VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

SEMESTRÁLNÍ PRÁCE

Brno, 2020 Jakub Jedlička

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ŠIFRÁTOR PRO HARDWAROVĚ OMEZENÉ ZAŘÍZENÍ

SEMESTRÁLNÍ PRÁCE
SEMESTRAL THESIS

AUTOR PRÁCE — **Jakub Jedlička**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. David Smékal**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Semestrální práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Jakub Jedlička ***ID:*** 198597
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Šifrátor pro hardwarově omezené zařízení

POKYNY PRO VYPRACOVÁNÍ:

V rámci práce se student seznámí s hardwarovou akcelerací založené na platformě FPGA. Seznamte se s programovacím jazykem VHDL, FPGA obvody řady Zynq-7000 a platformou Xilinx Vivado. Nastudujte šifrovací algoritmy určené pro lehkou kryptografii. Následně šifru popište jazykem VHDL, odsimulujte a implementujte na hardwarovém zařízení FPGA s omezenými hardwarovými zdroji.
Výstupem semestrální práce je návrh šifrování v prostředí Vivado a funkční simulace ověřující správnou funkčnost návrhu.
V navazující bakalářské práci věnujte pozornost konkrétnímu FPGA obvodu s čipem Artix s omezenými hardwarovými zdroji a zprovozněte šifrátor na reálném zařízení. Úkolem bude také zprovoznění jednotlivých periferií komunikující s FPGA obvodem Artix (komunikace pomocí rozhraní USB 2.0, Micro SD, Ethernet, a jiné).

DOPORUČENÁ LITERATURA:

[1] BURDA, Karel. Aplikovaná kryptografie. Brno: VUTIUM, 2013. ISBN 978-80-214-4612-0.

[2] PINKER, Jiří, Martin POUPA. Číslicové systémy a jazyk VHDL. 1. vyd. Praha: BEN - technická literatura, 2006, 349 s. ISBN 80-7300-198-5.

Termín zadání: 2.10.2020 ***Termín odevzdání:*** 11.12.2020

Vedoucí práce: Ing. David Smékal

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

PROHLÁŠENÍ

Prohlašuji, že svou semestrální práci na téma „Hardwarový šifrátor pro lehkou kryptografii“ jsem vypracoval samostatně pod vedením vedoucího semestrální 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 semestrální práce dále prohlašuji, že v souvislosti s vytvořením této semestrální práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato semestrální práce se věnuje šifrám lehké kryptografie a následně výběru a implementaci zvolené šifry na obvod FPGA. Práce se skládá z pěti hlavních kapitol. První tři kapitoly obsahují teoretickou část a v posledních dvou kapitolách je popsán výběr a implementace šifry.

V první kapitole jsou blíže přiblíženy obvody FPGA. První část této kapitoly je zaměřena na popis jednotlivých částí a jejich rozložení na FPGA čipu. V druhé části je detailněji představen a popsán FPGA čip z rodiny Zynq-7000 od nejznámějšího výrobce firmy Xilinx, Inc.

V druhé kapitole jsou objasněny základy kryptografie, v jejíž první části jsou definovány základní pojmy kryptografie a v následující části jsou popsány hašovací funkce, asymetrické a symetrické kryptosystémy. Větší pozornost je věnována symetrickým kryptosystémům.

V kapitole třetí je detailněji rozebrán princip lehké kryptografie. Zde jsou obecně analyzovány kryptosystémy lehké kryptografie a jejich známí zástupci, ze kterých jsou podrobněji popsáni dva zástupci blokových šifer.

Ve čtvrté kapitole je popsán postup výběru algoritmu, který bude následně implementován na FPGA čip. V první části jsou rozebrány výhody a nevýhody pro implementaci daných typů kryptosystémů pro šifrovaní zdrojových dat. V další části jsou porovnány konkrétní algoritmy pro implementaci na FPGA čipy.

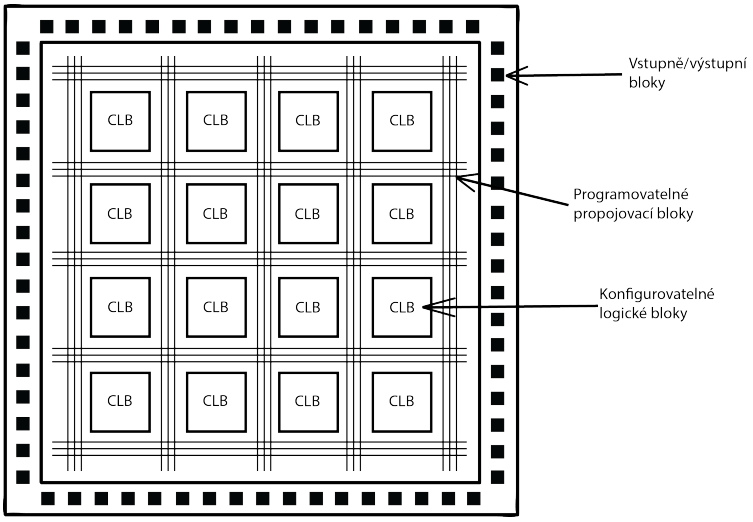
V poslední páté kapitole je popsána praktická implementace vybraného algoritmu. Algoritmus bude implementován na vývojářskou desku ZYBO Z7-20 a tato implementace bude detailněji popsána. Uvedený algoritmus bude po implementaci otestován na vstupních testovacích datech a šifrovaná data budou porovnána s ověřenými šifrovanými daty.

1 Programovatelné hradlové pole - FPGA

1.1 FPGA

Programovatelná hradlová pole neboli FPGA byli prvně představeny v 80. letech společností Xilinx, Inc. Do nedávna byly FPGA používány k prototypovaní a emulacím při procesu návrhu a výroby integrovaných obvodů určených pro aplikaci tzv. ASIC.[1] Nicméně v poslední době se FPGA čipy rozšířily mimo návrh ASIC obvodů a jsou vyráběny v mnoha variantách pro různá odvětví. Hlavní výhodou obvodů FPGA je možnost následného přeprogramovaní i po jejich výrobě.[2]

FPGA se skládají ze tří hlavních typů programovatelných stavebních bloků. Rozložení těchto tří bloků je zobrazeno na obrázku 1.1.



Obr. 1.1: Architektura FPGA čipu [3]

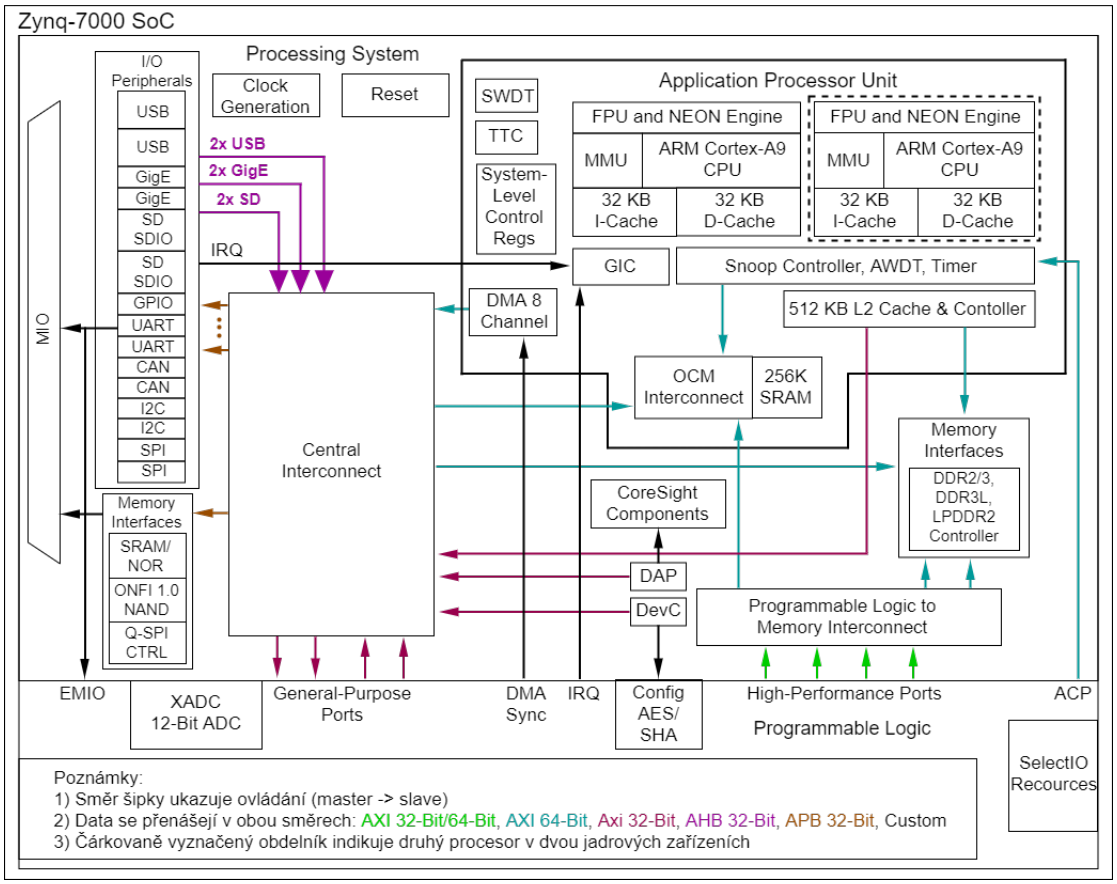
- IOB - vstupně/výstupní bloky
- PIB - programovatelné propojovací bloky
- CLB - konfigurovatelné logické bloky

CLB mohou být naprogramovány tak, aby implementovaly různé logické funkce. Při propojení pomocí PIB vytvářejí velké logické okruhy. Vstupně výstupní bloky IOB propojují interní a externí zdroje. Na FPGA čipu se dále nachází SRAM[1], která uchovává konfiguraci CLB, PIB a IOB bloků.[1]

[1]Statická paměť.

1.2 Xilinx Zynq-7000

Čipy z rodiny Zynq-7000 nabízí flexibilitu a škálovatelnost FPGA čipů, zatímco poskytují sílu, výkon a použitelnost, která je často spojována s ASIC čipy. O výpočetní sílů se stará processing system (PS), který je založen na technologii ARM©[2] Cortex™-A9, který obsahuje paměť přímo na čipu, rozhraní pro externí paměť a sadu periferních rozhraní. FPGA se často označuje jako programmable logic (PL), kde pro Zynq-7000 je založen na bázi 28nm výrobního procesu od firmy Xilinx. Výhodou rodiny Zynq-7000 je, že processing system a programmable logic jsou integrovány na jednom čipu. Touto integrací lze dosáhnout většího výkonu než když jsou tyto dvě části rozděleny na více čipech. Vývoj pro všechny typy programmable logic od Xilinxu probíhá v prostředí Vivado Design Suite.[4]



Obr. 1.2: Architektura Zynq-7000 čipu [4]

[2]Advanced RISC machine je typ procesoru, který má zmenšenou sadu instrukcí a není energeticky náročný.

2 Kryptografie

2.1 Základní pojmy

Kryptologie je věda zabývající se konstrukcí a překonáváním matematických metod zajišťující důvěrnost a autentičnost zpráv. Tato věda v sobě obsahuje kryptografii a kryptoanalýzu.
Kryptografie je věda zabývající se konstrukcí matematických metod pro zajištění důvěrnosti a autentičnosti zpráv.
Kryptoanalýza je věda zabývající se překonáváním matematických metod, které zajišťují důvěrnost a autentičnost zpráv.
Kryptografický systém zkráceně kryptosystém je systém algoritmů, který zajišťuje důvěrnost a autentičnost zpráv.[6]
Klíč je parametr procesu šifrovaní a dešifrovaní.[5]
Zpráva jsou vstupní data, která jsou potřeba šifrovat, někdy označována jako čistý nebo prostý text.[7]
Šifrovaný text jsou výstupní data z procesu šifrovaní, která jsou v nečitelné podobě.
Šifra je určitý postup neboli algoritmus pomocí kterého se šifruje a dešifruje zpráva.[5]
Šifrování je proces transformace určité zprávy do nečitelné podoby.[7]
Šifrátor je zařízení, které šifruje zprávu pomocí šifry na šifrovaný text.
Dešifrování se nazývá opačný proces šifrovaní neboli přetváříme šifrovaný text na původní zprávu.[7]

2.2 Princip kryptosystémů

V roce 1883 Auguste Kerchoffs sepsal podmínky pro vytváření kryptosystémů, kde jedna z podmínek se rozšířila a stala se uznávanou konvencí známou jako Kerchoffsův princip. Který říká:

> *„Znalost algoritmu a velikosti klíče stejně jako dostupnost známého prostého textu jsou standardní předpoklad v moderní kryptoanalýze. Protože útočník může tyto informace případně získat, není proto dobré se spoléhat na jeho utajení při posuzovaní jeho kryptografické síly.*[1] “

[1] Přeloženo z anglického jazyka, originální znění v Modern Cryptography na straně 208 [7]

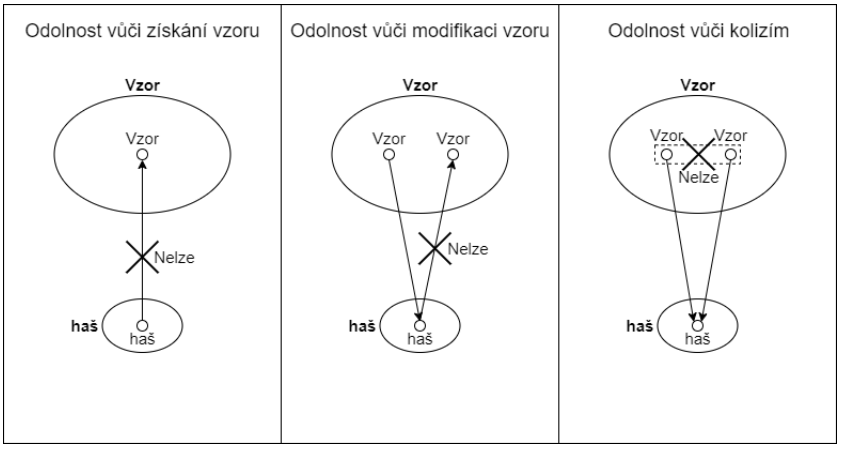
Propojením Kerchoffsova principu a Shannonovy teorie informace lze poskytnout shrnutí pro dobře navržený kryptografický systém:

- Algoritmus by neměl obsahovat žádnou část, která je tajná.
- Rozložení informace v zašifrovaném textu by mělo být po celé délce rovnoměrné až náhodné.
- Správný klíč by měl zaručit, aby šifrování a dešifrování bylo efektivní.
- Velikost klíče by měla udávat náročnost na dešifrování zašifrovaného textu bez použití správného klíče.[7]

2.3 Hašovací funkce

Jednosměrné funkce jsou takové, pro které lze vypočítat obraz funkce, ale z obrazu by nemělo být možné vypočítat vzor obrazu. Jednosměrné funkce se dělí na funkce s volitelnou a pevnou délkou výstupu.[6] Skupina s pevnou délkou výstupu se nazývá hašovací funkce. Hašovací funkce pracuje na principu, kde vstup je bitový řetězec proměnné délky a výstupem je řetězec o konstantní bitové délce takzvaný haš[2].[7]
Od hašovacích funkcí jsou zpravidla požadovány tyto vlastnosti:

- Odolnost vůči získaní vzoru
- Odolnost vůči modifikaci vzoru
- Odolnost vůči kolizím
- Efektivita



Obr. 2.1: Požadované Vlastnosti haš funkce [6].

Jednosměrnost neboli odolnost vůči získaní vzoru by měla splňovat, že z množiny hašů by nemělo být prakticky možné nalézt vzor, pro který by platila funkce H(vzor) = haš, kde H je hašovací funkce.

[2]Často označován anglicky hash

Odolností vůči modifikaci vzoru se požaduje, aby prakticky bylo téměř nemožné po vypočtení haše z prvního vzoru nalézt druhý vzor, který by měl stejný haš.

Při odolnosti vůči kolizím by mělo být téměř nemožné najít dva rozdílné vzory pro které by vycházel stejný haš.[6]

Pro splnění efektivnosti by daná hašovací funkce měla být realizovatelná v polynomiálním čase, nejlépe však v lineárním.[7]

Nejčastější využití hašovacích funkcí nalezneme v digitálních podpisech, kryptografických aplikacích generujících pseudonáhodné data a asymetrických kryptosystémech. V digitálních podpisech je hašovací funkce použita k vytvoření otisku zprávy, který slouží k ověření autentičnosti a integritě. Generace pseudonáhodných dat se využívá ve spoustě případů od autentifikačních protokolů po protokoly elektronické komerce. V asymetrických kryptosystémech se používá k ověření správnosti šifrovaného textu. Tento mechanizmus je nezbytný k zabezpečení ochrany proti útokům.[7]

2.4 Asymetrická kryptografie

Asymetrická kryptografie někdy nazývaná kryptografie s veřejným klíčem používá dva rozdílné klíče. Tyto klíče se nazývají veřejný klíč označovaný jako „public key“ a soukromý klíč označovaný jako „private key“. Veřejný klíč se používá k šifrovaní zprávy a soukromý klíč k dešifrovaní zašifrované zprávy pomocí veřejného klíče. Hlavní silou této metody je, že příjemce vypočítá veřejný a soukromý klíč, ale sdílí pouze veřejný. Odesílatel pomocí asymetrické šifry a veřejného klíče šifruje zprávu, kterou následně pošle příjemci, který vydal veřejný klíč. Příjemce pomocí soukromého klíče dešifruje přijatou zprávu. Výhoda této metody spočívá v tom, že při odeslání špatnému příjemci nebo odcizení zprávy by neměl nesprávný příjemce dešifrovat zprávu bez znalosti soukromého klíče. Tyto dva klíče jsou propojené složitými matematickými výpočty, kdy veřejný klíč nenese informace o soukromém klíči. Z tohoto důvodu při požadavcích kladených na bezpečnost v dnešní době v reálném čase není možné vypočítat z veřejného klíče soukromý klíč. Nejznámější a nejpoužívanější šifrou je RSA (Rivest–Shamir–Adleman).[8]

2.5 Symetrická kryptografie

Symetrická kryptografie se odlišuje od asymetrické kryptografie počtem klíčů, kdy symetrická kryptografie využívá pouze jednoho soukromého klíče, který slouží pro šifrovaní a dešifrovaní. Tento klíč si obě dvě strany, které chtějí komunikovat, musejí vyměnit. Problém může nastat při ustanovení a výměně klíče, kde lze klíč jednoduše odchytit pokud spojení není zabezpečené. Bezpečná výměna a ustanovení klíče

Obr. 2.2: Princip asymetrického kryptosystému [7].

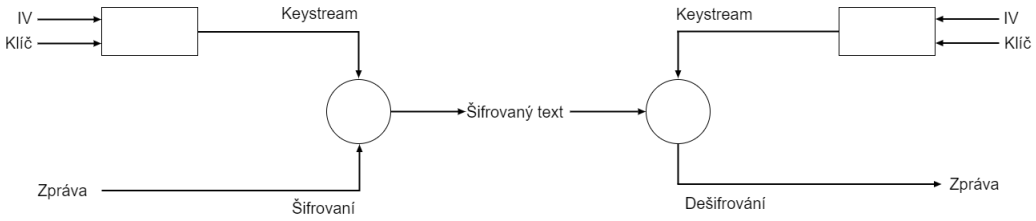
může proběhnout fyzickou výměnou, která v moderní době není možná v globálním měřítku. Při ustanovení a výměně klíče se přes nezabezpečenou síť využívá asymetrické kryptografie. V některých případech symetrické algoritmy využívají dva klíče - jeden k šifrovaní a druhý k dešifrovaní. Při odcizení šifrovacího klíče lze jednoduše vypočítat dešifrovací klíč a naopak. Jelikož se při vytváření šifry počítá, že algoritmus na ustanovení klíčů je veřejně známý, měl by být rozsah klíčů velmi velký. Pokud je rozsah klíčů malý, lze jej prolomit hrubou silou. Symetrické šifry dělíme na proudové a blokové.[8]

2.5.1 Proudové šifry

Mezi hlavní výhody proudových šifer patří bezchybovost nebo velice malá chybovost. Tyto šifry musí zpracovávat proud bitů a každý bit se šifruje samostatně. Oproti blokovým šifrám jsou rychlejší a méně hardwarově náročné. Lze je rozdělit na dva typy. Prvním typem jsou synchronní a druhým jsou asynchronní. Rozlišujeme je podle generace takzvaného keystreamu, což je generující klíčový proud znaků.[9]

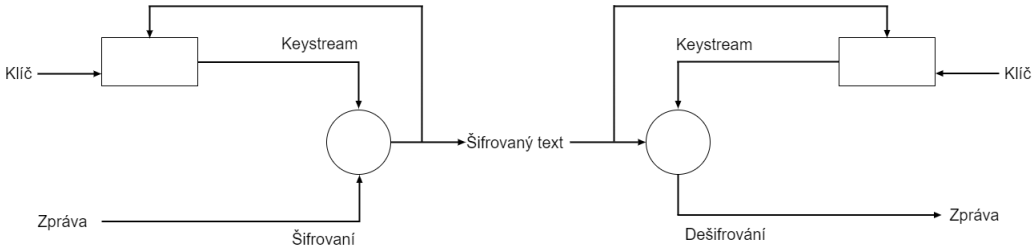
U synchronních proudových šifer nezáleží na šifrovaném a nešifrovaném textu při generaci keystreamu. Generace keystreamu je pouze závislá na klíči a šifrovacím algoritmu, ke kterému se u moderních šifer přidává ještě inicializační vektor (IV).[10] Jedním z požadavků na takovou šifru je synchronizace odesílatele a příjemce, to znamená, že každá strana musí mít stejný stav algoritmu a sdílený klíč. Výhodou je, že dojde-li při přenosu šifrované zprávy k záměně znaku, zbytek zprávy nebude ovlivněn. Z čehož plyne, že synchronní proudové šifry jsou vhodné pro použití tehdy, když na přenosové lince dochází k velké chybovosti. Nevýhodou těchto šifer je, že nelze zprávu dešifrovat při ztrátě synchronizace odesílatele a příjemce. Při ztrátě synchronizace se pro dešifrovaní musí použít speciální metody. [9]

U asynchronních proudových šifer oproti synchronním šifrám je keystream generován šifrovacím algoritmem za pomoci klíče a předchozích šifrovaných znaků zprávy. Stejně jako u synchronních šifer by měl být příjemce a odesílatel synchronizován.



Obr. 2.3: Princip šifrovaní a dešifrování synchronních proudových šifer.[10]

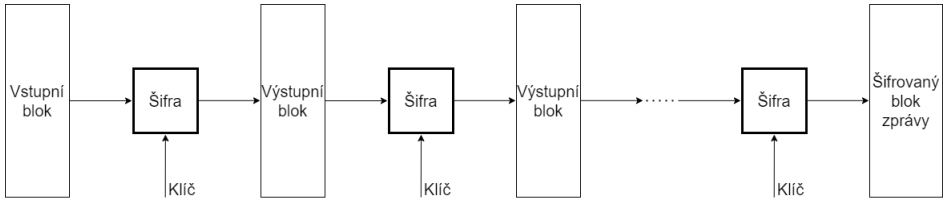
Dojde-li však k desynchronizaci, měla by se šifra sama po několika šifrovaných znacích synchronizovat. Pokud při přenosu nastane chyba, při dešifrovaní několik následujících znaků bude dešifrování provedeno chybně, ale zbytek zprávy bude dešifrován v pořádku, jestliže nenastanou další chyby.[9]



Obr. 2.4: Princip šifrovaní a dešifrování synchronních proudových šifer.[11]

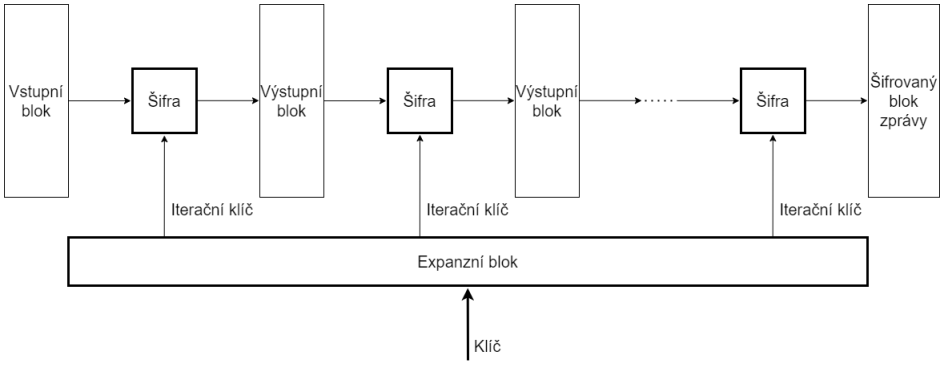
2.5.2 Blokové šifry

Blokové šifry na rozdíl od proudových provádí šifrovaní po blocích, kde každý blok je stejně dlouhá n-tice bitů. Moderní blokové šifry jsou založeny na principu kaskádových šifer. Kaskádové šifry fungují na základě řetězení několika šifer. Do následující šifry je vstupním blokem výstupní blok z předcházející šifry. Poslední výstupní blok je výsledný šifrovaný blok zprávy. V blokových šifrách se používá princip iterované šifry, která se dělí na Fiestelovu síť a substitučně-permutační síť.[6]



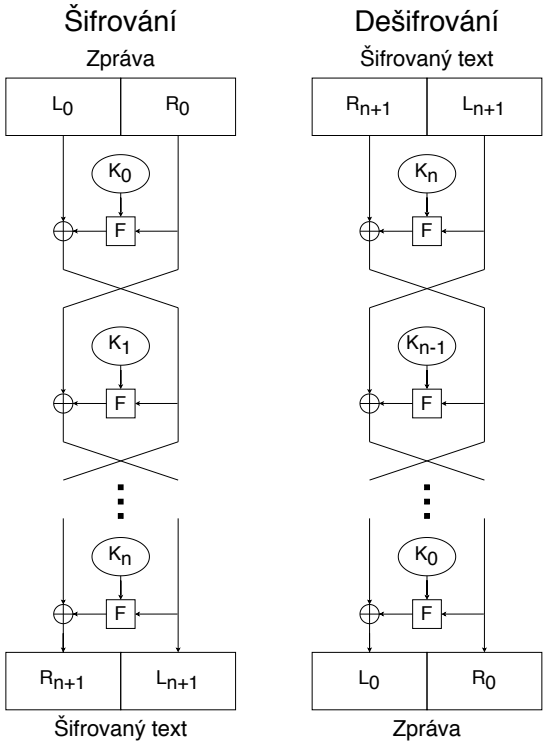
Obr. 2.5: Princip kaskádové šifry.[6]

Iterované šifry využívají princip kaskádových šifer, kdy se používá zřetězení stejné šifry po několik iterací. Jako šifra pro iteraci je nejčastěji využívána některá z jednoduchých šifer. Z klíče se odvodí několik iteračních klíčů v expanzním bloku. Tyto klíče jsou následně použity postupně pro každou iteraci. Jednoduchá šifra společně s klíčem pomocí vhodných blokových operací jako jsou permutace, substituce a aritmetické operace, šifrují data do výstupního bloku, který je následně použit jako vstup do další iterace.[6]



Obr. 2.6: Princip šifrovaní iteračních kaskádových šifer.[6]

Prvním typem iterované šifry je Fiestelova síť. Tento typ šifry funguje na principu rozdělení vstupního bloku na levou a pravou polovinu. Pravá polovina a iterační klíč vstupují do konverzní funkce, která má výstup o délce poloviny vstupního bloku. Výsledkem iterace je blok o velikosti vstupního bloku, kde novou levou stranou je předcházející pravá strana a pravou stranou je předcházející levá strana. Jak je patrné, tak šifrována byla pouze levá strana, z čehož vyplývá, že k zašifrovaní vstupního bloku jsou potřeba nejméně dvě iterace. Z tohoto důvodu bývá počet iterací sudý a dostatečně velký do té míry, aby se zabezpečila difúze šifry.[6]



Obr. 2.7: Princip šifrovaní a dešifrování Fiestelovy sítě.[12]

Druhým typem jsou substitučně-permutační síťě zkráceně SPN. Tento typ šifer je založen na substituci a následné permutaci vstupních bitů. Při substituci se využívá takzvaných S-boxů, které pracují s částí zprávy, nejčastěji o velikosti 4 a 8 bitů, kdy se posloupnost těchto bitů nahradí za jinou posloupnost. Permutace se realizuje pomocí takzvaných P-boxů. Permutace se snaží dosáhnout co největší difúze, aby bylo ovlivněno maximální množství S-boxů.[13]

Nejčastějšími operacemi používané v substitučně-permutačních síťích jsou:

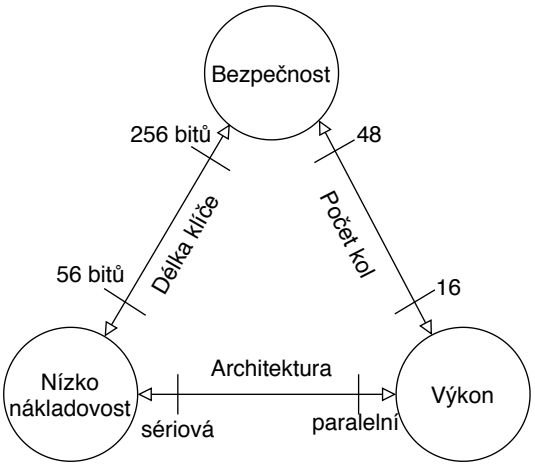
- SubBytes – nelineární substituce bitů, které operují nezávisle na sobě a jsou zaměňovány dle předem dané tabulky substituce, která je zároveň klíčem.
- ShiftRow – v této operaci se bity posouvají v řádku matice do stran a každý řádek se posouvá o jiný počet míst.
- MixColumns – spočívá v záměně sloupců, následně je každý sloupec vynásoben polynomem o stejné hodnotě.
- AddRoundKey – každý bit je zkombinován s iteračním klíčem, kdy po kombinaci bitů zprávy a iteračního klíče dostaneme výsledný šifrovaný text.[9]

3 Lehká kryptografie

3.1 Úvod

Lehká kryptografie je relativně mladé odvětví, ve kterém se prolíná kryptografie, informatika a elektroinženýrství. Zaměřuje se na vytváření, adaptaci nebo efektivní implementaci kryptografických primitiv a protokolů pro zařízení s omezeným výpočetním výkonem, kterým jsou například mikrokontroléry nebo RFID[1] čipy. Z důvodu omezeného výkonu nelze na tyto zařízení implementovat standartní kryptografické algoritmy.[14]

Při návrhu algoritmů lehké kryptografie je nutno udělat kompromis mezi bezpečností, náklady a výkonem. Tento kompromis lze vidět na obrázku 3.1.



Obr. 3.1: Kompromis při návrhu algoritmu.[14]

Nejčastěji bývají splněny pouze dva ze tří požadavků na návrh algoritmu a to buď bezpečnost a nízkonákladovost, bezpečnost a výkon nebo výkon a nízkonákladovost. Docílení všech tří požadavků tak, aby byli správně optimalizované, bývá velmi náročné. Například vysokého výkonu na hardwaru a bezpečnosti lze dosáhnout zřetězením architektur, které zakomponovávají několik protiopatření proti útokům postranními kanály. Výsledný návrh bude mít vysoké plošné požadavky, což koreluje s vysokými náklady. Naopak lze navrhnout algoritmus, který je bezpečný a má malé náklady. Tento algoritmus bude omezen výkonem.[14]

Pro zařízení, které využívají lehké kryptografie, byl stanoven požadavek na minimální bezpečnost o velikosti 80 bitů. Tato délka se zdá být nedostatečná, ale vzhledem k informaci, která je chráněna, je dostatečná. Při pokusu o prolomení takového

[1] Identifikace na rádiové frekvenci z anglického Radio Frequency Identification

zabezpečení by útočník musel vynaložit velké úsilí nebo finanční obnos. Zisk z tohoto pokusu by byl ve většině případů zcela adekvátní.[15]

Lehká kryptografie využívá hašovací funkce, proudové a blokové šifry, které byly speciálně pro tuto potřebu vyvinuty.[15]

3.2 Hašovací funkce lehké kryptografie

V lehké kryptografii hašovací funkce fungují na stejném principu jak je uvedeno v kapitole 2.3 Hašovací funkce. Jediný rozdíl je v bitové délce výstupu. V případě hašovacích funkcí, které se používají na zařízeních u kterých není omezený výkon, se využívá výstup hašovací funkce o velikosti 128 bitů a více. Ve většině případů se přešlo na hašovací funkce typu SHA-256 a SHA-512, které mají délku 256 bitů respektive 512 bitů. Hašovací funkce z rodiny SHA jsou nejpoužívanější hašovací funkce v kryptografii.

Lehká kryptografie používá hašovací funkce o velikosti výstupu 80 bitů a více. Nejznámější hašovací funkcí je PHOTON, který má více variant délky výstupu.Tyto délky výstupu jsou v rozsahu od 80 bitů do 256 bitů. Další známou hašovací funkcí je SPONGENT, který nabízí varianty výstupu od 88 bitů do 256 bitů.[16]

3.3 Proudové šifry lehké kryptografie

Proudové šifry pro lehkou kryptografii fungují na stejném principu, který je popsán v kapitole 2.5.1 Proudové šifry. Proudových šifer pro lehkou kryptografii není navrženo mnoho. Nejpoužívanější šifrou je šifra A5/1, jejíž využití se vyčísluje v miliardách. Jedním z důvodů proč jich není vytvořeno mnoho, může být, že stávající proudové šifry jsou několikanásobně méně náročné na implementaci. Z tohoto důvodu lze stávající šifry uzpůsobit jejich implementaci nebo šifru samotnou tak, aby šifry bylo možné použít na výpočetně omezených zařízeních.[13]

Mezi zástupce proudových šifer patří E_0. Tato šifra se často vyskytuje ve sféře mobilních telefonů. Využívá se pro přenos paketů, které udělují důvěrnost při přenosu protokolem Bluetooth. Algoritmus používá klíč, který má délku 128 bitů. Dalším zástupcem je šifra A5/1. Tato šifra se využívá v sítích GSM[2] a je i součástí jejich standardu. Využívá 64bitový klíč a 22bitový inicializační vektor. Dále byla v roce 2007 představena šifra Grain, která se využívá hlavně u zařízení, které mají omezenou paměť a spotřebu energie. Funguje na principu dvou posuvných registrů a nelineární funkci, kdy její klíč má délku 80 bitů. Existuje i verze, která využívá klíč o délce 128 bitů.[13]

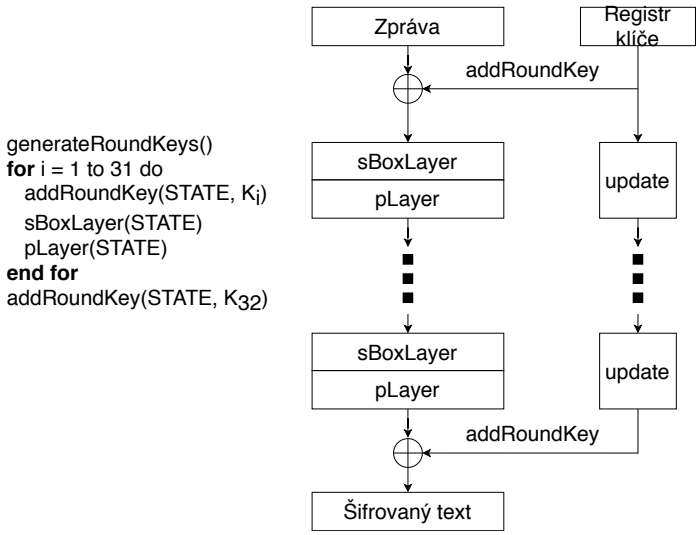
[2]Globální systém pro mobilní komunikaci

3.4 Blokové šifry lehké kryptografie

Princip blokových šifer byl detailně probrán v kapitole 2.5.2 Blokové šifry. Nejznámější blokovou šifrou, která je založena na principu Fiestelovy síťě je šifra DES. Z šifry DES vychází čínská šifra LBlock, která je jednou z hlavních šifer používaných v lehké kryptografii a byla navržena pro zařízení s nedostatkem výpočetního výkonu. Nejznámější blokovou šifrou, která je založena na principu subtitučně-permutační sítě, je šifra AES. Z této šifry vznikla jedna z nejznámějších lehkých šifer – evropská šifra PRESENT. Šifra PRESENT je podobně jako LBlock navržena z ohledem na zařízení, která mají omezené prostředky výpočetního výkonu.[17]

3.4.1 PRESENT

Šifra PRESENT využívá bloky o délce 64 bitů a podporuje velikost klíče 80 a 128 bitů. Ve standardu tvůrci navrhují využití 80bitové varianty. Počet kol opakovaní šifry je 31, v každém kole je použit klíč určený pro dané kolo, který provádí exkluzivní disjunkci (XOR) se vstupním 64bitovým blokem. Následně po použití operace XOR se provádí lineární bitová permutace a nelineární substituce. Nelineární substituce využije jednoho 4bitového S-boxu, který je použit paralelně 16 krát v každém kole. Výsledkem bude šifrovaný blok textu.[18]



Obr. 3.2: Algoritmický princip šifry PRESENT.[18]

Substituční vrstva – využívá 4bitové S-boxy, které paralelně nahrazuje. V tabulce 3.1 je uveden postup nahrazovaní bitů. Na obrázku 3.2 je tato vrstva označena pod názvem sBoxLayer.[18]

Tab. 3.1: Tabulka pro nahrazení S-boxů.[18]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Permutační vrstva – v následují tabulce 3.2 je uveden postup permutace, kdy bit i je předchozí pozice bitu a *P(i)* uvádí novou pozici bitu. Tato vrstva je označena na obrázku 3.2 jako pLayer.[18]

Tab. 3.2: Permutační tabulka.[18]

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Key schedule – podporuje 80bitový nebo 128bitový klíč, v následujícím postupu se zaměříme na 80bitový klíč. Klíč dodaný uživatelem je uložen v registru klíče K, je reprezentován jako posloupnost bitů $k_{79}\,k_{78}\ldots k_0$. V kole i se používá klíč, ve kterém je použito prvních 64 bitů z levé strany uložených v registru K. V kole i se tedy použije klíč:

$$K_i = k_{63}\,k_{62}\ldots k_0 = k_{79}\,k_{78}\ldots k_{16}$$

Po extrakci klíče se registr K upraví:

$$\begin{aligned} k_{63}\,k_{63}\ldots k_{63}\,k_{63} &= k_{18}\,k_{17}\ldots k_{20}\,k_{19} \\ k_{79}\,k_{78}\,k_{77}\,k_{76} &= S(k_{79}\,k_{78}\,k_{77}\,k_{76}) \\ k_{19}\,k_{18}\,k_{17}\,k_{16}\,k_{15} &= (k_{19}\,k_{18}\,k_{17}\,k_{16}\,k_{15}) \oplus \mathsf{round\_counter} \end{aligned}$$

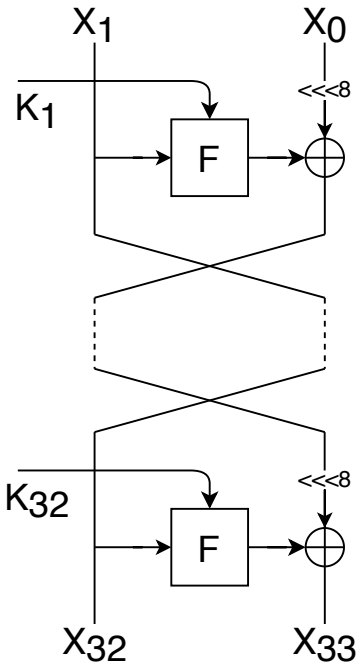
Registr klíče K je posunut o 61 bitů doleva, první čtyři bity z levé strany se použijí k substituci s S-boxem. V poslední části se využije operace XOR mezi bity $k_{19}\,k_{18}\,k_{17}\,k_{16}\,k_{15}$ z registru klíče K a hodnotou round_counter, která udává počet kol i které proběhly.[18]

3.4.2 LBlock

Šifra LBlock využívá bloky o velikosti 64 bitů s klíčem o délce 80 bitů. Počet kol opakovaní je 32. V další části textu budou použity tyto notace[19]:

M	64 bitová zpráva
C	64 bitová šifrovaný text
K	80 bitový klíč
K_i	32 bitový klíč pro kolo i
F	funkce pro kolo
S	S-boxová vrstva
P, P_1	permutační operace pro 32 bitů
$<<< 8$	bitový posun o 8 bitů vlevo

Algoritmus šifrovaní je znázorněn na obrázku 3.3. Na začátku šifrovaní se rozdělí zpráva M na 2 části X_1 a X_0, každá z částí má velikost 32 bitů. Po posledním kole se výsledné dva bloky X_{32} a X_{33} spojí do jednoho 64bitového bloku, který obsahuje šifrovaný text C.[19]



Obr. 3.3: Algoritmický princip šifry LBlock.[19]

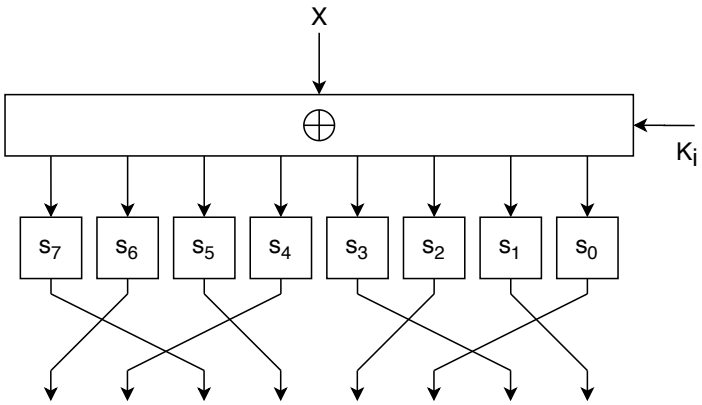
Průběh jednoho kola i lze vyjádřit jako:

$$X_i = F(X_{i-1}, K_{i-1}) \oplus (X_{i-2} <<< 8)$$

Funkci kola F lze vyjádřit jako:

$$\{0,1\}^{32} \times \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$$
$$(X, K_i) \longrightarrow U = P(S(X \oplus K_i))$$

Funkce konfúze S – obsahuje 8 paralelně poskládaných 4×4 S-boxů, jejichž obsah je zobrazen v tabulce 3.3, kde jsou označeny $s_0 \ldots s_7$. Každý blok se rozdělí na části o velikosti 4 bitů. Na každou část je použit S-box. Tímto lze zaručit, že každý bit z 32bitového bloku bude ovlivněn.[19]



Obr. 3.4: Funkce pro kolo šifry LBlock.[19]

Funkce difuze P – je permutace osmi 4 bitových slov, které jsou výstupem funkce S.
Key schedule – 80bitový klíč K je uložen v registru klíče K_r jako posloupnost bitů $K = k_{79}\, k_{78} \ldots k_0$. Klíč pro kolo K_i je vytvořen z prvních 32 bitů z levé strany, kde se následovně K_r posune o 29 bitů vlevo. V další části na prvních 8 bitů z levé strany jsou použity dva S-boxy s_8 a s_9, které lze dohledat v tabulce 3.3. V poslední částí se využije operace XOR mezi bity $k_{50}\, k_{49}\, k_{48}\, k_{47}\, k_{46}$ a hodnotou i, která udává hodnotu aktuálního kola. Výstupem je prvních 32 bitů z levé strany registru K jako klíč K_{i+1}. Postup transformace klíče pro kolo i je zobrazen níže, kde i je pro kola 1 až 31.[19]

(A) $K = k_{79}\, k_{78} \ldots k_{49}\, k_{48}$
(B) $K <<< 29$
(C) $k_{79}\, k_{78}\, k_{77}\, k_{76} = s_9[k_{79}\, k_{78}\, k_{77}\, k_{76}]$
(D) $k_{79}\, k_{78}\, k_{77}\, k_{76} = s_8[k_{79}\, k_{78}\, k_{77}\, k_{76}]$
(E) $k_{50}\, k_{49}\, k_{48}\, k_{47}\, k_{46} \oplus i$
(F) $K_{i+1} = k_{79}\, k_{78} \ldots k_{49}\, k_{48}$

Tab. 3.3: S-box tabulka pro šifru LBlock.[19]

s_0	14	9	15	0	13	4	10	11	1	2	8	3	7	6	12	5
s_1	4	11	14	9	15	13	0	10	7	12	5	6	2	8	1	3
s_2	1	14	7	12	15	13	0	6	11	5	9	3	2	4	8	10
s_3	7	6	8	11	0	15	3	14	9	10	12	13	5	2	4	1
s_4	14	5	15	0	7	2	12	13	1	8	4	9	11	10	6	3
s_5	2	13	11	12	15	14	0	9	7	10	6	3	1	8	4	5
s_6	11	9	4	14	0	15	10	13	6	12	5	7	3	8	1	2
s_7	13	10	15	0	14	4	9	11	2	1	8	3	7	5	12	6
s_8	14	9	15	0	13	4	10	11	1	2	8	3	7	6	12	5
s_9	4	11	14	9	15	13	0	10	7	12	5	6	2	8	1	3

4 Výběr algoritmu

V této kapitole je popsán výběr vhodného algoritmu k implementaci na FPGA. První část se zaměří na výběr podle typu šifer tak, aby byla vybrána vhodná šifra, která se hodí pro šifrovaní souborů a dat uložených na mediích. Druhá část popisuje výběr konkrétní šifry z vybraného typu šifer. Výběr se zabývá jak jejich hardwarovými požadavky, tak i základními požadavky na bezpečnost.

4.1 Výběr typu šifry

Výběr probíhal ze tří typů šifer – hašovací funkce, proudové a blokové šifry. Postupně budou popsány klady a zápory každého typu šifer a nakonec bude uvedena vybraná šifra.

Hašovací funkce mají výhodu v jejich rychlosti a jednoduchosti. Další předností je, že jestliže je vstupní blok dat vždy různě dlouhý, výsledek hašovací funkce bude mít vždy stejnou výstupní délku. Hlavní nevýhoda spočívá v jednosměrnosti hašovacích funkcí. Jelikož je nemožné z hašovací funkce získat původní data z tohoto důvodu se tyto funkce používají nejčastěji pro ověření elektronických podpisů. Z důvodu jednosměrnosti jsou hašovací funkce pro potřeby této práce nevhodné, protože budou šifrována data, ze kterých by mělo být možné získat původní hodnotu, která byla zašifrována.

Výhodou proudových šifer je jejich malá hardwarová náročnost, protože šifrují data bit po bitu. Hlavní nevýhodou je možnost změny dat při narušení přenosu. Šifry tohoto typu se nejčastěji využívají při přenosu dat přes síť, kde se přenáší velké objemy dat mezi jednotlivými body. Jelikož při implementaci šifry na FPGA se nepočítá se šifrováním dat síťového provozu, označil bych tento typ šifer také za nevhodný.

U blokových šifer je hlavní výhodou, že se šifrují po určitém stejném bloku dat, kde každý bit v těchto datech je šifrován mezi několik bitů ve výsledném šifrovaném textu. Další nesporný klad spočívá v problematické záměně dat v šifrované zprávě. Při záměně části bitů v šifrovaném textu lze po dešifrovaní rozpoznat, jestli s daty bylo manipulováno. Jedinou možností útočníka by pak bylo upravená data šifrovat pomocí stejného algoritmu a klíče. Nevýhodou je pomalejší šifrovaní na rozdíl od výše zmíněných typů. Chyba při šifrovaní neovlivní pouze daný bit nebo bity, ale celý šifrovaný blok dat. Tyto šifry jsou vhodné pro šifrovaní dat, u kterých je známa délka.

Pro implementaci šifrovaní souborů uložených na mediích z těchto tří popsaných typů a z výše uvedených kladů a záporů lze říci, že pro šifrovaní souborů, u kterých známe předem velikost, vychází blokové šifry jako nejvhodnější. Hašovací funkce

nemohou být použity kvůli své jednosměrnosti, proudové šifry nejsou často užívány k šifrovaní souborů, které se nepřenáší přes síť.

4.2 Výběr šifry

Blokových šifer pro lehkou kryptografii je velké množství, výběr byl proveden z následujících šifer – LBlock, LED, mCRYPTON, PRESENT, SIMON. Tyto šifry jsou nejznámějšími zástupci blokových šifer pro lehkou kryptografii.

Tab. 4.1: Porovnaní blokových šifer z hlediska výkonu [13].

Název	Oblast [GE]	Propustnost při 100 KHz [kb/s]	Efektivnost [bps/GE]
LBlock	1320	200,00	48,69
LED-64	966	5,10	5,28
LED-128	1265	3,40	2,69
mCRYPTON-64	2420	492,31	203,43
mCRYPTON-96	2681	492,31	183,63
mCRYPTON-128	2949	492,31	119,84
PRESENT-80	1570	200,00	127,39
PRESENT-128	1884	200,00	106,16
SIMON 32/64	566	22,20	39,22
SIMON 48/96	763	15,00	19,66
SIMON 64/96	838	17,80	21,24
SIMON 64/128	1000	16,70	16,70
SIMON 96/96	984	14,80	15,04
SIMON 128/128	1317	22,90	17,31
SIMON 128/256	1883	21,10	11,21

V tabulce 4.1 jsou vypsány šifry a jejich varianty dle alfabetické posloupnosti. Gate equivalent (GE) je jednotka, která udává počet požadovaných integrovaných obvodů (IC). Je odvozena dělením plochy IC s oblastí negovaného logického součinu (NAND) s nejnižší řídící silou.

Při srovnání hodnot GE jsou na implementaci nejvhodnější šifry SIMON, kdy čísla uvedená v názvu udávají velikost bloku a velikost klíče. Pro upřesnění šifry SIMON 32/64, která zabírá na obvodech nejméně plochy (566 GE), má velikost bloku 32 bitů a velikost klíče 64 bitů. Nejvíce náročnou šifrou je mCRYPTON ve všech jejich variantách. Navzdory hardwarové náročnosti je její nespornou výhodou

propustnost, která má hodnotu 492,31 kb/s pro všechny její varianty. Nejmenší propustnost má šifra LED, kterou následuje šifra SIMON. Obě tyto šifry mají podobné požadavky na využitou oblast, ale šifra SIMON je všech kategoriích o poznaní lepší. K propustnosti se váže i efektivnost, kdy pozice nejlepších a nejhorších cifer zůstávají nezměněné. Nejvyváženějšími šiframi jsou LBlock a PRESENT. Tyto šifry nemají vysoké nároky na použitou oblast, ale zároveň dosahují několikanásobné propustnosti oproti šifrám s podobnou hodnotou GE. Při porovnání efektivnosti šifer LBlock a PRESENT je šifra PRESENT značně efektivnější než LBlock, i když obě dvě mají stejnou propustnost. Tento rozdíl je tvořen v odlišnosti návrhu šifer, jelikož PRESENT využívá substitučně-permutační síť a LBlock Fiestelovu síť.

Tab. 4.2: Porovnaní blokových šifer z hlediska bezpečnosti [13].

Název šifry	Délka bloku [bit]	Délka klíče [bit]	Počet kol	Počet překonaných kol	
LBlock	64	80	21	23	72 %
LED-64	64	64	32	14	54 %
LED-128	64	64	48	23	67 %
mCRYPTON-64	64	64	13	7	54 %
mCRYPTON-96	64	96	13	7	54 %
mCRYPTON-128	64	128	13	7	54 %
PRESENT-80	64	80	31	26	84 %
PRESENT-128	64	128	31	26	84 %
SIMON 32/64	32	64	32	23	72 %
SIMON 48/96	48	96	36	25	69 %
SIMON 64/96	64	96	42	30	71 %
SIMON 64/128	64	128	44	31	70 %
SIMON 96/96	96	96	52	37	71 %
SIMON 128/128	128	128	68	49	72 %
SIMON 128/256	128	256	72	53	74 %

V tabulce 4.2 jsou porovnány šifry z pohledu bezpečnosti. Nejbezpečnější šifrou lze označit LED, u níž se podařilo překonat pouze polovinu z 32 kol pro její 64bitovou verzi klíče. Za druhou nejbezpečnější lze považovat šifru mCRYPTON, kde bylo překonáno 7 kol ze 13. Nejméně bezpečnou šifrou z vybraných je PRESENT, kdy zbývalo překonat 5 kol z 31, aby byla šifra prolomena. Šifry LBlock a SIMON se pohybují kolem hranice 70% překonaných kol. Vezmeme-li v úvahu, že z pohledu bezpečnosti je výhodnější delší klíč jeví se šifra SIMON 128/256 jako nejvhodnější v kategorii podle délky klíče. Následují SIMON, PRESENT, mCRYPTON a LED

v jejich variantách využívající 128bitového klíče. Jestliže spojíme tyto dvě informace dohromady, lze šifru mCRYPTON označit jako nejbezpečnější, po ní následuje šifra LED ve variantě 128 bitů. Zbytek šifer bych označil jako velmi podobných, kdy šifry SIMON s klíči o velikostech 256 a 128 jsou ze zbylých šifer nejbezpečnější. Za nejméně bezpečnou šifru lze považovat SIMON v 64bitové variantě.

Do úvahy vezmeme obě dvě porovnání, z nichž budeme vybírat nejvhodnější šifru, která je jak bezpečná tak dostatečně efektivní a zároveň nezabírá velkou oblast. Jak bylo uvedeno v předchozí části, šifra SIMON zabírá nejmenší plochu, dá se považovat za bezpečnou, ale její efektivnost a propustnost zůstává velice malá. Šifru mCRYPTON lze považovat za jednu z nejbezpečnějších, její propustnost a efektivnost je nejlepší z porovnávaných šifer, ale její oblast, kterou potřebuje k implementaci je nejméně dvojnásobná oproti ostatním implementacím. Šifra LED je podobná šifře SIMON v porovnávaných hodnotách s rozdílem její bezpečnosti, kdy její 128bitovou variantu lze považovat za druhou nejbezpečnější. Šifra PRESENT byla společně s LBlock vybrána jako nejoptimálnější šifry z pohledu výkonu. Z hlediska bezpečnosti lze obě dvě šifry považovat za dostatečně bezpečné. Pokud porovnáme tyto dvě šifry vzájemně z pohledu výkonu, je jejich propustnost zcela stejná. Šifra PRESENT je lepší v kategorii efektivnosti, ale obě její varianty mají větší nároky na použitou oblast. Z pohledu bezpečnosti mají obě dvě šifry délku bloku 64 bitů a jejich nejmenší varianty mají 80 bitů. Jedinou výhodou šifry LBlock je, že má méně překonaných kol než šifra PRESENT.

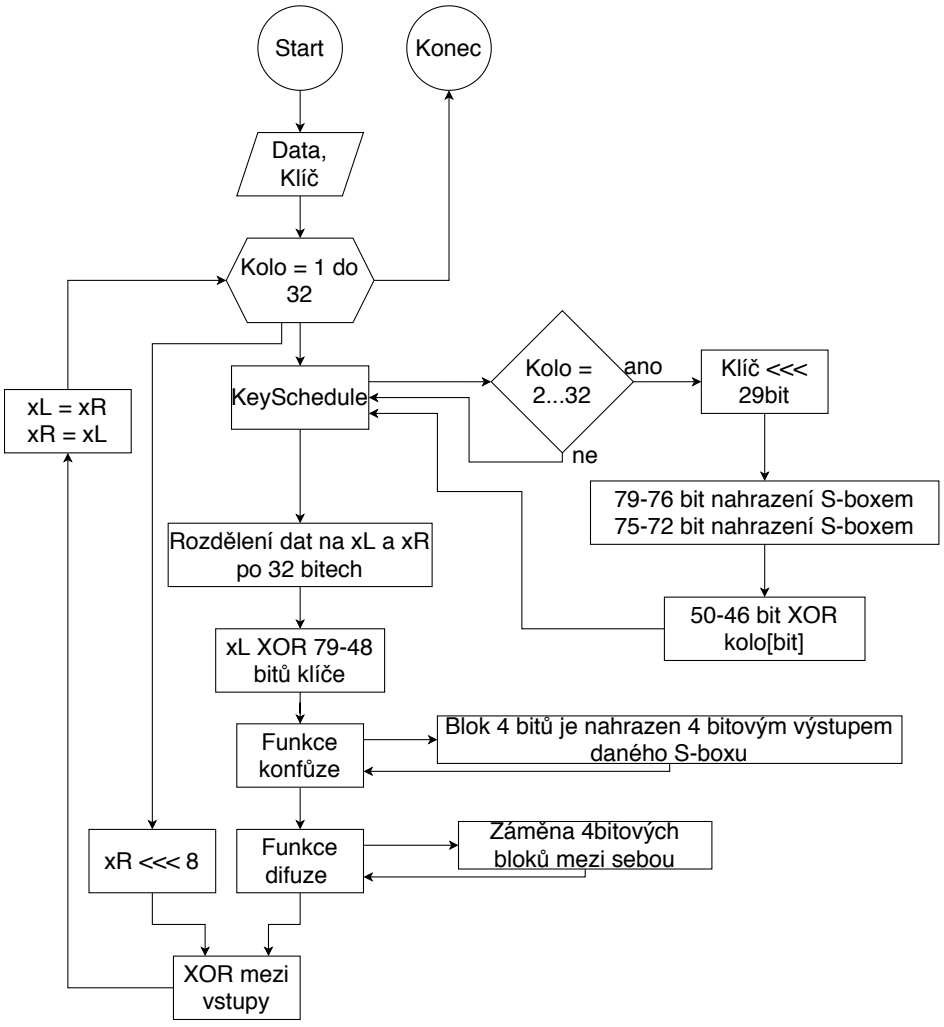
Porovnáním kladů a záporů všech šifer lze docílit, že konečné rozhodnutí bude výběr mezi šiframi LBlock a PRESENT, tyto šifry jsou nejvíce optimální z obou kategorií. Na základě předchozích porovnání byla vybrána šifra LBlock, z důvodu jejího menšího požadavku na oblast. Dalším kriteriem byla její lepší bezpečnosti z pohledu známých překonání počtu kol.

5 Implementace na FPGA

V této kapitole se řeší přímá implementace vybrané šifry LBlock, detailně popsána v kapitole 3.4.2 LBlock, na FPGA čip, který je ipmlementován na vývojové desce ZYBO Z7. Algoritmus je do strojové podoby přepsán pomocí jazyka VHDL[1], který slouží k popsaní hardwarové sekvence instrukcí.

5.1 Implentace šifry LBlock pomocí VHDL

Implementace se skládá z částí, které jsou vytvořeny podle vývojového diagramu, zobrazeného na obrázku 5.1.



Obr. 5.1: Vývojový diagram šifry LBlock.

[1]VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language

Tyto části budou podrobněji probrány v následujících podkapitolách, pod názvy RoundFunction, ConfusionFunction, DiffusionFunction, KeySchedule, LBlockLoop a LBlockTop.

5.1.1 Implementace KeySchedule

Implementace tohoto procesu je založena na úpravě klíče v každém kole, kromě prvního. V prvním kole se vezme prvních 32 bitů z levé strany vstupního klíče. V následujících kolech to jsou kola 2-32. V těchto kolech probíhají operace bitového posunu jako rotace o 29 bitů. Tento posun je na řádku 1 ve výpisu kódu 5.1. Zde je využit vnitřní signál procesu, který slouží jako proměnná pro bitový posun klíče. Funkce S9 řádek 3 a S8 řádek 9 je využití S-boxů, které se aplikují na prvních 8 bitů rozdělených do dvou stejně dlouhých bitových bloků. Řádky 15 až 18 definují zbylých 72 bitů, kdy se s bity 50-46 a bitovou hodnotou aktuálního kola provádí operace XOR.

Výpis 5.1: Key schedule

```
1  switched <= key_in(50 downto 0) & key_in(79 downto 51);
2
3      S9 : Sbox9
4          port map (
5              data_in => switched(79 downto 76),
6              data_out => key_out(79 downto 76)
7      );
8
9      S8 : Sbox8
10         port map (
11             data_in => switched(75 downto 72),
12             data_out => key_out(75 downto 72)
13     );
14
15     key_out(71 downto 51) <= switched(71 downto 51);
16     key_out(50 downto 46) <= switched(50 downto 46)
17     XOR round_value(4 downto 0);
18     key_out(45 downto 0) <= switched(45 downto 0);
```

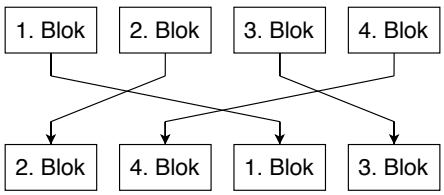
5.1.2 Implementace ConfusionFunction

Implementace modulu ConfusionFunction funguje na principu tzv. obalu neboli modul volá několik submodulů. Vstupem uvedeného modulu je blok o velikosti 32 bitů.

Tento modul následně volá 8 submodulů, které implementují dané S-boxy $s_7 - s_0$. Vstupem každého S-boxu je 4bitový blok, kde do S-boxu s_7 jdou první 4 nejvýznamnější bity a následně se k dalším S-boxům přiřazují 4bitové bloky sestupně od nejvýznamnějšího bloku. Výstupem těchto submodulů jsou přeměněné bloky bitů, které se následně spojují do výstupního 32bitového bloku v pořadí $s_7 - s_0$. Tento 32bitový blok je výsledným výstupem modulu ConfusionFunction.

5.1.3 Implementace DiffusionFunction

Modul DiffusionFunction je jednoduchou implementací funkce difuze. Funguje na principu vstupu 32bitového bloku dat. Blok je rozdělen na menší 4bitové bloky, které se následně vyměňují mezi sebou. Tato výměna se vždy opakuje po čtyřech 4bitových blocích. Výměna probíhá následovně tak, že první blok se přemísťuje na třetí pozici, druhý blok se přemísťuje na první pozici, třetí blok na čtvrtou pozici a čtvrtý blok na druhou pozici. Popsaný proces výměny proběhne dvakrát tzn. že pro každou polovinu vstupního bloku daný postup bude vykonán jednou. Po procesu výměny se všechny tyto 4bitové bloky spojí do 32bitového výstupního bloku, který je zároveň výstupem uvedeného modulu. Na obrázku 5.2 je znázorněn postup výměny čtyř 4bitových bloků.



Obr. 5.2: Výměna čtyř 4bitových bloků.

5.1.4 Implementace RoundFunction

Modul RoundFunction se stará o imlementaci funkce kola F. Vstupem do modulu je aktuální klíč kola a blok dat z levé strany Fiestelovy sítě. V počáteční části se zavolá submodul, který má za úkol vyextrahovat ze vstupního 80bitového klíče prvních 32 bitů. Tento submodul je vytvořen hlavně z důvodu přehlednosti a čitelnosti kódu. Vyextrahovaný klíč se uloží do interního signálu, nesoucí název key32bit. Následně je vypočítán interní signál resultOfXOR, který je výsledkem operace XOR mezi key32bit a vstupním blokem dat. Volá se submodul Sbox popsaný v 5.1.2 Implementace ConfusionFunction. Jeho vstupem je signál resulOfXOR a výstupem interní signál resultOfSbox. V poslední části se volá modul popsaný v 5.1.3 Implementace DiffusionFunction, který využívá signál resultOFSbox jako vstupní blok

dat. Výstupní data modulu, který implementuje funkci difuze, jsou uložena do výstupu modulu RoundFunction.

5.1.5 Implementace LBlockLoop

V tomto modulu se uskutečňuje celé kolo pro šifru LBlock. Vstupem je 80bitový klíč, levá a pravá strana Fiestelovy sítě. Výstupem je pravá strana, která je rovna hodnotě na vstupu levé strany a levá strana, která je výstupem úprav kola. Tyto úpravy se uvádí v postupných krocích. Nejprve se zavolá modul RoundFunction, který byl popsán v 5.1.4 Implementace RoundFunction. Data pravé vstupní strany, která se upraví bitovou rotací o 8 bitů, se uloží do interního signálu shifted. Na tento signál společně s výstupem RoundFunction je následně použita operace XOR. Výsledek exkluzivní disjunkce je použit na výstup levé strany. Celý postup ve VHDL je znázorněn na výpisu kódu 5.3.

Výpis 5.2: LBlockLoop

```
1  RF : RoundFunction
2      port map (
3          key_in => fullKey_in,
4          xL_in => xLdata_in,
5          xL_out => xLhelp
6      );
7
8  shifted <= xRdata_in(23 downto 0) &
9             xRdata_in(31 downto 24);
10
11 dataHelp <= shifted XOR xLhelp;
12
13 xLdata_out <= dataHelp;
14 xRdata_out <= xLdata_in;
```

5.1.6 Implementace LBlockTop

V modulech obsahující slovo „top“ bývá nejčastěji uložena ovládací logika ostatních modulů. Moduly s tímto přívlastkem se používají k lepší organizaci hierarchie souborů. Tento modul implementuje časování každého kola pomocí signálu cls (clock). Při nástupu hrany cls neboli změny hodnoty z 0 na 1, se provede jedno celé kolo šifry LBlock a úpravy klíče. Signál nxState se skládá ze tří stavů, tyto stavy jsou WAITDATA, ROUND a FINAL. Když je signál ve stavu WAITDATA, načítá se

hodnota klíče a data do interních signálů. Dále se nastaví hodnota kola a nový stav ROUND. Při stavu ROUND se nastavují hodnoty signálů z volaných modulů KeySchedule a LblockLoop, aby mohly být použity jako vstupní signály při další nástupní hraně clk. Dále se zde inkrementuje hodnota kola a kontroluje se zda hodnota kola nedosáhla 32. Jestliže hodnota je rovna 32, data se zapíší na výstup a hodnota stavu se změní na FINAL. Nastane-li hodnota menší než 32, stav se nezmění a klíč pro další kolo bude předán signálu klíče.

Výpis 5.3: LBlockTop

```
1  process(clk, reset)
2  begin
3      elsif CLK'EVENT and clk = '1' then
4          case nxState is
5              when WAITDATA =>
6                  RoundFullKey <= Key_in;
7                  xLdata_in_intern <= data_in(63 downto 32);
8                  xRdata_in_intern <= data_in(31 downto 0);
9                  busy <= '1';
10                 nxState <= ROUND;
11                 roundCounter <= "000001";
12
13             when ROUND =>
14                 xLdata_in_intern <= xLdata_out_intern;
15                 xRdata_in_intern <= xRdata_out_intern;
16                 roundCounter <= roundCounter + '1';
17                if roundCounter = "100000" then
18                     data_out <= xRdata_out_intern &
19                                 xLdata_out_intern;
20                     nxState <= FINAL;
21                 else
22                     nxState <= ROUND;
23                     roundFullKey <= roundKeyHelp;
24                 end if;
25         end case;
26     end if;
27 end process;
```

5.2 Testovaní správné implementace šifry

Testovaní je založeno na kontrole výstupních dat pomocí simulace ve Vivado Design Suite. Existují dvě metody zadávání vstupů do simulace. Těmito metodami je zadávání vstupů automatizovaně nebo manuálně.

Manuální zadávání vstupních signálů je vhodné, jestliže máme malý vzorek vstupů a neprovádíme tyto testy opakovaně. V dlouhodobém časovém úseku jsou tyto testy považovány za příliš komplikované, jelikož při každém testu musíme nastavit dané hodnoty manuálně.

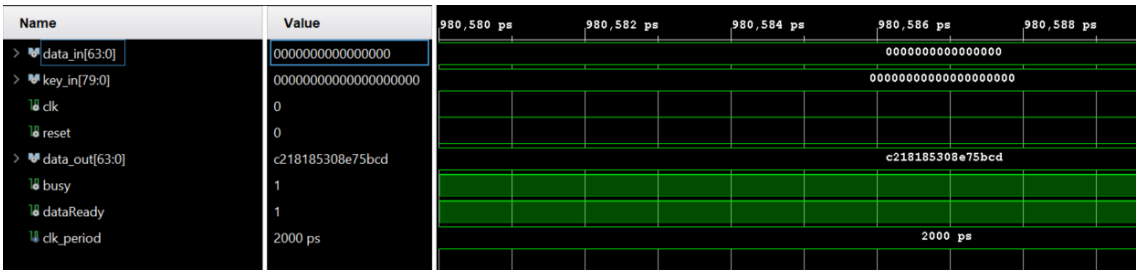
Automatizované testy se využívají ke kontrole více hodnot a jakmile jsou napsány, lze je opakovaně simulovat. Testy se implementují pomocí tzv. testovacího stolu (anglicky testbench). Výsledky těchto testů mohou být kontrolovány manuálně nebo automatizovaně. V práci jsou použity pouze testy s manuální kontrolou. Výsledky simulace jsou manuálně kontrolovány s dodanými tabulkami v dokumentaci šifry (*Wenling, 2011* [19]) nebo vlastnoručně vypočtenými hodnotami.

Za nejdůležitější lze považovat testovaní implementace celé šifry, která se nachází v modulu LBlockTop. Zde byly aplikovány dvě testovací hodnoty vytvořené tvůrci šifry LBlock, které byli využity pro testovaní. Jedná se dva vstupní bloky a dva klíče, jak je patrné z tabulky 5.1.

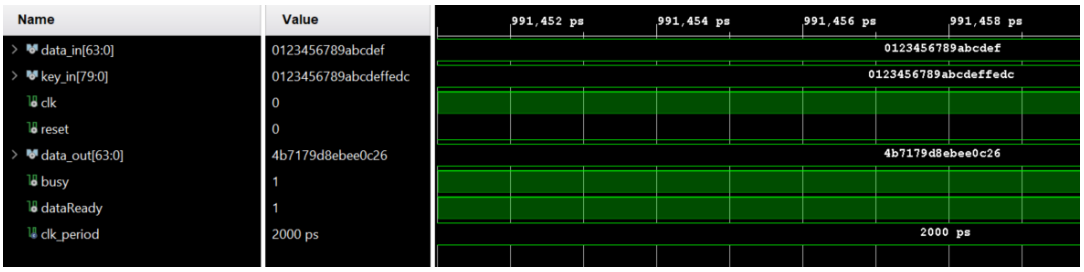
Tab. 5.1: Výsledné vektory pro LBlock v hexadecimalním tvaru [19].

	Zpráva	Klíč	Šifrovaný text
1	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	c2 18 18 53 08 e7 5b cd
2	01 23 45 67 89 ab cd ef	01 23 45 67 89 ab cd ef fe dc	4b 71 79 d8 eb ee 0c 26

Hodnoty zprávy a klíče z tabulky 5.1 byli použity jako vstupy, kdy výsledný šifrovaný text odpovídal hodnotám dosažených v testové simulaci. Oba dva výsledky lze vidět na obrázku 5.3 pro zprávu 1, respektive na obrázku 5.4 pro zprávu 2.



Obr. 5.3: Výsledek simulace pro zprávu 1.



Obr. 5.4: Výsledek simulace pro zprávu 2.

Závěr

Cílem této semestrální práce bylo seznámení se šiframi lehké kryptografie a implementace vybrané šifry do jazyka VHDL. Šifry byli nejprve voleny tak, aby splňovali požadavek na šifrovaní souborů šifrou lehké kryptografie. Z důvodu tohoto požadavku jsem zvolil blokové šifry, které jsou nejlepší proto, že je lze na rozdíl od hašovacích funkcí dešifrovat. Vybíral jsem mezi šiframi LBlock, LED, mCRYPTON, PRESENT a SIMON. Dospěl jsem k závěru, že nejvhodnější šifrou pro implementaci je šifra LBlock, kterou následuje šifra PRESENT. Uvedené šifry tvoří nejoptimálnější dvojici co se týká poměru náklady, výkon a bezpečnost. Postup šifrovaní těchto dvou šifer byl detailněji rozebrán z důvodu, že každá z šifer využívá jinou metodu šifrovaní.

Zvolená šifra LBlock založená na Feistelově síti byla implementována ve vývojovém prostředí Vivado Design Suite. Celá šifra byla rozdělena na několik modulů starajících se o logiku transformací vstupních dat. Výstupem hlavního modulu, který je založen na sekvenčním postupném šifrovaní, jsou šifrovaná data. Ke každému modulu byli implementovány simulační testy, kdy výstupní hodnoty z hlavního modulu odpovídaly hodnotám v dokumentaci šifry LBlock.

Semestrální práce bude v navazující bakalářské práci rozšířena o implementaci na vývojovou desku ZYBO Z7. Ze vstupů desky, jako jsou např. rozhraní USB 2.0, Micro SD a jiné, budou následně načítána data. Vstupní data budou šifrovány a uloženy zpět na tyto vstupy.

Pokračovaní mé práce je také možné rozšířit o přidaní šifry PRESENT a jejím porovnaní se šifrou LBlock. U tohoto rozšíření lze měřit rychlost provedení šifry na stejných vstupních datech. Dále lze provádět měření využití šifer pomocí hodnoty LUT (LookUp Table).

Závěrem bych chtěl poděkovat Ing. Davidu Smékalovi za cenné rady a připomínky i odbornou a metodickou pomoc, kterou mně poskytl.

Literatura

[1] SEKANINA, Lukáš, 2004. *Evolvable components: from theory to hardware implementations.* Berlin: Springer-Verlag. ISBN 3-540-40377-9.

[2] CHANDRASEKHAR, Vikram, 2007. *CAD for a 3-dimensional FPGA* [online]. [cit. 16. 10. 2020]. Dostupné z URL: <`http://dspace.mit.edu/handle/1721.1/40520`> Diplomová práce. Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science.

[3] Introduction to FPGA | Structure, Components, Applications, 2020, *Electronicshub.org* [online]. [cit. 16. 10. 2020]. Dostupné z URL: <`https://www.electronicshub.org/wp-content/uploads/2020/01/FPGA-Structure.jpg`>

[4] XILINX, INC. *Zynq-7000 SoC Data Sheet: Overview* [online]. DS190 (v1.11.1) July 2, 2018. [cit. 17. 10. 2020]. Dostupné z URL: <`https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf`>

[5] BURDA, Karel, 2015. *Úvod do kryptografie.* Brno: Akademické nakladatelství CERM. ISBN 978-80-7204-925-7.

[6] BURDA, Karel, 2013. *Aplikovaná kryptografie.* Brno: VUTIUM. ISBN 978-80-214-4612-0.

[7] MAO, Wenbo, 2004. *Modern Cryptography: Theory and Practice.* 5th ed. Upper Saddle River: Prentice Hall. ISBN 0-13-066943-1.

[8] SMART, Nigel, 2003. *Cryptography: An Introduction.* London: McGraw-Hill College. ISBN 9780077099879.

[9] HAVLÍČEK, Jiří, 2013. *Šifrovací algoritmy lehké kryptografie* [online]. Brno [cit. 22. 10. 2020]. Dostupné z URL: <`https://www.vutbr.cz/studenti/zav-prace/detail/66565`> Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Vlastimil Člupek.

[10] FONTAINE, Caroline, 2011. Synchronous Stream Cipher. *Encyclopedia of Cryptography and Security* [online]. Boston, MA: Springer [cit. 22. 10. 2020]. ISBN 978-1-4419-5905-8. Dostupné z URL: doi: <`https://doi.org/10.1007/978-1-4419-5906-5_376`>

[11] FONTAINE, Caroline, 2011. Self-Synchronizing Stream Cipher. *Encyclopedia of Cryptography and Security* [online]. Boston, MA: Springer [cit. 22. 10. 2020].

ISBN 978-1-4419-5905-8. Dostupné z: doi: <https://doi.org/10.1007/978-1-4419-5906-5_371>

[12] Feistel cipher. *Wikipedia* [online]. [cit. 3. 11. 2020]. Dostupné z URL: <https://en.wikipedia.org/wiki/Feistel_cipher#/media/File:Feistel_cipher_diagram_en.svg>

[13] ŠALDA, Jakub, 2017. *Lehká kryptografie* [online]. Praha [cit. 6. 11. 2020]. Dostupné z URL: <https://vskp.vse.cz/69739_lehka_kryptografie.> Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky. Vedoucí práce RNDr. Radomír Palovský, CSc.

[14] POSCHMANN, Axel, 2009. *LIGHTWEIGHT CRYPTOGRAPHY: Cryptographic Engineering for a Pervasive World* [online]. Bochum [cit. 3. 1. 2020]. Dostupné z URL: <https://www.ei.ruhr-uni-bochum.de/media/crypto/veroeffentlichungen/2011/09/16/thesisp.pdf> Disertační práce. Ruhr-University Bochum, Germany, Faculty of Electrical Engineering and Information Technology.

[15] KLÍMA, Vlastimil, 2011. *Co je to lehká kryptografie?* [online]. [cit. 3. 11. 2020]. Dostupné z URL: <https://cryptography.hyperlink.cz/2011/ST_2011_10_16_17.pdf>

[16] B.T. HAMMAD, N. JAMIL, M.E. RUSLI a M.R Z'ABA, 2017. A survey of Lightweight Cryptographic Hash Function. *International Journal of Scientific & Engineering Research* [online]. 2017(8) [cit. 5. 11. 2020]. ISSN 2229-5518. Dostupné z URL: <https://www.ijser.org/researchpaper/A-survey-of-Lightweight-Cryptographic-Hash-Function.pdf

[17] NEKUŽA, Karel, 2016. *Odlehčená kryptografie pro embedded zařízení* [online]. Brno [cit. 6. 11. 2020]. Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace?zp_id=93665.> Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Zdeněk Martinásek, Ph.D.

[18] BOGDANOV Andrey, L. R. KNUDSEN, G. LEANDER, Christof PAAR, Axel POSCHMANN, M.J.B. ROBSHAW, Y. SEURIN a C. VIKKELSOE, 2007. *PRESENT: An Ultra-Lightweight Block Cipher. Cryptographic Hardware and Embedded Systems - CHES 2007.* [online]. Berlin: Springer, Berlin, Heidelberg, s. 455-466 [cit. 7. 11. 2020]. ISBN 978-3-540-74735-2. Dostupné z: doi: <https://doi.org/10.1007/978-3-540-74735-2_31>

[19] WENLING, Wu a Lei ZHANG, 2011. LBlock: A Lightweight Block Cipher. Applied Cryptography and Network Security [online]. Germany: Springer, Berlin, Heidelberg, s. 327-344 [cit. 2020-11-08]. ISBN 978-3-642-21554-4. Dostupné z: doi: <https://doi.org/10.1007/978-3-642-21554-4_19>

Seznam symbolů, veličin a zkratek

FPGA Programovatelné hradlové pole – Field-programmable Gate Array

ASIC Integrované obvody určené pro aplikaci – Application Specific Integrated Circuit

IOB Vstupně/výstupní blok – Input/Output block

PIB Programovatelný propojovací blok – Proglammable Interconnect Blocks

CLB Konfigurovatelné logické bloky – Configurable Logic Blocks

SRAM Statická paměť s přímým přístupem – Static Random Access Memory

ARM Pokročilý procesory s redukovanou instrukční sadou – Advanced Reduced Instruction Set Computer Machine

RSA Šifra Rivest–Shamir–Adleman – Rivest–Shamir–Adleman cipher

SPN Substitučně-permutační síť – Substitution–permutation network

S-box Zaměňovací tabulka – Substitution box

P-box Permutační tabulka – Permutation box

SHA Bezpečný hašovací algoritmus – Secure Hash Algorithm

GSM Globální systém pro mobilní komunikaci – Global System for Mobile Communications

DES Standart datového šifrovaní – Data Encryption Standard

AES Standard pokročilého šifrování – Advanced Encryption Standard

XOR Exkluzivní disjunkce – Exclusive Disjunction

GE Ekvivalent brány – Gate Equivalent

IC Integrovaný obvod – Integrated Circuit

NAND Negovaný logický součin – Negation of Logical Conjunction

VHDL Hardwarový deskriptivní jazyk pro velmi rychlé integrované obvody – Very High Speed Integrated Circuit Hardware Description Language

LUT Vyhledávací tabulka – Look-Up Table

A Obsah přilohy

Tato kapitolo obsahuje popis příloh. Přílohy obsahují z velké části kód VHDL, kterým byl implementovány jednotlivé části šifry LBlock a jejích testovací moduly. Zdrojové kódy jsou napsány v jazyce VHDL 2008 umístěné ve složce vhd. Tato složka se dělí na dvě části. Těmito složkami jsou sources a simulations. Složka sources obsahuje zdrojové kódy implementace šifry LBlock a složka simulations obsahuje zdrojové kódy pro simulaci a testovaní jednotlivých modulů. Mimo zdrojové kódy se nachází v kořenovém adresáři také elektronická verze této bakalářské práce, zdrojové soubory pro LaTeXa soubor README.txt obsahující popis příloh.

Výpis z důvodu přehlednosti a celistvosti je vysázen na následující straně. Všechny soubory lze také nalézt v online repozitáři na Githubu s url adresou `https://github.com/jedla97/HW-encoder-for-lightweigth-cryptography`.

- `/` — kořenový adresář
  - `vhd` — zdrojové soubory
    - `sources` — zdrojové soubory hlavní logiky
      - `DiffusionFunction.vhd`
      - `KeySchedule.vhd`
      - `LBlockLoop.vhd`
      - `LBlockTOP.vhd`
      - `RoundFunction.vhd`
      - `RoundKey.vhd`
      - `Sbox.vhd`
      - `Sbox0.vhd`
      - `Sbox1.vhd`
      - `Sbox2.vhd`
      - `Sbox3.vhd`
      - `Sbox4.vhd`
      - `Sbox5.vhd`
      - `Sbox6.vhd`
      - `Sbox7.vhd`
      - `Sbox8.vhd`
      - `Sbox9.vhd`
    - `simulations` — zdrojové soubory pro simulaci
      - `DiffusionFunction_tb.vhd`
      - `KeySchedule_tb.vhd`
      - `LBlockLoop_tb.vhd`
      - `LBlockTOP_tb.vhd`
      - `RoundFunction_tb.vhd`
      - `RoundKey_tb.vhd`
      - `Sbox_tb.vhd`
      - `Sbox0_tb.vhd`
      - `Sbox1_tb.vhd`
      - `Sbox2_tb.vhd`
      - `Sbox3_tb.vhd`
      - `Sbox4_tb.vhd`
      - `Sbox5_tb.vhd`
      - `Sbox6_tb.vhd`
      - `Sbox7_tb.vhd`
      - `Sbox8_tb.vhd`
      - `Sbox9_tb.vhd`
  - `bakalarska_prace_Jedlicka_Jakub.pdf` — elektronická verze bakalářské práce
  - `latex` — zdrojové soubory práce v LaTeXu
    - `jedlicka_HW_sifrator.zip` — Zazipované soubory pro LaTeX
    - `jedlicka_HW_sifrator` — soubory pro lokální použití LaTeXu
  - `README.txt` — popis příloh