middle“ by using a digital signature in the Diffie-Hellman Protocol over an unsecured communication channel. In conclusion, the concept is designed for secure communications between the display panel and the control Terminal. There are described in detail the three proposed simulation in the programme.

Keywords: communication protocol, cryptography, protocol Diffie-Hellman, digital signature, the attack „Man in the middle“.

PROCHÁZKA, J. *Zabezpečený přístup k informačnímu panelu.* Brno: FEKT VUT v Brně, 2012. 46 s. Vedoucí práce Ing. Jiří Sobotka.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zabezpečený přístup k informačnímu panelu“ 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 ……. .....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Jiřímu Sobotkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne …….. .....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Existují projekty, ve kterých je propojitelnost a vzájemná komunikace více zařízení mezi sebou klíčová. Jedná-li se o různá zařízení stejného výrobce, obvykle je nabízen i nástroj pro jejich centrální správu a ovládaní. Ne vždy však produktové portfolio jednoho výrobce pokrývá všechny potřeby zákazníka, který je pak nucen koupit produkty od různých výrobců. Oproti očekávání pak může být výsledné řešení obtížně spárovatelné, protože každý výrobce dodává vlastní ovládací systém (program), ze kterých nelze vytvořit jednotný a spolupracující celek.

Tento problém řeší řídící jednotky - zařízení, které je možné naprogramovat k ovládání okolních zařízení. Ačkoliv se může zdát, že problém z předchozího odstavce je vyřešen, můžeme narazit na omezení přímo na straně řídícího zařízení. Moderní řídící jednotky zajišťují řídící, komunikační a informační funkce. Od uživatelů přijímají povely prostřednictvím dotykového panelu nebo počítačové aplikace. Protože od dotykového panelu lze očekávat jednoduchou logiku a počítačové aplikace jsou jednoduché a jednoúčelové, dají se využívat v mnoha zařízení.

Protože řídící jednotka plní svůj primární úkol - řídí připojená zařízení na základě uživatelských povelů - není možné se bez ní jednoduše obejít. Proto je potřeba přijít s takovým řešením, které ji bude využívat a stávající ovládací prostředky vhodně doplňovat. Toto řešení spočívá v analýze komunikačního protokolu, který používá dodávané aplikace na počítači při komunikaci s řídící jednotkou.

Rozhraní slouží k připojení specifických zařízení do systému, k obsluze zařízení a k zavedení potřebných signálů. Sběrnice slouží k datové komunikaci periferních zařízení s řídící jednotkou.

Práce obsahuje některé anglické výrazy pojmů všeobecně známých v terminologii komunikačních technologií. U těchto výrazů se český překlad běžně nepoužívá a pro daný pojem je ustálen již zmíněný anglický výraz.

V první kapitole jsou popsány teoretické poznatky ke komunikačnímu rozhraní.

Ve druhé kapitole jsou rozepsány poznatky o komunikačních protokolech, které souvisí se specifickým rozhraním.

Ve třetí kapitole jsou shrnuty metody zabezpečení přenosu zobrazovaných informací. Je zde podrobně rozebrán protokol Diffie-Hellman i jeho slabina, útok metodou „Man in the middle“. Princip sestavení klíče a postup sestavení digitálního podpisu.

V poslední kapitole jsou uvedeny postupy simulace v programu CrypTool. Jsou zde navrženy simulace protokolu Diffie-Hellman mezi řídícím modulem a dvěma zobrazovacími panely. Navržená simulace útoku „Man in the middle“ v protokolu Diffie-Hellman mezi řídícím modulem a zobrazovacím panelem. Poslední simulace je zaměřena na princip zabezpečení digitálních dat digitálním podpisem s kombinací asymetrického šifrování RSA.

1 KOMUNIKAČNÍ ROZHRANÍ

Komunikačním rozhraním se míní nástroj pro jakékoli elektronické zařízení, tak aby mohlo zařízení komunikovat s dalším zařízením. Stále více se využívají rozhraní, u kterých není potřeba mechanického propojení. Takovým komunikačním rozhraním je například WiFi.

1.1 Komunikační rozhraní CAN

Standardní rozhraní využívaná spolu s mikrokontroléry, pro vnitřní komunikační síť, senzory a funkční jednotky. CAN je navržený speciálně pro automobilové aplikace, ale nyní se používá také i v jiných oblastech, jako jsou průmyslové automatizace, nebo v zdravotnickém zařízení.

Vývoj rozhraní CAN začal původně v roce 1983 firmou Robert Bosch GmbH. Rozhraní bylo oficiálně vydané v roce 1986 na kongresu Society of Automotive Engineers v Detroitu. První rozhraní CAN vyrobil Intel a Philips a v roce 1987 byl uveden na trh. Uváděná maximální teoretická rychlost přenosu na rozhraní je 1 Mb/s.

1.2 Komunikační rozhraní IBIS

Sběrnice slouží k datové komunikaci periferních zařízení s dalším zařízením. Sběrnice IBIS vznikla zhruba před 15 lety v Německu. Stala se přirozeným standardem pro evropskou veřejnou dopravu, včetně naší. Česká verze protokolu nese označení IPIS. Sběrnice je sice relativně pomalá, ale umožňuje připojení naprosté většiny periferií standardních zařízení.

IBIS se dnes používá pro spojení počítače s vizuálními tably, s pokladnou, s označovači jízdenek, s ukazateli času a pásma, s digitálním hlásičem, s jednoduchou radiostanicí aj.

1.3 Komunikační rozhraní RS485

RS485 je plnohodnotné průmyslové rozhraní. Jedná se o sériovou datovou komunikaci po dvouvodičovém stíněném kabelu typu twisted pair (zkroucený pár), kde jednotlivé logické stavy jsou reprezentovány rozdílovým napětím mezi oběma vodiči. Jedním párem vodičů lze propojit mezi sebou více zařízení (až 32), která si mají mezi sebou přenášet data. Aby to však bylo možné, je nutné zajistit řízení přístupu na rozhraní. Jde tedy o obousměrný přenos dat, ale protože nelze po jednom páru najednou přenášet data oběma směry, je zde vyžadováno řízení směru přenosu. Vzdálenost "krajních" jednotek je až 500 m a přenosová rychlost až 200 kb/s.

2 KOMUNIKAČNÍ PROTOKOLY

Komunikační protokol je systém formátů digitálních zpráv a pravidel pro výměnu těchto zpráv, podle kterých probíhá komunikace mezi výpočetními systémy a v oblasti telekomunikací. Protokol může mít standardní formát.

Protokoly mohou obsahovat signalizaci, ověřování, rozpoznání chyb nebo schopnost opravit vzniklé chyby.

Komunikační protokol definuje syntaxi, sémantiku a synchronizaci komunikace. Případné chování je obvykle nezávislé na vzájemné komunikaci. Protokol lze tedy realizovat hardwarově, softwarově nebo kombinací obou.

Doposud v informatice neexistuje žádný obecně uznávaný standardní zápis definice protokolu

Komunikující systémy používají dobře definované formáty pro výměnu zpráv. Každá zpráva má přesný význam a má v úmyslu vyvolat reakci definovanou v přijímači. Protokol proto popisuje syntaxi, sémantiku a synchronizaci komunikace.

Přesný popis komunikačního protokolu usnadňuje interoperabilitu různých implementací programů, které se podílejí na vzájemné komunikaci. Otevřený přístup ke specifikaci protokolu urychluje rozvoj a rozšiřování technologií do nejrůznějších oborů lidské činnosti.

2.1 Protokol CAN

Protokol CAN je mezinárodní standard definovaný v ISO 11898. Tento CAN protokol je definován i v ISO 16845, který zaručuje zaměnitelnost mezi čipy. [1]

2.1.1 Principy výměny dat

Topologie CAN je založena na vysílacím mechanismu broadcast. Broadcast je vysílání zpráv, které jsou přijaty v síti všemi připojenými síťovými rozhraní. CAN je založen na protokolu přenosu orientovaných zpráv. Definuje více obsah zprávy, než uzlu nebo adresy uzlu. Každá zpráva má identifikátor zprávy, který je unikátní v rámci celé sítě, protože definuje obsah a priority zprávy. Výše zmíněný identifikátor je důležitý, pokud několik uzlů soutěží o přístup na sběrnici.

V důsledku zaměření na řešení systému je dosaženo vysoké míry systémové a konfigurační flexibility. Přidat uzel do existující sítě CAN je snadné, bez hardwarové nebo softwarové úpravy stávajícího uzlu, dokud nový uzel je čistě přijímač. To umožňuje modulární koncept, a také umožňuje příjem více údajů a synchronizaci distribuovaných procesů. Přenos dat je ovlivněn dostupností určitých typů uzlů, které umožňují jednoduchý servis a modernizaci sítě.

2.1.2 Přenos dat v reálném čase

V reálném čase se zpracování naléhavosti zpráv, které se mají vyměňovat v síti, může velmi lišit. Například při zatížení motoru, musí být zprávy zasílány častěji a s menším

zpožděním, než za jiných okolností, například teplota motoru.

Priority, které se přenáší ve zprávě, se srovnávají s dalšími méně naléhavými zprávami, které jsou určeny pomocí identifikátoru každé zprávy. Priority jsou stanoveny během návrhu systému v podobě odpovídající binární hodnoty a nelze je dynamicky změnit. Identifikátor s nejnižším binárním číslem má nejvyšší prioritu.

Konflikty přístupu na sběrnici jsou řešeny bitovým rozhodčím identifikátoru, zapojeným v každém uzlu, pozorující každý bit sběrnice. Všechny uzly s recesivním přenosem a dominantním pozorováním ztrácí přístup na sběrnici. Všechny tyto „zahozené“ uzly se automaticky stávají příjemci zpráv s vyšší prioritou a nemohou se pokusit o přenos, dokud sběrnice není opět k dispozici.

Předávání žádostí je zpracováváno v pořadí jejich důležitosti pro systém jako celek. To je výhoda zejména v situacích přetížení. Vzhledem k přístupu na sběrnici je priorita přiřazena na základě zpráv. Je možné zajistit nízkou latenci jednotlivých časů v systému. [1]

2.1.3 Formáty zpráv

CAN protokol podporuje dva formáty zprávy. Jediný podstatný rozdíl je v délce identifikátoru. Základní rámec zprávy podporuje identifikátor o délce 11 bitů. Tento formát je znám jako CAN 2.0 A. Rozšířený rámec zprávy CAN podporuje identifikátor o délce 29 bitů. Tento rozšířený rámec je znám jako CAN 2.0 B. [2]

Obrázek 1: Formát rámce CAN.

2.1.4 Základní formát zprávy CAN

Formát rámce CAN je zobrazen v Obrázku 1. CAN zpráva začíná start bitem, který se nazývá „Start Of Frame“, dále jen SOF. Poté následuje „Arbitration field“, který se skládá z bitu, který se nazývá „Remote Transmission Request“ dále jen RTR. Tento bit slouží k rozlišení mezi datovým rámcem a všesměrovým rámcem v žádosti o údaje, tzv. „remote frame“. Následující „Control field“ obsahuje „IDentifier Extension“ zkráceně IDE, který rozlišuje mezi základním rámcem a rozšířeným rámcem CAN, stejně jako „Data Length Code“, zkráceně DLC, sloužící k označení počtu následujících datových bajtů v datovém poli. Pokud zpráva obsahuje „remote frame“, DLC obsahuje počet požadovaných datových bajtů. Následující datové pole je schopno uchovat až 8 datových bajtů. Integrita rámce je zaručena následujícím kontrolním součtem „Cyclic Redundant Check“, dále jen CRC. Potvrzení je prováděno polem „ACKnowledge“, zkráceně ACK. Bit ACK je odeslán jako recesivní a přepsán v přijímači na dominantní bit, který obdržel data správně. Správné zprávy jsou uznány přijímačem bez ohledu na výsledek přejímací zkoušky. Konec zprávy je označen „End Of Frame“, zkráceně EOF.

Pole „Intermission Frame Space“, dále jen IFS, je minimální počet bitů oddělující po sobě navazující zprávy. Pokud jiné stanice spustí přenos, sběrnice zůstává po tomto nečinná. [1]

2.1.5 Rozšířený formát zprávy CAN

Rozdíl mezi zprávou rozšířeného formátu rámce a formátu rámce základní zprávy je použitá délka identifikátoru. Identifikátor tvořený z 29 bitů se skládá z 11 bitů identifikátoru „základní identifikátor“ a 18 bitů rozšíření, dále jen „identifikátor rozšíření“. Rozdíl mezi CAN základním formátem a rozšířeným formátem rámce se provádí pomocí bitu IDE, který je přenášen jako dominantní v případě 11-ti bitového rámce a v případě 29-ti bitového rámce se přenáší jako recesivní. Pokud se tyto dva formáty vyskytnou společně na jedné sběrnici, je stanovena zpráva s vyšší prioritou pro přístup na sběrnici. V případě kolize s různými formáty a stejnými identifikátory má zpráva složená z 11 bitů vždy přednost před 29 bitovou zprávou.

Rozšířený formát má své výhody a nevýhody. Zpoždění sběrnice je delší, minimálně 20 bitů. Zprávy v rozšířeném formátu vyžadují větší šířku pásma, asi o 20 %. Výkon detekce chyb je nižší, protože polynom pro 15-ti bitové CRC je optimalizováno pro rámce délky do 112 bitů.

CAN kontroléry, které podporují formáty rozšířených rámců zpráv, jsou také schopny odesílat a přijímat zprávy v základním formátu rámce CAN. CAN kontroléry, které umí pracovat pouze se základním formátem rámce, nemusí interpretovat správně rozšířené rámce. Existují však CAN kontroléry, které podporují pouze formáty základního rámce a pokud rozpozná rozšířený formát rámce zprávy, bude ho ignorovat.

2.1.6 Odhalení a signalizace chyb

Na rozdíl od ostatních sběrnicových systémů, CAN protokol nepoužívá potvrzovací zprávy, ale místo toho signalizuje chyby ihned, jakmile se vyskytnou. Pro detekci chyb protokol CAN implementuje tři mechanismy na úrovni zpráv.

První mechanismus je cyklický redundantní součet CRC. Zabezpečuje informace v rámci přidání kontrolní sekvence FCS na konci přenosu. Přijatá sekvence FCS je přepočítána a je znova testováno přijaté FCS. Pokud se neshodují, došlo k chybě CRC.

Druhý mechanismus se nazývá kontrola rámce. Tento mechanismus ověřuje strukturu přenášeného rámce a kontroluje bitové pole z pevně daného formátu a velikosti rámce. Chyby zjištěné kontrolou rámce jsou označeny za chybu formátu.

Poslední mechanismus je potvrzení chyby. Přijímač zprávy odesílá potvrzení přijatého rámce. Pokud vysílač neobdrží potvrzení, je indikována chyba potvrzení.

CAN protokol implementuje také dva mechanismy pro detekci chyb na úrovni bitů.

Prvním mechanismem je sledování. Schopnost vysílače detekovat chyby je založena na sledování signálů sběrnice. Každý uzel, který vysílá, také zaznamenává úroveň sběrnice, a tím zjišťuje rozdíly mezi odeslanými bity a přijatými bity. To umožňuje spolehlivou detekci globálních chyb a chyb na místním vysílači.

Druhým mechanismem, v detekci chyb na úrovni bitů, je vkládání bitů. Kódování jednotlivých bitů je testováno na bitové úrovni. Bitová reprezentace CAN používaná kódování „Non Return to Zero“, zkráceně NRZ. Synchronizační hrany jsou generovány pomocí vkládání bitů. To znamená, že po pěti po sobě jdoucích bitech vloží převodník

bit do bitového toku. Tento vložený bit má doplňkovou hodnotu, která je odstraněna v přijímači.

Pokud byla objevena jedna nebo více chyb, pomocí výše uvedených mechanismů, je současný přenos přerušen odesláním chybového hlášení „chyba rámce“. Tím se zabrání přijímat zprávy dalším uzlům, a tak se zajišťuje konzistence dat v celé síti. Po předání chybné zprávy, která byla zrušena, odesílatel se znova automaticky pokusí o přenos. Uzly mohou opět soupeřit o přístup na sběrnici.

Nicméně účinné a efektivní metody popsané výše, nemohou zabránit v případě vzniku vadných uzlů přerušení všech vyslaných zpráv, a to včetně správných zpráv. Pokud se nepřijmou opatření pro vlastní kontrolu, bude takto sběrnicový systém blokován. Proto protokol CAN poskytuje mechanismus pro rozlišení ojedinělé chyby od trvalé chyby a místní selhání uzlu. Provádí se statistické vyhodnocení situace uzlu s cílem rozpoznání uzlu s vlastní chybou a případně vstoupení do provozního režimu, ve kterém zbytek sítě není negativně ovlivněn. Může to vést až k rozdělovacímu uzlu, který se sám vypne pro prevenci ostatních uzlů, neboť chybně detekoval zprávy jako nesprávné. [1]

2.2 Protokol HDLC

„High-Level Data Link Control“, rovněž známý také pod zkratkou HDLC. Komunikační protokol spojové vrstvy, který spadá do druhé vrstvy referenčního modelu OSI.

HDLC je protokol vyvinutý mezinárodní organizací pro normalizaci ISO. Spadá pod ISO normu 3309 a ISO 4335. Tento komunikační protokol je vyžíván po celém světě. Protokol HDLC podporuje jak poloduplexní tak i plně duplexní komunikační spojení. Postupy uvedené v protokolu HDLC jsou navrženy tak, aby umožňovaly synchronní transparentní přenos dat. Výhoda HDLC je ta, že kontrolní informace jsou vždy ve stejné pozici. Bitové vzorky použité pro kontrolu se dramaticky liší od těch v datech, tím dochází ke snížení rizika chyb.

2.2.1 Operační režimy

Komunikační protokol HDLC má tři operační režimy, nebo jinak řečeno módy.

Prvním z nich je režim „Normal Response Mode“, dále jen NRM. Tento režim se používá v nevyvážených konfiguracích. V tomto režimu může sekundární uzel přenášet data pouze na požádání speciálně definovaných instrukcí primárního uzlu. Tento režim se obvykle používá s konfigurací „Point-To-Point“ nebo „Multipoint“. Je povolen pouze jeden primární uzel.

Druhý režim je „Asynchronous Response Mode“, zkráceně ARM. Tento asynchronní režim se také používá v nevyvážených konfiguracích. ARM umožňuje sekundárním uzlům zahájit přenos bez předchozího výslovného oprávnění od primárního uzlu. Tento režim se také používá s konfigurací „Point-To-Point“ i plně duplexní spojení. Sekundárním uzlům umožňuje odeslání rámce asynchronně s ohledem na primární uzel. [3]

Posledním režimem je „Asynchronous Balanced Mode“ zkráceně ABM. Režim

ABM se používá hlavně na plně duplexním spoji „Point-To-Point“ pro komunikaci a připojení uzlů. V tomto případě má každý uzel rovné postavení a roli. Primární uzel i sekundární uzel smí samovolně zahájit vysílání. Tento režim se používá v protokolu známém jako X.25.

2.2.2 Formát rámce

Standardní rámec protokolu HDLC zvládá zpracovat data i kontrolní zprávy. Tento formát rámce je vidět na Obrázku 2.

Obrázek 2: Formát rámce HDLC. [4]

Polem „FLAG“ musí začít každý rámec, a také tímto polem musí končit. Uzel, který je připojen, musí neustále naslouchat sekvenci „FLAG“. Tato sekvence se skládá z osmi bitů a vypadá následovně 0111 1110. Sekvence jsou neustále přenášeny mezi rámci pro udržení aktivního spojení. Další dvě bitové posloupnosti jsou využívány v HDLC jako signály pro uzly. Těmito dvěma bitovými posloupnostmi jsou.

1. Následuje-li po sobě sedm jedniček, ale méně jak patnáct, signalizuje se přerušení signálu. Uzel tímto pozná, že nastala chyba ve spojení.

2. Jeli patnáct, nebo více po sobě jdoucích jedniček, znamená to, že kanál je v nečinném stavu.

Doba mezi přenosem aktuálního rámce se nazývá „interframe time fill“. Tato doba je dosažena nepřetržitým zasíláním pole „FLAG“ mezi rámci.

HDLC nespoléhá na konkrétní kód pro spojení. To znamená, že pozice n-tého oktetu má specifický význam, bez ohledu na ostatní bity v témže oktetu. Pokud oktet obsahuje bitovou sekvenci 0111 1110, ale není pole „FLAG“, protokol HLDC používá techniku zvanou „bit-stuffing“ k rozlišení této posloupnosti bitů z pole „FLAG“. Jakmile vysílač zjistí, že odesílá pět po sobě jdoucích jedniček, vloží nulu za pátou jedničku a tím zabrání vytvoření falešného pole „FLAG“.

Přijímací uzel kontroluje příchozí rámec. Pokud zjistí pět po sobě jdoucích jedniček, zjišťuje následující bit. Pokud následující bit je nula, je odstraněn. Jeli ovšem následující bit jednička, zjišťuje hodnotu osmého bitu. Pokud je osmý bit nula, přijímací uzel zjistil přerušení, nebo byl signál odeslán. Proces pokračuje v kontrolování následujících bitů pro určení vhodných opatření. To je způsob, jakým protokol HDLC dosahuje transparentnosti dat. HDLC se nezabývá konkrétními daty uvnitř datového toku, pouze se snaží udržet bezchybnost polí „FLAG“.

Délka adresy je obvykle 0, 8 nebo 16 bitů, v závislosti na linkové vrstvě protokolu. V porovnání protokol SDLC používá pouze velikost pole adresy 8 bitů. Protokol HDLC je odvozen z protokolu SDLC. Pole adresy identifikuje primární nebo sekundární uzel, a zda se jedná o přenos rámce nebo příjem. Každý uzel má jedinečnou adresu. V

nevyvážené konfiguraci odkazuje pole na sekundární uzel s příkazy i odpověďmi. Vyvážená konfigurace obsahuje rámec s příkazem cílové adresy uzlu s odezvou rámce adresy vysílající stanice. [5]

Protokol HDLC používá řídící pole k určení způsobu řízení procesu komunikace. Toto pole obsahuje příkazy, odpovědi a sekvenci čísel sloužící k zachování odpovědnosti toku dat. Definuje funkce rámce a iniciuje logiku pro kontrolu pohybu mezi vysílajícím a přijímajícím uzlem. Existují tři formáty řídících polí.

1. „Information Transfer“, tento formát se používá k přenosu dat mezi dvěma zařízení.

2. „Unnumbered“, tento formát řídícího pole se používá pro kontrolní účely. Slouží k inicializaci spojení, odpojení a dalších řídicích funkcí.

3. „Supervisory“, tento formát řídícího pole provádí kontrolní funkce, jako je například potvrzení rámců, žádost o opětovné vysílání nebo žádost o dočasné pozastavení přenášení rámců. Jeho použití závisí na použitém operačním režimu.

Sady specifických příkazů a odpovědí v HDLC pro jednotlivé typy řídících polí jsou různé.

Funkce příkazů a odpovědí pro formát „Information Transfer“ je přenášet postupně číslované rámce, které obsahují pole informací datového spojení.

Pro formát „Unnumbered“ se příkazy a odpovědi používají k rozšíření počtu řídicích funkcí. Tento formát obsahuje pět modifikačních bitů, které umožňují rozšíření funkcí až o 32 dalších příkazů a odpovědí. Níže jsou rozepsány nejdůležitější funkce tohoto řídícího formátu.

- Sada „Set Normal Response Mode“, zkráceně SNRM, umístí sekundární uzly do NRM. NRM neumožňuje sekundárním uzlům odesílat nevyžádané rámce. Hlavní uzel má kontrolu spojení.
- Sada „Set Asynchronous Response Mode“, zkráceně SARM, umožňuje sekundárním uzlům přenášet rámce bez dotazování hlavního uzlu.
- Sada „Set Asynchronous Balanced Mode“, zkráceně SABM, nastaví provozní režim spojení na ABM.
- „Disconnect“, zkráceně DISC, umístí sekundární uzly do režimu odpojení.
- Sada „Set Normal Response Mode Extended“ SNRME zvětší velikost řídícího pole o dva oktety namísto jednoho v NRM. Používá se pro rozšířené řazení, které platí i pro SARME a SABME.
- Sada „Set Initialization Mode“, zkráceně SIM, se používá k přizpůsobení sekundárního uzlu, k zahájení specifického postupu inicializovat své řídící funkce.
- „Unnumbered Poll“, zkráceně UP, provádí průzkum uzlů bez ohledu na pořadí nebo potvrzení.
- „Unnumbered Information“, zkráceně UI, se používá k odesílání informací do vedlejších uzlů.

- „Exchange Identification“, zkráceně XID, se používá k přizpůsobení identifikace sekundárního uzlu a poskytnutí identifikačních informací primárnímu uzlu.
- „Reset“, zkráceně RSET, se používá k obnovení stavové proměnné určeného uzlu.
- „Test“, hodnota funkce TEST se používá k přizpůsobení řešení sekundárního uzlu reagovat na odpověď TEST při první příležitosti. Provádí základní test spojení.
- „Unnumbered Acknowledgment“, zkráceně UA. Sekundární uzly používají potvrzení o doručení a přijetí. Zahrnuje příkazy SNRM, SARM, SABM, SNRME, SARME, SABME, RSET, SIM nebo DISC.
- „Disconnected Mode“, zkráceně DM. Označený sekundární uzel je v odpojeném režimu.
- „Request Initialization Mode“, zkráceně RIM, je žádost sekundárního uzlu o inicializaci pro primární uzel. Jakmile sekundární uzel odešle RIM, může pouze reagovat na příkazy SIM, DISC, TEST nebo XID.
- „Request Disconnect“, zkráceně RD. Žádost zaslaná sekundárním uzlem informovat primární uzel. Sekundádní uzel si v žádosti přeje odpojit od provozního režimu NDM nebo ADM.
- „Frame Reject“, zkráceně FRMR, používá sekundární uzel v provozním režimu na hlášení, kdy v přenosu rámce došlo k chybě a přenos rámce nelze opravit.

Příkazy a odpovědi řídícího pole „Supervisory“ jsou používány k provádění číslovaných kontrolních funkcí, jako je například potvrzení, dotazování, dočasné pozastavení přenosu informací, nebo chyba obnovení. Rámec řídícího pole s formátem S nemůže obsahovat pole informací. Primární uzly mohou využívat rámec s formátem S a P bitem nastavený na hodnotu jedna. Požadují odpověď od sekundárního uzlu. Nejdůležitější příkazy a odpovědi jsou definovány následovně:

- „Receive Ready“, zkráceně RR, se používá v primárním nebo sekundárním uzlu, který označuje připravenost přijímat rámce nebo potvrzení již dříve přijatých rámců.
- „Receive Not Ready“, zkráceně RNR, se používá k označení nepřipravenosti přijímat jakékoliv rámce nebo potvrzení. Opět používané pro primární nebo sekundární uzly.
- „Reject“, zkráceně REJ, se používá k požadavku opakovaného přenosu rámců.
- „Selective Reject“, zkráceně SREJ. Uzel požaduje přenos konkrétních rámců. SREJ musí být předán pro každý chybný rámec. Každý rámec je považován jako samostatná chyba. Jen jeden SREJ může být vyslán v jednom okamžiku.

Protokol HDLC nemusí vždy obsahovat datové pole. Datové pole obsahuje pouze při přenosu informací, využívající v řídícím poli. Toto datové pole obsahuje aktuální data odesílatele, která posílá k příjemci. [5]

Pole kontrolního součtu „Frame Check Sequence“, zkráceně FCS, obsahuje 16bitovou nebo 32bitovou cyklickou redundantní kontrolu. Pole FCS se využívá pro detekci chyb.

2.3 Protokol Modbus

Tento standard komunikačního protokolu je velmi dobře adaptivní, flexibilní a lze jej snadno implementovat na různá rozhraní mnoha výrobců. Zároveň se díky použitým licencím jedná o otevřený protokol, takže je možné ho upravovat, a tím přizpůsobit požadovaným nárokům. Tím se odlišuje od proprietárních protokolů.

Modbus má své kořeny v pozdních sedmdesátých letech minulého století. Protokol Modbus je sériový komunikační protokol vydaný v roce 1979 firmou Modicon – nyní působící pod firmou Telemecanique Schneider Electric. Komunikační rozhraní Modbus je založeno na architektuře typu master a slave. Komunikace mezi uzly Modbus je dosaženo zprávami. Protokol Modbus je otevřený standard, který popisuje strukturu zpráv. Fyzická vrstva rozhraní Modbus může fungovat na původním rozhraní RS-232, ale v nejnovějších implementacích je většina použita na rozhraní RS-485. Toto rozhraní RS-485 umožňuje využít delší vzdálenosti na sběrnici a vyšší rychlosti. Za krátkou dobu se Modbus stal standardem průmyslových komunikačních sítí. [6]

2.3.1 Struktura zprávy Modbus

Komunikační protokol Modbus je postaven na zprávách. Formát těchto zpráv Modbus je nezávislý na použitém typu fyzického rozhraní. Rozhraní RS232 používá stejné zprávy jako na Modbus TCP přes ethernet. To dává rozhraní Modbus velmi dlouhou životnost. Bez ohledu na typ připojení lze použít stejný protokol. Z tohoto důvodu Modbus dává možnost snadno inovovat hardware v síťové struktuře, bez nutnosti velké změny v softwaru. Zařízení může komunikovat s několika uzly Modbus, i když jsou spojeny s různými typy rozhraní, aniž by bylo nutné použít jiný protokol pro každé připojení.

Na rozhraní RS485 nebo RS232 odesílá Modbus zprávy v jednoduché formě. V tomto případě je síť určena pro Modbus. Při použití univerzálnějších systémů v síti jako je protokol TCP/IP v síti ethernet, zprávy Modbus jsou uloženy v paketech s formátem nezbytným pro fyzické rozhraní. V takovém případě Modbus a další typy připojení mohou být nastaveny společně na stejném fyzickém rozhraní ve stejnou dobu. Přestože komunikace probíhá převážně typem „peer-to-peer“ (doslovně řečeno rovný s rovným, je označení typu síťové komunikace, ve které probíhá přenos přímo mezi uzly), zprávy Modbus jsou schopny fungovat i na typu sítě „point-to-point“ (síťová komunikace typu bod-bod, využívá přímé spojení mezi uzly) a „multidrop“ (vícebodového propojení uzlů na jednom okruhu, jeden uzel vysílá data, ostatní uzly naslouchají).

Každá zpráva Modbus má stejnou strukturu a obsahuje čtyři základní prvky, jak je vidět v Obrázku 3.

Obrázek 3: Formát zprávy Modbus. [7]

Pořadí těchto prvků je stejný pro všechny zprávy, aby bylo možné snadněji analyzovat obsah zprávy Modbus.

Pole zprávy „Address“ určuje adresu příjemce. Pole „Function code“, zkráceně FC, definuje typ kódu zprávy. Pole „Data“ obsahuje blok dat s dalšími informacemi. Poslední pole zprávy Modbus „Error check“ slouží v komunikaci pro kontrolu chyb.

Komunikaci zahajuje vždy uzel master v síti Modbus. Master odešle zprávu a v závislosti na obsahu zprávy, slave přijímá opatření a reaguje. Může existovat více uzlů master v síti. Pole zprávy „Address“ řeší a definuje, které zařízení by mělo reagovat na vyslanou zprávu. Všechny ostatní uzly v síti zprávu ignorují, pokud adresní pole neodpovídá jejich vlastní adrese. [7]

2.3.2 Dva režimy protokolu Modbus

Sériové připojení Modbus používá dva základní přenosové režimy. Prvním z nich je režim ASCII a druhý režim se nazývá RTU (Remote Terminal Unit). Režim přenosu v sériové komunikaci definuje způsob, jakým jsou zprávy Modbus kódovány. Zprávy v režimu ASCII jsou v čitelném formátu ASCII („American Standard Code for Information Interchange“, jde o kódovou tabulku, která přesně definuje znakovou sadu). Zprávy v režimu RTU využívají binární kódování. Zpráva se tak stává nečitelná při sledování oproti zprávě v režimu ASCII. Zprávy v režimu RTU mají menší velikost, což umožňuje další výměnu dat ve stejném časovém úseku. Všechny uzly v jednom segmentu sítě musí používat stejný režim sériového přenosu. Zařízení nakonfigurované pro použití v režimu ASCII nemohou překládat zprávy pro režim RTU a naopak.

Při použití Modbus v režimu ASCII jsou všechny zprávy kódovány v šestnáctkové soustavě, s čitelnými znaky ASCII. Pouze znaky 0-9 a A-F se používají v režimu ASCII. Pro komunikaci je pro každý bajt informací zapotřebí dvou bajtů, protože každý bajt komunikace může definovat pouze 4 bity v šestnáctkové soustavě. Při použití režimu RTU jsou informace vyměňovány v binární formě, kde každý bajt informace je v komunikaci kódován jedním bajtem.

Zprávy Modbus jsou odesílány ve speciálním formátu rámce. Přijímačům zprávy tím usnadňuje způsob, jak zjistit začátek a konec zprávy. Při použití režimu ASCII se k zahájení a ukončení rámce používají speciální znaky. K signalizaci začátku zprávy se používá dvojtečka „:“ a každá zpráva je ukončena kombinací CR a LF. Zkratka CR značí „Carriage return“ - speciální řídící znak, který posune ukazatel na začátek řádku. Zkratka LF značí „Line feed“ - speciální řídící znak, který posune ukazatel na další řádek. Režim RTU při komunikaci používá časové mezery mezi jednotlivými rámci zprávy. Každou zprávu musí předcházet časová mezera o minimální délce 3,5 znaků. Zjistí-li přijímač mezeru nejméně 1,5 znaků, poté předpokládá, že přichází nová zpráva a vymaže vyrovnávací paměť. Hlavní výhodou režimu ASCII je povolená mezera mezi zprávami s maximální délkou 1 sekundy. V režimu RTU je nutné každou zprávu odeslat jako souvislý proud. [7]

2.3.3 Vlastnosti režimu ASCII

Přehled základních vlastností režimu ASCII protokolu Modbus.

Tabulka 1: Režim ASCII

Znaky	Čísla 0-9 a písmena A-F
Kontrola chyb	Metodou LRC (Longitudinal Redundancy Check)
Znak začátku zprávy	Znak dvojtečka „:“
Znak konce zprávy	Kombinace CR a LF
Mezery mezi zprávami	Jedna sekunda
Počet bitů start	Jeden bit
Počet bitů dat	Sedm bitů
Počet bitů stop	Jeden bit

2.3.4 Vlastnosti režimu RTU

Přehled základních vlastností režimu ASCII protokolu Modbus.

Tabulka 2: Režim RTU

Znaky	Pouze binární hodnoty 0-255
Kontrola chyb	Metodou CRC (Cyclic Redundancy Check)
Znak začátku zprávy	Minimální délka 3,5 znaků
Znak konce zprávy	Minimální délka 3,5 znaků
Mezery mezi zprávami	Délka 1,5 znaků
Počet bitů start	Jeden bit
Počet bitů dat	Osm bitů
Počet bitů stop	Jeden bit

2.3.5 Adresování v Modbus

První informace, v každé zprávě Modbus, je adresa příjemce. Tento parametr obsahuje jeden bajt informací. Platné adresy jsou v rozmezí 0 až 247. Hodnoty 1 až 247 jsou přiřazeny k jednotlivým zařízením v síti a hodnota adresy 0 se používá pro všesměrové vysílání. Zprávy odeslané na tuto adresu 0, budou přijímat všechna zařízení. Tyto zařízení slave nikdy neodpoví na všesměrové vysílání zprávy Modbus. Na zprávy, odeslané na hodnotu adresy 1 až 247, odpovídá zařízení slave na stejnou adresu uvedenou v požadavku od zařízení master. Tímto způsobem zařízení master vidí, že zařízení slave skutečně reagoval na vyslaný požadavek.

V rámci zařízení Modbus jsou uživatelské registry, vstupy a výstupy řazeny číselně. Ve zprávách Modbus se nepoužívají stejné adresy pro nastavení hodnot nebo čtení. Ve zprávách se používají adresy s posunutou hodnotou o jedna. Má-li se načíst hodnota výstupu číslo 18, v požadavku zprávy Modbus se musí zadat hodnota číslo 17. Pro vstup, evidenci a posun registrů musí být z adresy zařízení odečtena správná adresa do struktury zprávy Modbus. To vede k mnoha chybám a tento problém je řešen při navrhování aplikací pro Modbus. [8]

2.3.6 Kódy funkcí Modbus

Druhý parametr v každé zprávě Modbus je kód funkce. To určuje typ zprávy a druh činnosti vyžadovaný zařízením slave. Parametr obsahuje jeden bajt informací. Platné

kódy funkcí jsou v rozmezí 1 až 255. Všechna zařízení Modbus nedokáží rozpoznat stejnou sadu kódů funkcí. Nejčastěji používané kódy jsou popsány v tabulce 3.

Tabulka 3: Kódy funkcí Modbus

Kód	Název funkce
01	Read coil status
02	Read input status
03	Read holding registers
04	Read input registers
05	Force single coil
06	Preset single register
07	Read exception status
15	Force multiple coils
16	Preset multiple registers
17	Report slave ID

Obvykle když zařízení slave v Modbus odpovídá na odpovědi, používá stejný kód funkce jako v požadavku. Nicméně když je zjištěna chyba, je kód funkce nejvyššího bitu zapnut. Zařízení master tak vidí rozdíl mezi úspěšnými a neúspěšnými odpovědi.

3 ŠIFROVACÍ METODY

Věda zabývající se šifrováním, tedy utajováním informací, se nazývá kryptografie. Naproti tomu věda, která se zabývá luštěním šifer je kryptoanalýza. Nadřazeným pojmem pro oba dva obory je kryptologie. Šifrujeme tak, aby se útočníkovi nevyplatilo šifru prolomit. Náklady útočníka na dešifrování by měly být větší než zisk, který by útočník měl v případě úspěšného dešifrování.

Základní rozdělení šifer je na symetrické, asymetrické a klasické metody. V současné době se využívají metody symetrické a asymetrické.

Symetrické šifry se používají pro šifrování i dešifrování stejným soukromým klíčem, který by měl znát pouze odesílatel a adresát, což je zároveň jeho největší slabina. Ostatním by měl být tento klíč utajen. Je nutné, aby se příjemce i odesílatel dohodli na jednom klíči, který si musí nějakým bezpečným způsobem vyměnit a který budou znát pouze oni dva. Kromě problému distribuce klíče má tento typ šifrování další nevýhodu a tou je počet účastníků komunikace. Jeli počet komunikujících stran malý, příliš to nevadí, ale pokud je počet komunikujících větší a chceme mít pro každou dvojici komunikujících stran jiný klíč, potřebujeme pro n účastníků $\frac{n\cdot(n-1)}{2}$ klíčů. Symetrické šifrování je mnohem jednodušší, než asymetrické šifrování. Jednak nepotřebuje tak výkonné počítače a jednak je jednodušší jeho princip. Symetrické kódy mají jako hlavní výhodu rychlost algoritmu. Velkým problémem je zde právě vymýšlení, distribuce a utajování klíčů. Používané šifry tohoto typu jsou AES, BlowFish, CAST, DES, Triple DES, RC2, RC4, RC5, IDEA a mnoho dalších metod. [9]

Asymetrické šifry používají jiný klíč pro šifrování než pro dešifrování. Klíčem pro šifrování nelze dešifrovat. Každý účastník komunikace má dva klíče. První z klíčů je veřejný a druhý klíč je soukromý. Cokoliv je zašifrováno jedním klíčem, lze dešifrovat pouze druhým klíčem a naopak. Velkou výhodou tohoto přístupu je, že jeden z klíčů může být zveřejněn a k dispozici komukoliv. K zašifrování zprávy se použije tento veřejný klíč. Schopen dešifrovat zprávu bude pouze držitel soukromého klíče. Při použití jiného klíče, který nepatří ke klíči, kterým se šifrovalo, dostane útočník nesmyslnou zprávu. Při komunikaci více účastníků je potřeba celkem $2 \cdot n$ klíčů, tedy počet přímo úměrný počtu účastníků n. Asymetrické šifrování má jednu velkou nevýhodu a tou jsou velmi náročné matematické operace. Tyto šifry jsou sice značně pomalejší než symetrické, ale odpadají některé problémy s klíči. Klíč pro šifrování nemusí být utajován, naopak může být zcela veřejný. Používanými šiframi jsou kryptografie nad eliptickými křivkami, kryptografie s Lucasovými funkcemi, RSA, Digital Signature Algorithm nebo Diffie-Hellmanův algoritmus. [10]

V praxi se používají symetrické i asymetrické šifry. Asymetrické šifry se používají zejména pro šifrování klíčů k symetrickým šifrám. Vlastní velké objemy dat se pak šifrují symetricky, protože symetrické šifry jsou rychlejší.

3.1 Diffie-Hellmanův protokol

Protokol byl poprvé vydán autory Whitfield Diffie a Martin Hellman v roce 1976.

Protokol Diffie-Hellmanův je specifická metoda pro výměnu klíčů po veřejných nezabezpečených sítí. Je to jedna z nejstarších praktických příkladů výměny klíčů implementována v oblasti kryptografie. Metody výměny klíčů protokolu Diffie–Hellman umožňuje dvou stranám, bez předchozí komunikace, vytvořit navzájem společně sdílený tajný klíč přes nezabezpečený komunikační kanál. Tento klíč lze pak použít k šifrování další komunikace pomocí symetrického šifrování.

Počet účastníků není omezen. Výhodou tohoto protokolu je, že naslouchající útočník komunikace tento klíč nemůže nijak zachytit, i když se jedná o nezabezpečený komunikační kanál. Klíč je sestaven dílčími informacemi všemi účastníky. Vytvořený kompletní klíč není nikdy zaslán po síti. Tento klíč je vytvořen na každé straně jednotlivého účastníka. Bezpečnost vytvořeného klíče je dána složitým výpočtem diskrétního logaritmu. [11]

Nevýhodou tohoto protokolu je, že neumožňuje autentizaci přenášených zpráv mezi jednotlivými účastníky komunikace. Slabinu protokolu využívá útoku „Man in the middle“.

Tento protokol bez kombinace s jinými bezpečnostními algoritmy je vhodný pouze v komunikaci, kde útočník nemůže aktivně zasahovat do komunikace.

3.1.1 Postup sestavení klíče

Nejjednodušší a originální implementace protokolu používá multiplikativní skupinu celých čísel s modulem *p*, kde *p* je prvočíslo a číslo *g* je primitivní kořen tvořen generátorem. Číslo *g* nemusí být nikterak veliké, stačí pouze dvě maximálně pět cifer. Strany si zvolí náhodné číslo, které přidají do rovnice výpočtu, a vydělením celé rovnice modulem *p* vznikne výsledek, který je zbytkem tohoto dělení.

Pro lepší pochopení principu výpočtů, je uveden jednoduchý příklad výpočtu klíčů. Odesílatel a příjemce se dohodli využít prvočíslo p = 97 a vygenerované číslo g = 17. Odesílatel se rozhodne pro tajné číslo a = 12 a potom odesílá příjemci A = $g^a \ mod \ p$.

A = $17^{12} \ mod$ 97 = 582622237229761 mod 97 = 64.

Příjemce se rozhodne pro tajné číslo b = 10 a potom odesílá odesílateli B = $g^b \ mod \ p$.

B = $17^{10} \ mod$ 97 = 2015993900449 mod 97 = 65.

Odesílatel vypočítává tajný klíč $k \ = B^a \ mod \ p$.

k = $65^{12} \ mod$ 97 = 5688009063105712890625 mod 97 = 22.

Příjemce vypočítává tajný klíč $k´ \ = A^b \ mod \ p$.

$k´$ = $64^{10} \ mod$ 97 = 1152921504606846976 mod 97 = 22.

Výpočty jsou bezchybné, jelikož odesílatel a příjemce došli shodně na stejný sdílený tajný klíč, kterým je číslo 22. Nyní tento tajný klíč mohou použít jako šifrovací klíč pro odesílání zpráv přes nezabezpečený komunikační kanál. Hodnoty a a b jsou tajné a nesmí být prozrazeny. Ostatní hodnoty rovnice jsou odeslány nezabezpečeným

komunikačním kanálem. V reálném prostředí jsou cifry prvočísla a tajných čísel samozřejmě mnohem větší, čím se zajistí větší bezpečnost.

3.1.2 Útok „Man in the middle“

Jak již bylo zmíněno, protokol Diffie–Hellman je náchylný k útoku jménem „Man in the Middle“. Jeho princip spočívá v tom, že útočník se vydává za příjemce či odesílatele. Protokol Diffie–Hellman nemá žádný nástroj, jak ověřit pravost komunikujících stran a proto je velmi snadné napadnout protokol útokem „Man in the Middle“. Odesílatel nemá ponětí, že komunikuje s útočníkem a stejně tak i příjemce na druhé straně netuší, že svá data odesílá útočníkovi. Útočník si zvolí své tajné číslo, jako kdyby to byl normální účastník komunikace. Prvočíslo p s generovaným číslem g odposlechne po nezabezpečeném komunikačním kanálu. Poté si stačí vyměnit klíč s odesílatelem a příjemcem a tak kontrolovat přenos dat, popřípadě může i modifikovat obsah dat. Tento útok je zobrazen a podrobně rozebrán v následujícím příkladě.

Odesílatel a příjemce se dohodli využít prvočíslo p = 197 a vygenerované číslo g = 23.

Odesílatel se rozhodne pro tajné číslo a = 12 a potom odesílá příjemci A = $g^a \ mod \ p$.

A = $23^{12} \ mod$ 197 = 21914624432020321 mod 197 = 188.

Příjemce se rozhodne pro tajné číslo b = 8 a potom odesílá odesílateli B = $g^b \ mod \ p$.

B = $23^8 \ mod$ 197 = 78310985281 mod 197 = 154.

Útočník se rozhodne pro tajné číslo c = 10 a potom odesílá odesílateli i příjemci C = $g^c \ mod \ p$.

C = $23^{10} \ mod$ 197 = 41426511213649 mod 197 = 105.

Odesílatel vypočítává tajný klíč $k \ = C^a \ mod \ p$.

k = $105^{12} \ mod$ 197 = 1795856326022129150390625 mod 197 = 40.

Útočník vypočítává tajný klíč $k_1 \ = A^c \ mod \ p$.

k_1 = $188^{10} \ mod$ 197 = 55154187683317729460224 mod 197 = 40.

Příjemce vypočítává tajný klíč $k´ \ = C^b \ mod \ p$.

$k´$ = $105^8 \ mod$ 197 = 14774554437890625 mod 197 = 37.

Útočník vypočítává tajný klíč $k_2 \ = B^c \ mod \ p$.

k_2 = $154^{10} \ mod$ 197 = 7502520803928269464576 mod 197 = 37.

Výpočty a hodnoty jednotlivých klíčů jsou totožné $k = k_1$ a druhý klíč $k´ = k_2$, útočník nyní může šifrovat i dešifrovat data odeslaná jak odesílatelem, tak i příjemcem. V danou chvíli se odesílatel i příjemce domnívají, že oba dva mají totožný klíč a přenos probíhá utajeně. V praxi je proto většinou nutné použít autentizaci účastníků komunikace. Zpravidla se využívá pro autentizaci účastníků digitální podpis s asymetrickým algoritmem RSA.

3.2 Digitální podpis

Digitální podpis představuje elektronickou variantu vlastnoručního podpisu na tištěném dokumentu. Dokáže ověřit autenticitu neboli záruku, že autorem je skutečně ten, kdo se za autora vydává. Umožňuje také ověřit integritu určitých digitálních dat. Zajišťuje pomocí kryptografických metod a dohledu nezávislé třetí strany certifikační autority identifikaci a nepopiratelnost původu dokumentů a digitálních dat v prostředí internetu. Využívá se nejčastěji pomocí technik asymetrické kryptografie RSA v kombinaci s matematickou funkcí hash. [12]

Funkce hash je algoritmus převodu jakýchkoliv vstupních dat na stejně dlouhý výstup v hexadecimální soustavě. Algoritmus hashe je geniální v tom, dojde-li byť k malé změně vstupních dat, na výstupu dojde k velké změně výsledného hashe. Ekvivalent k hash se někdy česky nazývá otisk.

Asymetrický systém RSA je postaveno na obtížnosti rozložení na násobky velkých čísel, faktorizaci. Rozložit velké číslo na součin prvočísel je velmi obtížná úloha. Z čísla n je tedy v rozumném čase prakticky nemožné zjistit prvočísla p a q, neboť není znám žádný algoritmus faktorizace, který by pracoval v polynomiálním čase vůči velikosti zápisu veřejného čísla n. Naproti tomu násobení dvou velkých čísel je velmi jednoduchá úloha.

3.2.1 Postup sestavení digitálního podpisu

Jsou zvolena vybraná digitální data například dokument, zpráva nebo soubor. Dále se zvolí jedna z matematických funkcí hash verzi MD5 nebo SHA-1.

Hash MD5 je nejznámější a nejpoužívanější hashovací funkcí. Tento hash MD5 má 128 bitový výstup. Hash verze SHA-1 má již 160 bitový výstup a princip této funkce je založen na velmi podobném principu hashe MD5. Existuje mnoho dalších hash funkcí.

Po zvolení funkce hash MD5 se spočítá hodnota hashe ze zvolených digitálních dat. Vznikne unikátní hexadecimální číslo, které se dále zašifruje algoritmem RSA.

Generátorem klíčů s využitím algoritmem RSA vygenerují dvě prvočísla a to prvočísla p a q. Nyní se sestaví výpočty veřejný klíč n a e. Z uvedených hodnot se nalezne soukromý klíč d. Veřejný klíč se zveřejní a tajný klíč se uchová bezpečně skryt. Po tomto kroku následuje šifrování zmíněným algoritmem RSA, vzniklý hash, který vznikl ze spočtené hodnoty hashe s digitálními daty. Vznikl zašifrovaný soubor s délkou 304 bitů s výstupními daty v hexadecimálním tvaru, digitální podpis.

Digitální podpis může být ještě zaručen třetí nezávislou stranou a to certifikační autoritou. Digitální certifikát se ukládá ve formátu X.509. V tomto formátu jsou uloženy veškeré informace o majiteli certifikátu i tvůrci certifikátu. Na základě toho principu může být důvěřováno cizím certifikátům, jelikož byl schválen a vydán certifikační autoritou. Pro zaručený digitální podpis bude nutné vyměnit si s asymetrickým šifrováním RSA bitovou délku 304 bitů, veřejný klíč n a veřejný klíč e. Poté vyplnit požadované parametry certifikátu a stvrdit certifikát kódem PIN. Po vytvoření a ověření certifikátu se může nyní vygenerovat digitální podpis a příslušný zašifrovaný soubor zaručeně digitálně podepsat. Názorný příklad pro lepší pochopení sestavení digitálního podpisu.

Odesílatel si zvolí požadovaný dokument nebo soubor, který má být digitálně podepsán a odeslán odesílateli.

Zvolí si matematickou funkci hash MD5, která vypadá například následovně.

30 20 30 0C 06 08 2A 86 48 86 F7 0D 02 05 05 00 04 10

Tímto hashem a zdrojovým souborem vygeneruje výsledný výstup se stejnou velikostí.

E7 58 2D 9C 71 D5 DA 11 71 29 3E F2 3F CD 21 DA

Generátorem klíčů se vygenerují prvočísla *p* a *q*.

$p = 11$

$q = 13$

Z vygenerovaných čísel se vypočítá veřejný klíč *n* vynásobením prvočísel *p* a *q*.

$n = p \cdot q = 143$

Následně se vypočte veřejný šifrovací exponent $\varphi(n)$ z prvočísel *p* a *q*.

$\varphi(n) = (p-1) \cdot (q-1) = 120$

Dále je vypočítán veřejný klíč *e*.

$e = 2^{16} + 1 = 65537$

Z těchto vypočítaných hodnot je následně dopočítán soukromý klíč *d*.

$d \cdot e \equiv 1 \left(mod\ \varphi(n)\right) = d \cdot 7 \equiv 1\ (mod\ 120) => d = 103$

Dojde k vytvoření certifikátu s parametry majitele digitálního podpisu vydaného certifikační autoritou. Můžou zde být informace o jménu, adrese, názvu nebo identifikační číslo. Tyto informace stvrdí PIN kódem. K vytvoření certifikátu je ještě potřeba bitová délka RSA parametru a veřejné klíče *n* a *e*. Uveden pouze zkrácený a neautorizovaný výpis certifikátu.

```
Version:                2 (X.509v3-1996)
SubjectName:            CN=Jaroslav Procházka [1334612574],
                        DC=cryptool, DC=org
IssuerName:             CN=CrypTool CA 2, DC=cryptool,
DC=org
SerialNumber:           04
Validity - NotBefore:   Mon Apr 16 23:43:08 2012
(120416214308Z)
NotAfter:               Tue Apr 16 23:43:08 2013
(130416214308Z)
Public Key Fingerprint:    7789 7C97 B27A 136A 76E9 5DB3
ACF2 D894
SubjectKey:             Algorithm rsa (OID 2.5.8.1.1),
Keysize = 512
                Public modulus (no. of bits = 301):
                  0  1B4C7E4F 0EC3C7AB  4C9BD6B3 2627F4C5
                 10  1FCEBE34 C29683BA  54762426 86C45A52
                 20  488B87BE 6537
                Public exponent (no. of bits = 17):
                  0  010001
```

Nyní dojde k zašifrování algoritmem RSA výsledný hash se zdrojovým souborem pomocí veřejného klíče *n* a soukromého klíče *d*.

0A 0D BB 12 5C DD FA D0 39 73 C9 22 13 84 0F 96 A3 7F DB D1 8B FC 4D C8 3F 73 B8 99 2E 8C 95 8A 13 A2 11 CD 17 0F

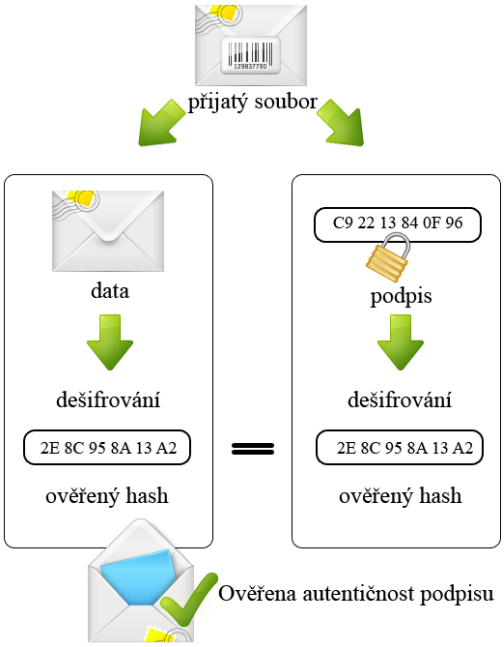
Tento výsledný zašifrovaný hash spolu s vytvořeným certifikátem vygeneruje digitálně podepsaný soubor. Hlavička tohoto digitálně podepsaného souboru vypadá následovně.

```
00000  53 69 67 6E 61 74 75 72 65 3A 20 20 20 20 20 20 0A Signature:       .
00012  0D BB 12 5C DD FA D0 39 73 C9 22 13 84 0F 96 A3 7F DB .».\ÝúĐ9sÉ"....Ł.Ű
00024  D1 8B FC 4D C8 3F 73 B8 99 2E 8C 95 8A 13 A2 11 CD 17 Ń.üMČ?s¸.....˘.Í.
00036  0F 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 53 .                 S
00048  69 67 6E 61 74 75 72 65 20 6C 65 6E 67 74 68 3A 20 20 ignature length:
0005A  33 30 34 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 304
0006C  20 41 6C 67 6F 72 69 74 68 6D 3A 20 20 20 20 20 20 52 Algorithm:      R
0007E  53 41 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 SA
00090  48 61 73 68 20 66 75 6E 63 74 69 6F 6E 3A 20 20 20 4D Hash function:   M
000A2  44 35 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 D5
000B4  4B 65 79 3A 20 20 20 20 20 20 5B 50 72 6F 63 68 E1 7A Key:      [Procház
000C6  6B 61 5D 5B 4A 61 72 6F 73 6C 61 76 5D 5B 52 53 41 2D ka][Jaroslav][RSA-
000D8  33 30 34 5D 5B 31 33 33 34 36 31 32 35 37 34 5D 20 20 304][1334612574]
000EA  20 20 20 20 20 20 20 20 20 20 20 20 20 20 4D 65 73 73               Mess
000FC  61 67 65 3A 20 20 20 20 20 20                         age:
```

Tímto stylem je poté zašifrován celý soubor nebo dokument s digitálním podpisem, který je odeslán příjemci po nezabezpečeném komunikačním kanálu.

3.2.2 Ověření digitálního podpisu

Příjemce přijme digitálně podepsaný soubor a při otevíraní dochází k procesu ověření důvěryhodnosti údajů certifikátu. Proces je velmi přehledně znázorněn na následujícím ilustračním Obrázku 4.



Obrázek 4: Ověření digitálního podpisu [13]

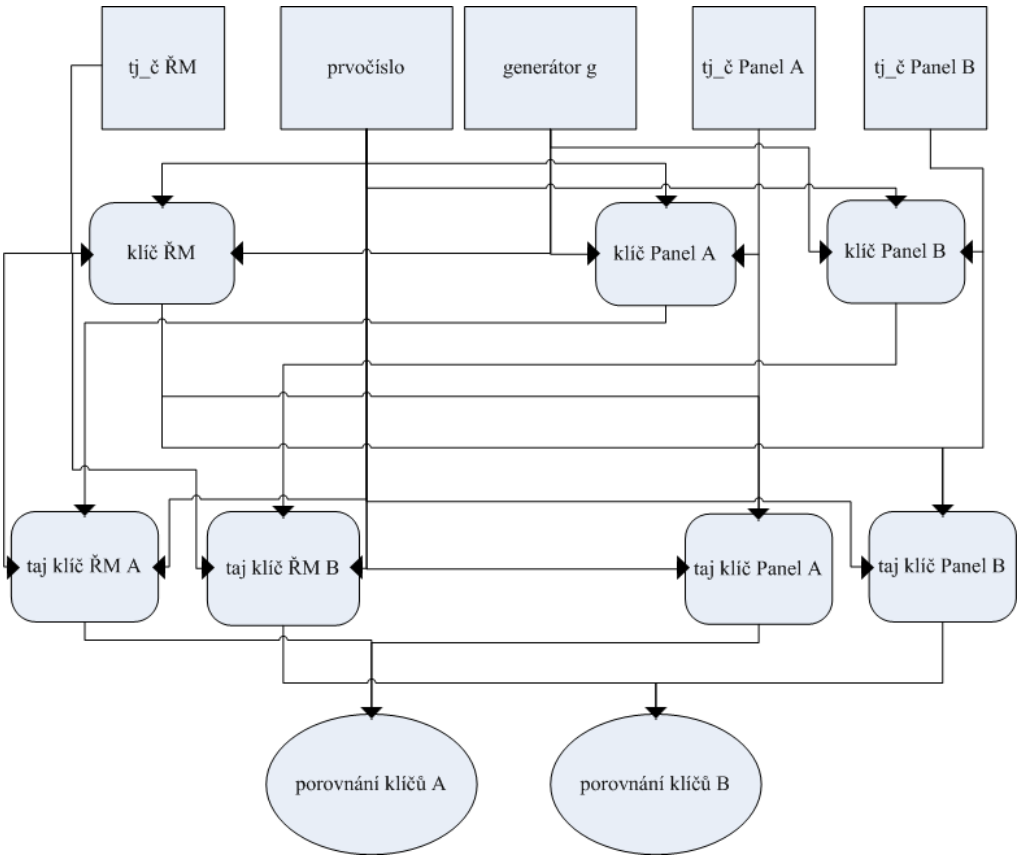
Digitálně podepsaný soubor se rozdělí na zašifrovaná data a digitální podpis. Digitální podpis je nyní dešifrován pomocí veřejného klíče *e* a veřejného klíče *n*. Dešifrováním dostává příjemce údaj hash, který se musí porovnat s údajem hashe přijatého souboru. Příjemce použije stejný hashovací algoritmus na přijatý zdrojový soubor a výsledkem je údaj hashe z tohoto souboru. Porovnáním obou hash kódů se zjistí autentičnost ověření výsledků. Z ověření stejných hash kódů je zajištěna důvěryhodnost odeslaného, digitálně podepsaného souboru, u kterých lze důvěřovat pravdivosti údajů v certifikátu vydaný certifikační autoritou. Platnost digitálního podpisu zaručuje, že se nikdo nepokusil padělat digitální podpis, nebo nedošlo ke změně přenášeného čísla pro protokol Diffie-Hellman, či cokoliv co by způsobilo změnu přenášených dat.

4 SIMULACE PROJEKTU

Navržený koncept modulu pro zabezpečený přenos komunikace mezi zobrazovacím panelem a terminálem je proveden v simulačním prostředí CrypTool. Verze sestavení programu při simulaci návrhu konceptu byla 2.0.4703. Simulace byla prováděna na operačním systému Windows 7. V práci jsou uvedeny pouze blokové přehledy simulací. Tyto blokové přehledy jsou přehlednější než navržené simulace v programu CrypTool, které by byly nepřehledné. Navržená schémata simulace jsou uvedena na konci práce v příloze. Zdrojové soubory se simulacemi jsou v příloze na přiloženém CD.

4.1 Simulace protokolu Diffie-Hellman

Navržená simulace a odzkoušení funkčnosti protokolu Diffie-Hellman je pro řídící modul a dva zobrazovací panely. Celkový blokový přehled navržené simulace je zobrazen na Obrázku 5.

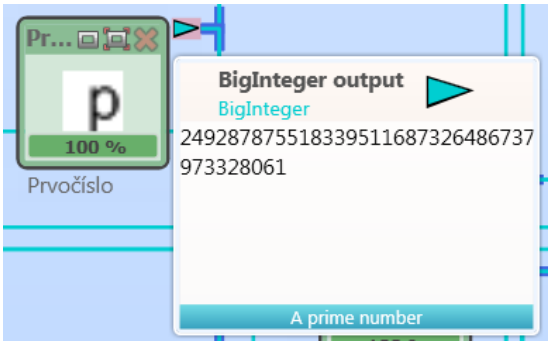


Obrázek 5: Celkový přehled simulace protokolu Diffie-Hellman

Generátor náhodných čísel označený *generátor g* slouží pro vygenerování primitivního čísla. Tyto generované číslice nemusí být nikterak příliš velké. Stačí pouze kombinace dvou až pěti ciferního čísla, které je poté zakombinováno v rovnici pro výpočet veřejného klíče dvou panelů i řídícího modulu.

Generátor prvočísla je pojmenovaný *prvočíslo*. Nachází se v simulaci hned vedle

generátoru primitivního čísla *g*. Vlevo, druhá řada shora. Zde se generují patřičně dost velká prvočísla, z důvodu bezpečnosti. Čím větší prvočíslo je zde vygenerované, tím je větší zabezpečení a nemožnost prolomení číselné kombinace. Modul *prvočíslo* s vygenerovaným prvočíslem z navržené simulace programu je zobrazen na Obrázku 6. Toto prvočíslo je použito ve všech výpočtech veřejných klíčů panelů i řídícího modulu. Prvočíslo je použito i pro výpočty tajných klíčů panelů a řídícího modulu.



Obrázek 6: Modul prvočíslo s generovaným číslem

Řídící modul je označen v simulaci *taj_č ŘM*, který je umístěn ve spodní části úplně vlevo. Tímto řídícím modulem se dále ovládají zobrazovací panely, kterým se odesílá zdrojové instrukce. V simulaci je zobrazen modul, který vyšle tajné číslo řídícího modulu, aby bylo možné se spojit s panely zobrazující informace. Tajný klíč řídícího modulu je poslán do modulu s názvem *klíč ŘM* a do dvou dalších modulů *taj klíč ŘM A* a *taj klíč ŘM B*, kde se později vypočtou a ověří tajné klíče panelů A a B.

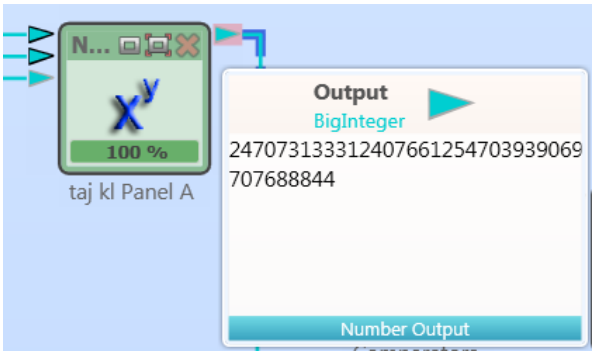
V modulu *klíč ŘM* se generuje vlastní klíč řídícího modulu. Tento klíč řídícího modulu je sestaven z čísel generátoru s názvem *generátor g* a generátoru prvočísel nazvaný v simulaci *prvočíslo*. Dále je zde zasláno tajné číslo z modulu *tj_č ŘM*. Kombinací těchto čísel se vypočte veřejný klíč řídícího modulu, který je poté zaslán potřebným panelům.

Zobrazovací panely jsou zde označeny pod názvy *tj_č Panel A* a *tj_č Panel B*. Panel A se nachází vlevo, úplně nahoře v okně simulace. Panel B je umístěn pod generátorem čísel *g*. Pro tyto panely se zvolí rozdílná tajná čísla. Tajná čísla zobrazovacích panelů slouží pro výpočet veřejných klíčů, který se provede pro každý panel zvlášť v modulu *klíč Panel A* a pro tajné číslo panelu B se provede výpočet v modulu *klíč Panel B*. Dále je tajné číslo zobrazovacích panelů použito i pro sestavení tajného klíče. Tajný klíč je vypočítán pro každý zobrazovací panel zvlášť v modulech *taj klíč Panel A*, pro zobrazovací panel A a pro panel B je to modul pojmenovaný *taj klíč Panel B*.

Modul *klíč Panel A* a *klíč Panel B* představují modul pro výpočet veřejného klíče zobrazovacích panelů. Oba dva moduly fungují na stejném principu výpočtu veřejného klíče, který je vypočítán z čísel *generátor g*, z generátoru *prvočíslo* a tajného čísla příslušejícího zobrazovacímu panelu. Vypočtené veřejné klíče jsou dále vyměněny s ostatními komunikujícími stranami, které slouží k výpočtu tajného klíče.

Výpočet tajných klíčů probíhá principálně u zobrazovacích panelů nebo u řídícího modulu stejně. Tajné klíče zobrazovacích panelů se vypočítávají v modulech nazvaných *taj klíč Panel A* a druhý modul nazvaný *taj klíč Panel B*. Tajný klíč pro řídící modul se

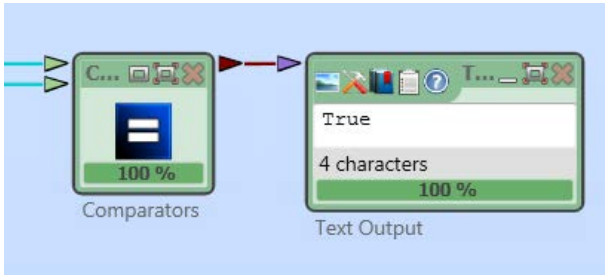
vypočítá v *taj klíč ŘM A* a pro druhý zobrazovací panel v *taj klíč ŘM B*. Stejný parametr pro výpočet všech tajných klíčů u zobrazovacích panelů nebo u řídícího modulu je generované číslo prvočíslo. Prvočíslo je používáno jako modulo pro výpočet veřejného klíče, tak i pro výpočet tajného klíče. Modulo je považováno za celočíselné dělení se zbytkem, kdy výsledkem je právě zmiňovaný zbytek po dělení. Dále je potřeba pro výpočet jednotlivých tajných klíčů unikátní vlastní tajné číslo zobrazovacích panelů a řídícího modulu, které znají pouze příslušné moduly. Poslední číslo do rovnice výpočtu se dosadí z výměny veřejných klíčů. Veřejný klíč z řídícího modulu je zaslán zobrazovacímu panelu *A* pro finální výpočet tajného klíče do modulu *taj klíč Panel A*. Vypočítaný tajný klíč modulu *taj klíč Panel A* ze simulace protokolu Diffie-Hellman je zobrazen na Obrázku 7.



Obrázek 7: Vypočítaný tajný klíč panelu A

Zobrazovací panel *A* si vymění veřejný klíč s řídícím modulem a zašle svůj veřejný klíč do modulu *taj klíč ŘM A*. Přesně takový proces proběhne i v případě druhého zobrazovacího panelu B, kde zobrazovací panel zašle veřejný klíč z modulu nazvaného *klíč Panel B* do řídícího modulu *taj klíč ŘM B*, pro výpočet tajného klíče pro tento pár. Pro kompletní výměnu zašle i řídící modul svůj vypočtený veřejný klíč z modulu *klíč ŘM* modulu *taj klíč Panel B*, kde dojde k výpočtu tajného klíče pro panel B.

Nyní se v bloku s názvem *porovnání klíčů A* provede kontrola jednotlivých vypočtených tajných klíčů z modulu *taj klíč Panel A* a řídícího modulu *taj klíč ŘM A*. Modul *porovnání klíčů B* slouží pro porovnání tajných klíčů z řídícího modulu *taj klíč ŘM B* a zobrazovacího panelu *taj klíč Panel B*. Ve výsledném hodnocení se zobrazí, zda porovnání jednotlivých tajných klíčů sobě navzájem odpovídá. Pokud jsou tajné klíče shodné, v zobrazení výsledků dojde k vyhodnocení signalizující textem „True“. Na Obrázku 8 je vidět vyhodnocení simulace srovnání dvou tajných klíčů modulu *taj klíč Panel A* a modulu *taj klíč ŘM A*. Po tomto vyhodnocení dojde k přenosu instrukcí pro zobrazovací panely.

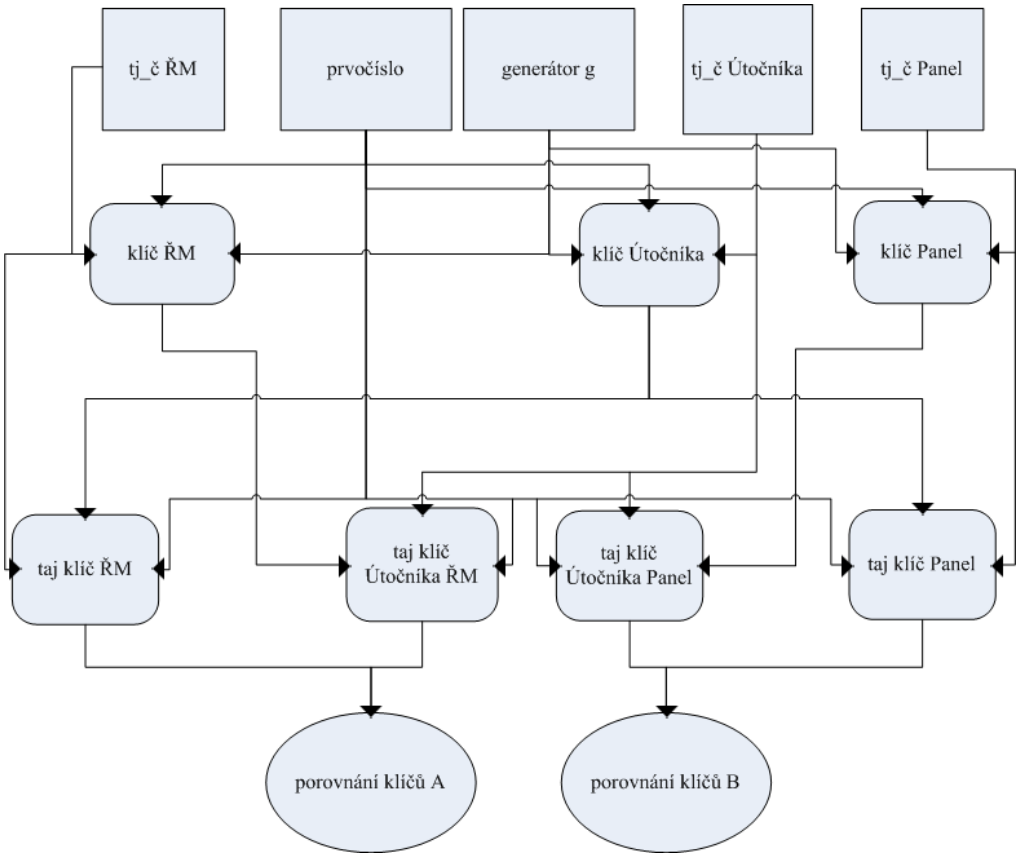


Obrázek 8: Porovnání tajných klíčů

Pokud došlo k nějaké chybě nebo ke změně čísel, porovnání dvou tajných klíčů se nebude shodovat. Vyhodnocení rozdílných tajných klíčů bude signalizovat textem „False“, a v tomto případě nebudou přijaty žádné instrukce pro zobrazovací panel od řídícího modulu.

4.2 Simulace útoku „Man in the middle“

Simulace navrženého útoku „Man in the middle“ a odzkoušení napadení protokolu Diffie-Hellman pro řídící modul a zobrazovací panel. Celkový blokový přehled navržené simulace je zobrazen na Obrázku 9.



Obrázek 9: Celkový přehled simulace útoku „Man in the middle“

I v této simulaci je zapotřebí generátor náhodných čísel označený *generátor g* sloužící pro vygenerování primitivního čísla. Princip generátoru čísla *g* je stejný jako v simulaci protokolu Diffie-Hellman pro řídící modul a dva zobrazovací panely. Generované číslo nemusí být nijak veliké, obvykle je toto číslo dvou až pěti ciferní. Při útoku na tento modul se útočník pokouší zachytit generované číslo *g* po nezabezpečeném komunikačním kanálu. Při úspěšném pokusu získání čísla *g*, schází útočníkovi získat generované prvočíslo.

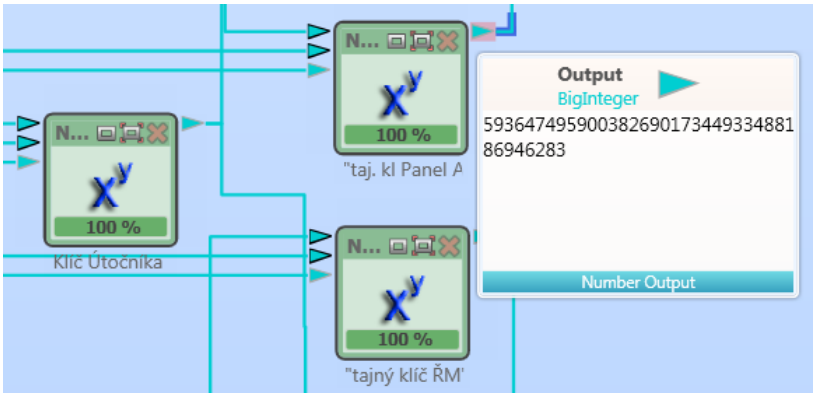
Generátor prvočísla má název v simulaci útoku *prvočíslo*. I toto velké prvočíslo je při úspěchu zachyceno útočníkem, který si již dokáže z těchto dvou zachycených čísel *g* a *p* vypočítat svůj vlastní klíč. Tento klíč je vypočítán v modulu nazvaný *klíč Útočníka*. Prvočíslo je využito ve všech výpočtech veřejných i tajných klíčů panelu, řídícího

modulu i útočníka.

V modulu *tj_č Útočníka* je zvoleno tajné číslo útočníka, které se využije pro výpočet veřejného klíče i ve výpočtu tajných klíčů pro komunikaci s řídícím modulem a zobrazovacím panelem. Útočník se nyní může vydávat jakoby za jednoho člena komunikace a se svým tajným klíčem může šifrovat i dešifrovat zprávy, jelikož je zcela nemožné zjistit tajný klíč řídícího modulu nebo zobrazovacího panelu.

Princip modulů pojmenovaných *tj_č ŘM* a *tj_č Panel* je totožný. V těchto modulech se zvolí tajná čísla jednotlivých modulů a jsou dále využita při výpočtu veřejného klíče a tajného klíče pro každý modul zvlášť. Řídící modul vypočítá svůj veřejný klíč v modulu nazvaném *klíč ŘM* a tento klíč odešle zobrazovacímu panelu. Stejný princip je i na straně zobrazovacího panelu, který svůj veřejný klíč vypočítá v modulu *klíč Panel* a odešle řídícímu modulu.

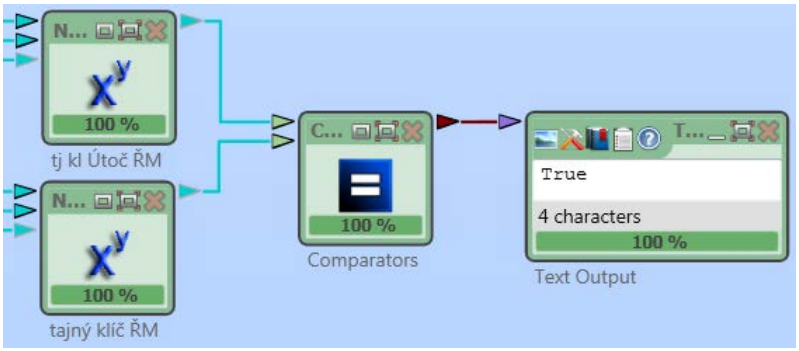
V tomto kroku nastává zlom, kdy svůj veřejný klíč odeslali útočníkovi, který to využije k vypočtení tajného klíče. Tajný klíč vypočítá pro každou stranu zvlášť. Tedy vypočte tajný klíč pro řídící modul i pro zobrazovací panel zvlášť. Pro řídící modul svůj tajný klíč vypočte v modulu *taj klíč Útočníka ŘM*. Pro zobrazovací panel vypočte tajný klíč v modulu pojmenovaném *taj klíč Útočníka Panel*. Po přijetí veřejných klíčů od řídícího modulu a zobrazovacího panelu, zašle útočník vlastní veřejný klíč těmto modulům. V modulu *taj klíč Panel* a *taj klíč ŘM* si nyní vypočítají své unikátní tajné klíče. Vypočítaný tajný klíč modulu *taj klíč Panel* v simulaci je zobrazen na Obrázku 10, spolu s celým blokem modulů útočníka. Nyní útočník získal veškerou kontrolu nad odeslanými instrukcemi a je schopen kontrolovat stav instrukcí a dokáže i bez problému změnit obsah odeslané instrukce, popřípadě odeslat svojí zcela novou instrukci.



Obrázek 10: Blok klíčů útočníka v simulaci se zobrazeným tajným klíčem Útočníka

Blok, který porovnává výsledky tajných klíčů útočníka nazvaný *taj klíč Útočníka Panel* a zobrazovacího panelu *taj klíč Panel*, se jmenuje *porovnání klíčů B*. Pokud se tajné klíče shodují, v porovnávacím modulu se zobrazí text „True“. Dojde-li k nějaké chybě, porovnávací modul zobrazí text „False“. V okamžiku shodných klíčů se zobrazovací panel domnívá, že komunikace probíhá mezi ním a řídícím modulem. Bez problémů bude přijímat veškeré instrukce. Na stejném principu bude pracovat porovnávací blok pojmenovaný *porovnání klíčů A*. Tento porovnávací blok porovnává tajné klíče z modulu útočníka *taj klíč Útočníka ŘM* a řídícího modulu pojmenovaný *taj klíč ŘM*. Opět dojde-li ke shodě tajných klíčů v porovnávacím bloku, řídící modul předpokládá, že odeslané instrukce směřují zobrazovacímu panelu. I v tomto případě má

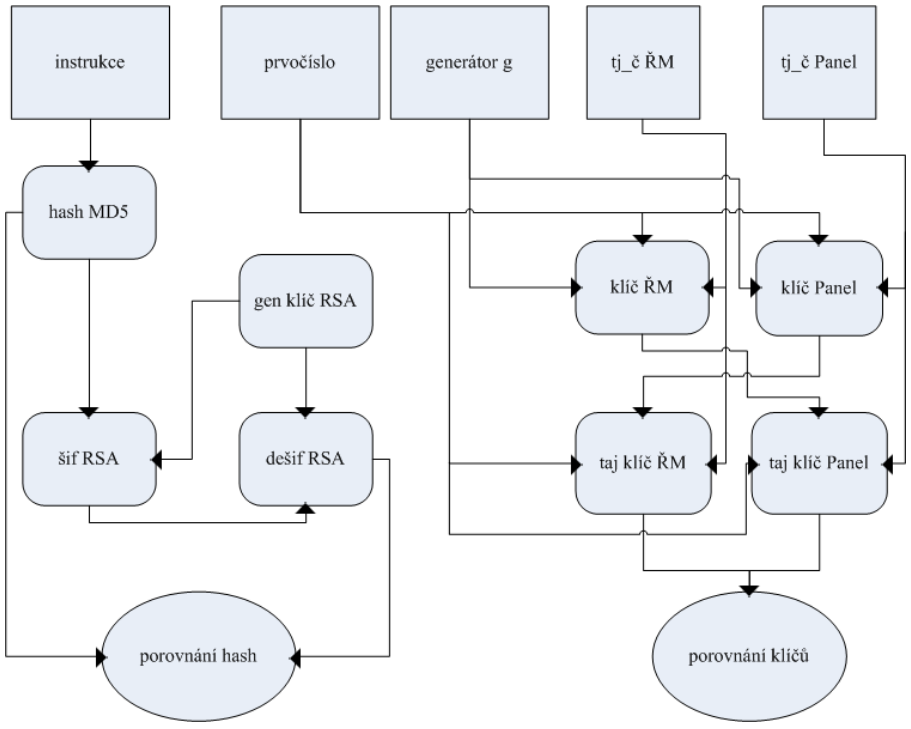
útočník veškerou kontrolu nad odeslanými instrukcemi. V navržené simulaci bylo dosaženo úspěchu napadení metodou „Man in the middle“ zobrazený na Obrázku 11, který zobrazuje porovnání tajných klíčů útočníka a řídícího modulu.



Obrázek 11: Blok porovnání klíčů A

4.3 Sestavení Digitálního podpisu

V této kapitole je popsáno sestavení a simulace navrženého protokolu Diffie-Hellman s jedním řídícím modulem a zobrazovacím panelem spolu s digitálním podpisem. Celkový blokový přehled navržené simulace je zobrazen na Obrázku 12.



Obrázek 12: Protokol Diffie-Hellman s digitálním podpisem

Digitální podpis odeslaných instrukcí řeší problém útoku „Man in the middle“ v protokolu Diffie–Hellman po nezabezpečeném komunikačním kanálu. V případě útoku metodou „Man in the middle“ zjistí útočník potřebné údaje k vytvoření vlastních klíčů a filtrace komunikace, ale již není schopen odeslanou instrukci pozměnit.

Digitálním podpisem instrukcí ztrácí možnost odeslat vlastní instrukci zobrazovacímu panelu. Odeslaná instrukce od útočníka nebude mít správný digitální podpis a zobrazovací panel takovou instrukci odmítne.

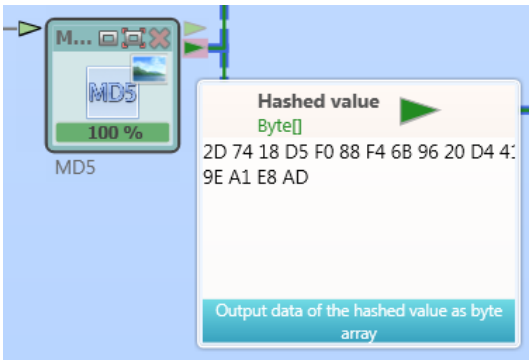
Simulace opět využívá již známý generátor náhodných čísel označený *generátor g,* který slouží pro vygenerování primitivního čísla. Generátor generuje primitivní číslo pro modul *klíč Panel* a *klíč ŘM*. Primitivní číslo o délce dvou až pěti cifer je zakombinováno v rovnici pro výpočet veřejného klíče zobrazovacího panelu a řídícího modulu.

Generátor prvočísla je i zde pojmenovaný *prvočíslo*. Je využito při výpočtu veřejného klíče pro zobrazovací panel nazvaný *klíč Panel*. Řídící modul toto generované číslo používá v modulu *klíč ŘM* pro vlastní výpočet veřejného klíče. Dále je využito k sestavení tajného klíče v zobrazovacím panelu *taj klíč Panel*. Nakonec je použito i pro výpočet tajného klíče v řídícím modulu, který je nazvaný *taj klíč ŘM*. Generátor prvočísla se nachází v simulaci vedle generátoru primitivního čísla *g*. Pozice je stále stejná i pro generátor primitivního čísla *g* pro lepší přehlednost v návrhu simulace.

Moduly pojmenované *taj_č Panel* a *tj_č ŘM,* slouží k vložení vlastního tajného čísla do rovnic výpočtů pro veřejné klíče i pro výpočet tajných klíčů. Řídící modul své tajné číslo posílá modulu jménem *klíč ŘM*, kde je vypočítán veřejný klíč. Tento veřejný klíč je dál šířen k výpočtu tajného klíče u zobrazovacího panelu. Zobrazovací panel *taj_č Panel* své tajné číslo odešle modulu nazvanému *klíč Panel*. V tomto modulu *klíč Panel* zobrazovací panel vypočte svůj veřejný klíč, který zašle řídícímu modulu. Veřejné klíče si navzájem vymění a slouží pro výpočet tajných klíčů. Řídící modul svůj tajný klíč vypočítá v modulu pojmenovaný *taj klíč ŘM*. Tajný klíč řídícího modulu je vypočítán ze všech hodnot zaslaných z modulů *prvočíslo, klíč Panel* a vlastní tajné číslo z modulu *tj_č ŘM*. Stejný princip je i u modulu výpočtu tajného klíče zobrazovacího panelu. Modul *taj klíč Panel* pro výpočet tajného klíče potřebuje moduly pojmenované *prvočíslo*, vyměněný klíč s řídícím modulem *klíč ŘM* a vlastní tajné číslo z modulu *tj_č Panel*. V obou případech proběhne výpočet tajných klíčů, které se porovnají v bloku nazvaném *porovnání klíčů*.

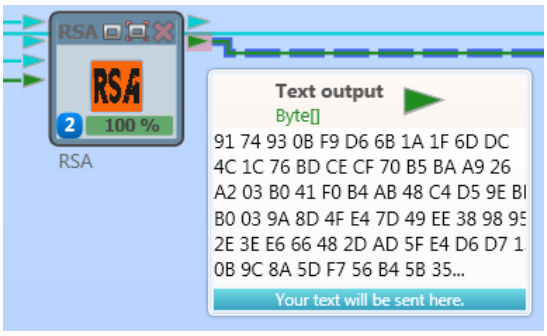
V bloku *porovnání klíčů* dojde k porovnání dvou tajných klíčů řídícího modulu a zobrazovacího modulu a při shodě tajných klíčů vypíše text „True“. Porovnávací blok zobrazující text „True“ signalizuje správné porovnání tajných klíčů řídícího modulu a zobrazovacího panelu. Nyní je možné mezi řídícím modulem a zobrazovacím panelem komunikovat a zasílat patřičné instrukce.

Pro případ napadením útočníkem metodou „Man in the middle“ je zde navržen princip digitálního podpisu, který zabrání jakkoliv pozměnit či zneužít vyslané informace po nezabezpečeném komunikačním kanálu. Sada příkazů nebo instrukcí je uložena v simulaci pod jménem *instrukce*. Požadované instrukce je potřeba nyní převést do matematické funkce nazývané hash. V navržené simulaci je pod názvem *hash MD5*. Jak je patrné z názvu, tento modul *hash MD5* využívá funkce algoritmu MD5 se 128 bitovou velikostí. Algoritmus hash je využíván především kvůli tomu, že dokáže převést jakékoliv množství znaků na fixní délku v hexadecimální soustavě a sebemenší změna ve zdrojovém souboru způsobí celkovou změnu ve výstupu dat. Na Obrázku 13 je znázorněn hash MD5 ze simulace programu, který zašifroval zaslané instrukce.



Obrázek 13: Hash MD5

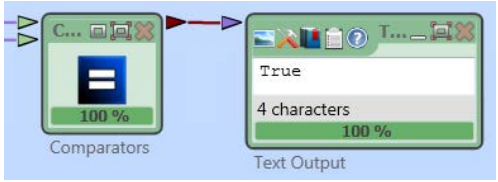
Nyní je výsledný hash šifrován asymetrickým algoritmem RSA. Šifrování výsledného hashe *hash MD5* algoritmem RSA je v návrhu označen textem *šif RSA*. K zašifrování pomocí algoritmu RSA je potřeba dvou klíčů. Těmi klíči v navržené simulaci jsou veřejný klíč *n* a soukromý klíč *d*. Tyto klíče jsou generované v generátoru RSA klíčů. Generátor RSA klíčů je v simulaci pojmenován *gen klíč RSA*. V generátoru *gen klíč RSA* se generuje ještě jeden veřejný klíč pro dešifrování algoritmu RSA, tímto veřejným klíčem je klíč *e*. Nyní jsou instrukce v podobě hashe zašifrované algoritmem RSA a jsou odeslány po nezabezpečeném komunikačním kanálu zobrazovacímu panelu. Zkrácený výpis zašifrovaných dat algoritmem RSA je zobrazen na Obrázku 14.



Obrázek 14: Zašifrované data algoritmem RSA

Zobrazovací panel přijme zašifrovaný hash instrukcí a nyní je dešifruje. Dešifrování probíhá v modulu nazvaném *dešif RSA*. K dešifrování instrukcí je zapotřebí dvou veřejných klíčů. Veřejný klíč *n* a veřejný klíč *e*. Pomocí těchto veřejných klíčů je možné dešifrovat přijatý hash vytvořený z instrukcí. Dešifrovaný hash se nyní porovná s originálním hashem, který se vytvoří z přenesených originálních instrukcí.

Porovnání číselných kombinací hash v hexadecimálním tvaru se provádí v bloku pojmenovaném *porovnání hash*, který je zobrazen na Obrázku 15. I v tomto případě je porovnání hashů v simulaci navrženého sestavení digitálního podpisu shodné.



Obrázek 15: Porovnání hash

V tomto porovnávacím bloku je srovnáván hash vytvořený z originálních instrukcí pojmenovaný v simulaci *hash MD5* a z dešifrovaného hashe, který byl digitálně podepsán, v simulaci nazvaný *dešif RSA*. Vyhodnocení jednotlivých kombinací hash sobě navzájem odpovídají. Zobrazením výsledků v bloku *porovnání hash* dojde k vyhodnocení signalizující textem „True“. V tomto případě se jedná o shodnou kombinaci hash, který touto shodou potvrzuje autentičnost odeslaných instrukcí zobrazovacímu panelu. Zobrazovací panel bez komplikací a s jistotou přijme instrukce a provede jejich účel. Došlo-li k nějaké změně čísel nebo ke změně instrukcí, porovnání dvou vytvořených hash se nebudou shodovat. V bloku porovnávajícím hashe se pro odlišné kombinace čísel zobrazí signalizace s textem „False“. Nyní nastane situace, kdy zobrazovací panel nepřijme žádné zaslané instrukce od řídícího modulu. Je pravděpodobné, že jsou instrukce poškozené, nebo se někdo pokusil o jejich změnu.

Takto odlišné hashe mohou signalizovat, že sada instrukcí pro zobrazovací panel od řídícího modulu byla napadena útočníkem. Zašifrované sady instrukcí digitálním podpisem mohou být napadeny útočníkem, který se pokouší o změnu sady instrukcí, pro vlastní užitek. Z důvodu zaměření této konkrétní práci se zobrazovacími informačními panely a řídícím terminálem, není příliš důležité šifrovat přímo sadu instrukcí, aby útočník nemohl přečíst konkrétní parametry instrukcí. Jelikož v tomto významu nejde o riziko zneužití informací ze sady instrukcí, které mají minimální užitek pro daného útočníka. Jde především o zabezpečení před zneužitím informačních panelů pro nevyžádaný text nebo informaci zobrazenou na tomto informačním panelu, který by vedl k následné mystifikaci případných uživatelů.

5 ZÁVĚR

Bakalářská práce řeší problematiku zabezpečení komunikace informačních panelů s řídící jednotkou. Základní princip zabezpečení komunikace je rozebrán ve třetí a čtvrté kapitole bakalářské práce.

Jedním z úkolů zadání bylo prostudovat metody komunikace informačních panelů s řídící jednotkou a zaměřit se především na komunikační protokoly a metody zabezpečení přenosu zobrazovaných informací. Metody komunikace probíhají po komunikačním rozhraní zajišťujícím správnou komunikaci, obsluhu specifických zařízení a přenos dat po sběrnici mezi řídící jednotkou a zobrazovacím panelem. Zadání tohoto úkolu je rozebráno v úvodní kapitole jedna. Komunikačními protokoly se zabývá kapitola dva, kde jsou podrobně popsány tři komunikační protokoly, které jsou nejvhodnější pro zadanou problematiku. Komunikační protokol je chápán jako standard pro digitální komunikaci a přenos digitálních dat mezi body řídícího terminálu a zobrazovacího panelu.

Další bod zadání obsahuje návrh koncepce modulu pro zabezpečenou komunikaci mezi zobrazovacím panelem a terminálem. Tímto bodem zadání se zabývá kapitola tři, ve které je popsána nejefektivnější metoda komunikace pro účel této práce. Touto metodou je protokol Diffie-Hellman. V kapitole jsou rozebrány postupy sestavení jednoduché komunikace mezi řídící jednotkou a zobrazovacími informačními panely. Je provedena názorná ukázka výpočtů tajných klíčů mezi komunikujícími stranami. Nachází se zde i princip útoku metodou „Man in the middle“. Zmíněná metoda útoku je největší slabinou protokolu Diffie-Hellman, jelikož protokol neumožňuje autentizaci komunikujících stran. Pro tuto slabinu protokolu je nejvhodnější způsob ochrany v digitálním podpisu, který zajistí autentizaci komunikujících stran. Digitální podpis je popsán v kapitole tři a následně je provedeno názorné sestavení daného digitální podpisu i s certifikační autoritou. Posledním bodem v této kapitole je ukázka ověření digitálního podpisu.

Poslední čtvrtá kapitola je věnována simulaci teoretických informací ohledně zabezpečení komunikace mezi řídící jednotkou a informačním panelem. Simulace protokolu Diffie-Hellman s jednou řídící jednotkou a dvěma informačními panely. Jsou zde vysvětleny jednotlivé moduly simulace a k čemu jsou jednotlivé moduly určeny. Dalším bodem je navržená simulace útoku metodou „Man in the middle“ v protokolu Diffie-Hellman s jednou řídící jednotkou a jedním informačním panelem. I v této simulaci jsou podrobně rozebrány jednotlivé moduly simulace. Poslední část je navržená simulace protokolu Diffie-Hellman s digitálním podpisem pro zabezpečení digitálních dat pro komunikaci po nezabezpečeném komunikačním kanálu. V simulaci je provedena výměna tajných klíčů protokolu Diffie-Hellman a jejich následné ověření s názorným vytvořením digitálního podpisu se zašifrovanými instrukcemi pro informační panel a jeho následné ověření s originálními daty.

Pro zabezpečení dat od řídící jednotky přes nezabezpečený komunikační kánal pro zobrazovací informační panel lze pro jednoduchost a velkou efektivnost využít protokolu Diffie-Hellman s digitálním podpisem. Zaručí spolehlivou integritu instrukcí s minimálním úsilím a náklady na realizaci.

LITERATURA

[1] PAZUL, Keith. *Controller Area Network (CAN) Basics* [online]. Microchip Technology Inc., 2002. Dostupné z WWW: http://www.cl.cam.ac.uk/research/srg/HAN/Lambda/webdocs/an713.pdf.

[2] KOŘÍNEK, František. *Analyzátor sběrnice CAN pro nákladní automobily*. Praha, 2010. 68 s. Diplomová práce. České vysoké učení technické v Praze.

[3] SHANKAR, A. UDAYA ; LAM, SIMON S. *An HDLC Protocol Specification and Its Verification Using Image Protocols* [online]. University of Texas at Austin : ACM Transactions on Computer Systems, 1983. Dostupné z WWW: http://www.cs.utexas.edu/users/lam/Vita/ACM/ShankarLam83.pdf.

[4] High-Level Data Link Control. In: *Wikipedia*: *the free encyclopedia* [online]. [cit. 2012-05-06]. Dostupné z WWW: http://en.wikipedia.org/wiki/HDLC

[5] BERRY, Gérard Berry; GONTHIER, Georges. *Incremental Development of an HDLC Protocol in ESTEREL*. Valbonne - France : Sophia-Antipolis, 1988. 50 s.

[6] ARM, JAKUB. *ŘÍZENÍ MODELU JEŘÁBU KOMUNIKUJÍCÍHO PROTOKOLEM MODBUS TCP*. Brno, 2011. 55 s. Bakalářská práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ.

[7] *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b* [online]. Modbus-IDA, 2006. Dostupné z WWW: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf.

[8] *MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b* [online]. Modbus-IDA, 2006. Dostupné z WWW: http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

[9] STALLINGS, William. *Cryptography and Network Security Principles and Practices*. Englewood Cliffs : Prentice Hall, 2005. 592 s.

[10] SCHNEIER, Bruce. Applied cryptography. 2nd edition. [s.l.] : John Wiley & Sons, 1996. 784 s. ISBN 0-471-11709-9.

[11] DIFFIE, Whitfield a Martin HELLMAN. New Directions in Cryptography. *New Directions in Cryptography* [online]. 1976, č. 6 [cit. 2012-04-18]. Dostupné z WWW: http://www.cs.jhu.edu/~rubin/courses/sp03/papers/diffie.hellman.pdf

[12] SKLENÁK, Vilém et al. *Data, informace, znalosti a Internet*. Praha: C H Beck, 2001, 507 s.

[13] SUKOVÁ, Lenka. *Elektronický podpis a První certifikační autorita*. Praha, 2009. Dostupné z WWW: http://isis.vse.cz/lide/clovek.pl?id=312;zalozka=13;lang=cz. Bakalářská práce. Katedra systémové analýzy.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

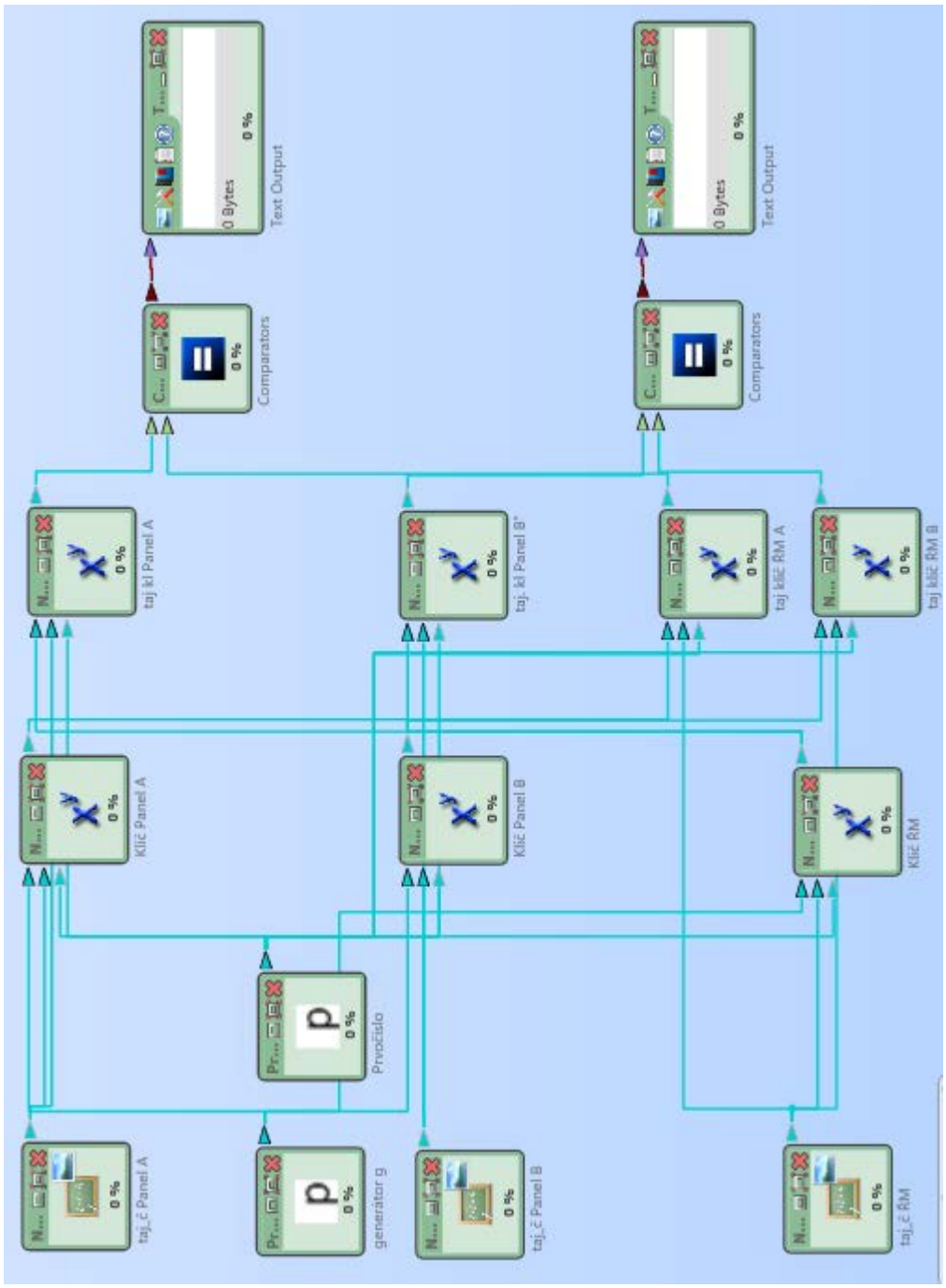
ABM	Asynchronous Balanced Mode
ACK	Acknowledge
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
ARM	Asynchronous Response Mode
CAN	Controller Area Network
CAST	Algoritmus pro šifrování, iniciály autorů Carlisle Adams a Stafford Taverns
CR	Carriage return
CRC	Cyclic Redundant Check
DES	Data Encryption Standard
DISC	Disconnec
DLC	Data Length Code
DM	Disconnected Mode
EOF	End Of Frame
FCS	Frame Check Sequence
FRMR	Frame Reject
FC	Function code
HDLC	High-Level Data Link Control
IDE	IDentifier Extension
IDEA	International Data Encryption Algorithm
IFS	Intermission Frame Space
ISO	International Organization for Standardization
IP	Internet Protocol
LF	Line Feed
NRM	Normal Response Mode
NRZ	Non Return to Zero
RC2	Rivest Cipher 2
RC4	Rivest Cipher 4
RC5	Rivest Cipher 5

REJ	Reject
RD	Request Disconnect
RIM	Request Initialization Mode
RNR	Receive Not Ready
RR	Receive Ready
RS485	Standard definující třídu asynchronních sériových linek
RSA	Algoritmus pro šifrování, iniciály autorů Rivest, Shamir, Aleman
RSET	Reset
RTR	Remote Transmission Request
RTU	Remote Terminal Unit
SABM	Set Asynchronous Balanced Mode
SARM	Set Asynchronous Response Mode
SDLC	Synchronous Data Link Control
SIM	Set Initialization Mode
SNRM	Set Normal Response Mode
SNRME	Set Normal Response Mode Extended
SOF	Start Of Frame
SREJ	Selective Reject
TCP	Transmition Control Protocol
Triple DES	Triple Data Encryption Standard
UA	Unnumbered Acknowledgment
UI	Unnumbered Information
UP	Unnumbered Poll
X.25	Typ protokolu založen na technologii přepojování paketů
XID	Exchange Identification

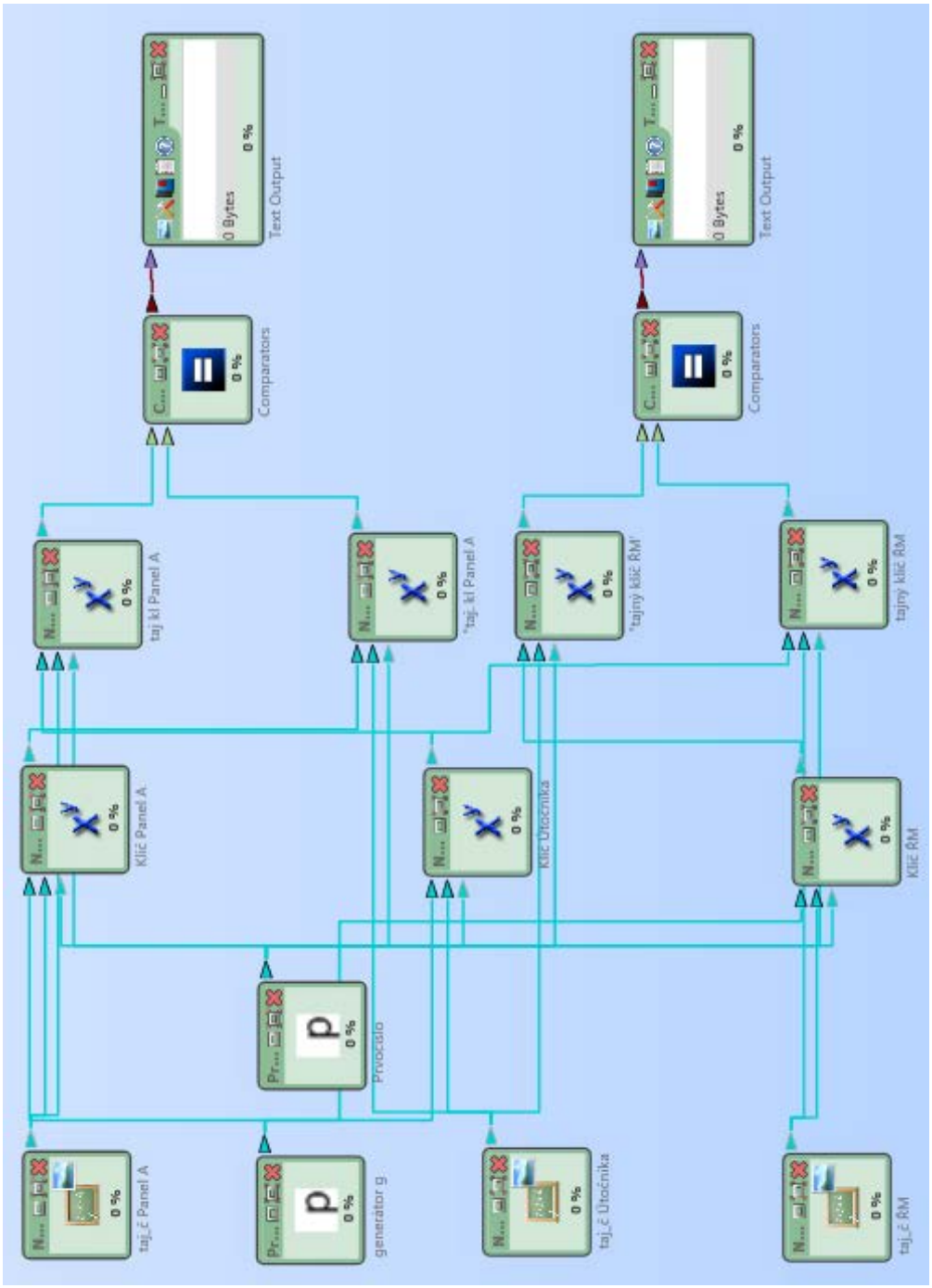
SEZNAM PŘÍLOH

A SCHÉMA ZAPOJENÍ V SIMULACI

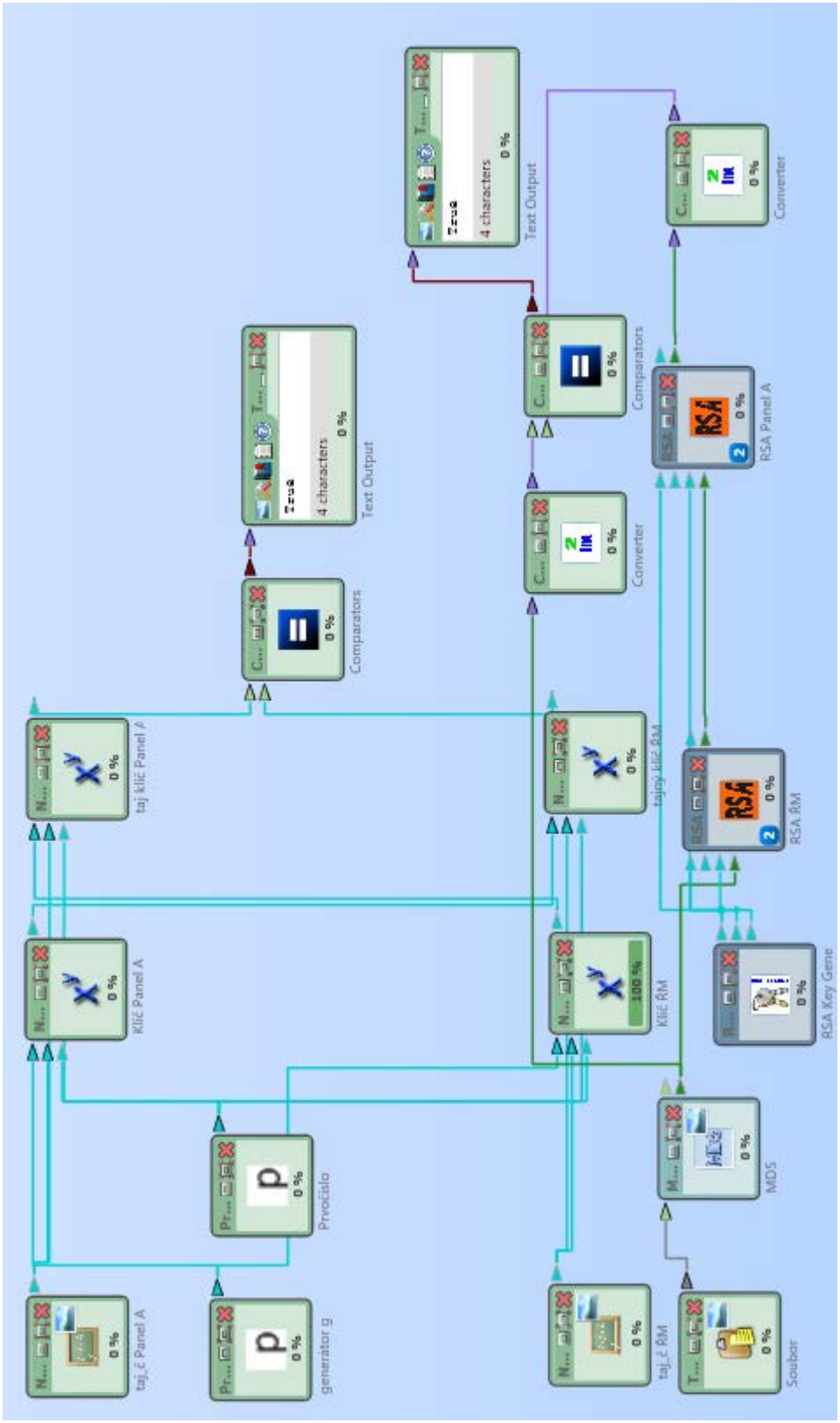
A.1 Protokol Diffie-Hellman



A.2 Útok „Man in the middle“



A.3 Sestavení Digitálního podpisu



B PŘILOŽENÉ CD SE ZDROJOVÝMI KÓDY

Kompaktní disk obsahuje tři zdrojové kódy navržené simulace. Zdrojový kód s protokolem Diffie-Hellman pod názvem souboru „DH-vymena_klicu“. Druhý soubor se jménem „Utok-Man_in_the_middle“ se simulací útoku „Man in the middle“. Poslední soubor se zdrojovým kódem simulace se jmenuje „digitalni_podpis“, který obsahuje navrženou simulaci digitálního podpisu. Kompaktní disk dále obsahuje elektronickou verzi bakalářské práce ve formátu pdf.