FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Výběr bitů pro SRAM PUF levného mikrokontroléru
Student:	Bc. Gabriela Hánová
Vedoucí:	prof. Ing. Róbert Lórencz, CSc.
Studijní program:	Informatika
Studijní obor:	Návrh a programování vestavných systémů
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Nastudujte metody výběru bitů pro PUF (fyzikálně neklonovanou funkci) realizovanou pomocí SRAM na nízkonákladovém mikrokontroléru (například STM32F030R8).
Navrhněte vlastní metodu vyhodnocení PUF na vybraném hardwaru a proveďte její implementaci.
Experimentálně vyhodnoťte efektivitu a spolehlivost navržené metody vzhledem k fyzikálním vlivům prostředí (např. teplota).

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 7. října 2019

Poděkování

Děkuji svému vedoucímu práce prof. Ing. Róbertu Lórenczovi, CSc., za pečlivé a trpělivé vedení práce a časté konzultace. Dále bych chtěla poděkovat Ing. Jiřímu Bučkovi za pomoc v počátcích implementace této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. srpna 2020 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2020 Gabriela Hánová. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Hánová, Gabriela. *Výběr bitů pro SRAM PUF levného mikrokontroléru.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce se věnuje fyzicky neklonovatelným funkcím (Physical Unclonable Function – PUF) a metodám výběru bitů pro PUF na nízkonákladovém mikrokontroléru.

Nejprve je provedena rešerše týkající se problematiky PUF se zaměřením na SRAM mikrokontroléru. Následně je popsán návrh metody výběru bitů pro PUF a jeho praktické použití pro generování kryptografických klíčů. Je zde navázáno na práce [1] a [2].

Výsledkem práce je návrh a implementace PUFu na mikrokontroléru, sloužící k identifikaci zařízení a generování kryptografických klíčů a jeho analýza v různých teplotních podmínkách.

Klíčová slova Bezpečnost hardwaru, PUF, SRAM, výběr stabilních bitů, samoopravné kódy, identifikace zařízení, generování klíčů.

Abstract

This thesis deals with Physical Unclonable Functions (PUFs) and bit selection algorithms for PUF on cheap microcontroller.

First, we provide a literature research concerning PUFs in general with a focus on PUFs suitable for SRAMs. Then we introduce bit selection algorithm for PUF and we use it as cryptographic key generator. We extend the research of [1] and [2].

The result of this work is the implementation of SRAM PUF on microcontroller. The PUF can be used for device identification or as a cryptographic key generator. The proposed PUF is analysed and tested at varying temperatures.

Keywords Hardware security, PUF, SRAM, stable bit selection, error correction codes, device identification, key generation.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Motivace

Propojení a vzdálené ovládání elektronických zařízení pomocí tzv. Internetu věcí (Internet of Things - IoT) se dnes stává čím dál více rozšířeným. IoT umožňuje využívat tato zařízení například k elektronickým platbám, monitorování domácích spotřebičů či polohy dopravních prostředků. Vzhledem k široké škále aplikací, které často pracují s osobními daty uživatelů, je nutné zajistit adekvátní bezpečnostní prvky jako identifikaci a autentizaci zařízení nebo generování kryptografických klíčů.

Kryptografické klíče se většinou ukládají do nevolatilní paměti, která je náchylná na semi-invazivní[1] nebo invazivní[2] útoky, a kterou je drahé a složité zabezpečit proti těmto a jiným útokům.[4]

Efektivní a levné řešení nabízí například tzv. fyzicky neklonovatelné funkce (Physical Unclonable Function - PUF). PUF využívá jedinečných fyzikálních vlastností zařízení, které vznikají vlivem náhodných a nekontrolovatelných vlivů během výrobního procesu a může sloužit pro generování jednoznačného a dostatečně náhodného identifikátoru zařízení nebo i kryptografického klíče. PUF je považován za bezpečnou a nízkonákladovou technologii, jelikož identifikátor zařízení nebo kryptografický klíč nezůstává uložen v paměti po vypnutí zařízení. Navíc dokážeme jednoduše detekovat invazivní útoky na zařízení, protože jakákoliv malá fyzická úprava PUFu povede k výrazně odlišnému identifikátoru či klíči.

Díky svým vlastnostem představuje PUF jednoduché a levné řešení k zabezpečení nízkonákladových zařízení a proto dnes můžeme nalézt na trhu stále více komerčních produktů, které jej využívají, například Intrinsic-ID [5] nebo LPC5500 od firmy NXP [6].

[1] Druh útoku, který vyžaduje proniknutí ochranou zařízení, která však nebude trvale poškozena.[3]

[2] Druh útoku, který vyžaduje proniknutí a trvalé poškození ochrany zařízení.[3]

Cíle práce

V této práci se budeme zabývat návrhem a implementací fyzicky neklonovatelné funkce (PUF) na nízkonákladovém mikrokontroléru, konkrétně na přípravku STM32F030R8. Dále představíme různé dosavadní metody výběru bitů pro PUF a na jejich základě navrhneme vlastní metodu vyhodnocení PUF a provedeme její implemetaci. Navazujeme zde na práce [1] a [2]. Navíc se budeme věnovat i aplikaci vytvořeného PUFu ke generování kryptografických klíčů. Následně provedeme experimentální vyhodnocení efektivity a spolehlivosti navržené metody v různých teplotních podmínkách.

Členění práce

Práce je rozčleněna do deseti kapitol včetně úvodu a závěru. První kapitola obsahuje obecný popis PUFu. Věnujeme se jeho vlastnostem a aplikacím, mezi které patří identifikace zařízení, autentizace a generování kryptografických klíčů. Dále popisujeme různé, již existující konstrukce PUFů.

Druhá kapitola navazuje v rešerši na kapitolu předchozí, ale věnuje pouze SRAM PUFu, protože jej využíváme v praktické části práce. Mimo obecného popisu konstrukce SRAM PUF zde popisujeme i metriky, pomocí kterých lze analyzovat statistické vlastnosti SRAM PUFu.

Třetí kapitola se věnuje rešerši na téma samoopravných kódů, konkrétně kódů BCH a repetičního, jelikož je využíváme v praktické části práce k opravě získaných odpovědí PUFu.

Ve čtvrté kapitole je provedena analýza současných řešení metod výběru bitů pro PUF.

V páté kapitole se věnujeme samotnému návrhu našeho PUFu. Představujeme zde nový přístup výběru stabilních bitů odpovědi PUF. Dále zde popisujeme i aplikaci našeho návrhu PUFu ke generování kryptografických klíčů, tedy zpracování výstupu samoopravnými kódy, za účelem zvýšení stability.

Šestá kapitola popisuje realizaci provedených experimentů. Představujeme zde vlastnosti mikrokontrolérů STM32F030R8 a celkové zapojení přípravků a tepelné komory použité k implementaci SRAM PUF.

V sedmé kapitole prezentujeme výsledky implementace a následných měření, které jsou prováděny na deseti přípravcích STM32F030R8. Ukazujeme využití našeho návrhu PUFu pro identifikaci zařízení se SRAM pamětí a při použití se samoopravným kódem i pro generování stabilních kryptografických klíčů.

V poslední kapitole na základě výsledků z předchozí kapitoly prezentujeme aplikaci vytvořeného SRAM PUFu pro identifikaci jednotlivých přípravků a při použití samoopravného kódu i pro generování stabilních kryptografických klíčů.

Kapitola 1

Fyzicky neklonovatelné funkce

Tato kapitola představuje obecný koncept fyzicky neklonovatelné funkce (Physical Unclonable Function – PUF). Zaměříme se na popis vlastností PUFů, výčet typů PUFů a jejich základních aplikací.

1.1 Úvod do PUF

Fyzicky neklonovatelná funkce (PUF) je taková funkce, která využívá technologické jedinečnosti dané výrobou každého integrovaného obvodu ke generování nepředvídatelné odpovědi. Pojem funkce zde může být zavádějící, neboť se nejedná o funkci v matematickém slova smyslu, ale spíše o funkci v technickém pojetí, tedy proceduru provedenou na určitém zařízení.[7]

PUF tedy umožňuje jednoznačně identifikovat zařízení na základě jeho fyzikálních vlastností, které vznikají vlivem náhodných a nekontrolovatelných vlivů během výrobního procesu. Mezi tyto vlastnosti řadíme například zpoždění na obvodu, nebo preference hodnot paměťových buněk po zapnutí napájení.[8] Takto získaný PUF lze poté využít jak k identifikaci a autentizaci zařízení, tak i ke generování kryptografických klíčů, jelikož je náročné ho duplikovat nebo predikovat [1].

První koncept PUFu, vytvořený pomocí optického média, představil Pappu v roce 2001 v [9]. Ve své práci definoval tzv. fyzickou jednosměrnou funkci (*Physical One-Way Function*) jako funkci, která je snadno vypočitatelná, ale obtížně se k ní hledá inverze. V roce 2002 Gassend et al v [10] prezentoval tzv. fyzicky náhodnou funkci (*Physical Random Function*), která využívala zpoždění kruhových oscilátorů způsobené nekontrolovatelnými vlivy během výroby. Již v této práci ovšem nalezneme i označení PUF, které se nakonec obecně ujalo a používají ho rozdílné konstrukce a koncepty, které ale sdílejí řadu vlastností.[8]

Vstupem PUFu je tzv. výzva (*challenge*) a fyzikální stav zařízení a jeho výstupem tzv. odpověď (*response*). Tedy na základě této výzvy PUF generuje (náhodnou) odpověď, která se odvíjí od jeho jedinečných fyzikálních

vlastností. PUF odpovědi jsou citlivé na změnu externích podmínek, jako například změnu teploty nebo napájecího napětí. Tyto odchylky v odpovědích jsou nicméně dostatečně malé na to, abychom stále byli schopni jednoznačně identifikovat PUF na základě získané odpovědi.

Vlastnosti PUFu mohou být hodnoceny teoreticky pomocí matematických modelů [11] nebo experimentálně analýzou PUF implementací [12]. Jelikož ale matematické modely nemohou zahrnovat všechny reálné fyzikální podmínky a výsledky experimentů se často liší kvůli různým použitým testovacím metodám, je bezpečnost PUFu stále předmětem výzkumu.[13]

1.2 Vlastnosti PUF

Tato kapitola se věnuje přehledu vlastností, kterými by měl PUF disponovat. Vycházíme zde z vlastností, které uvedl Maes et al v [7] v roce 2012. Prvních šest označuje jako nezbytné, tedy jsou to takové vlastnosti, které PUF obecně definují. Zbylé vlastnosti jsou sice žádoucí, ale mohou být garantovány pouze pro určité konstrukce PUFu. Obdobně jsou tyto vlastnosti popsány i v [8] a [4].

Konstruovatelnost

Konstruovatelnost (*Constructibility*) je základní a nezbytná vlastnost PUFu. Vyžaduje, aby byl PUF realizovatelný v rámci fyzikálních zákonů. V praxi tato vlastnost znamená hlavně dostupnou cenu nákladů výroby PUFu.

Vyhodnotitelnost

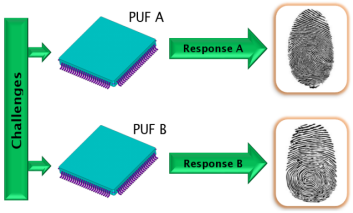
Vyhodnotitelnost (*Evaluability*) označuje vlastnost PUFu odpovědět na výzvu v polynomiálním čase a prostoru. V praxi ale ukazuje spíše na to, zda PUF vyhodnotí odpověď v zadaném a omezeném čase, ploše, napájení, spotřebě a ceně.

Reprodukovatelnost

Reprodukovatelnost (*Reproducibility*) PUFu značí, že s vysokou pravděpodobností dostaneme na jednom zařízení na stejnou výzvu pokaždé stejnou odpověď. Případná odchylka (způsobená různými fyzikálními vlivy) by měla být tak malá, aby se dala odstranit (například samoopravnými kódy). Tato vlastnost odlišuje PUF od náhodných generátorů čísel.[4]

Jedinečnost

Jedinečnost (*Uniqueness*) PUFu značí, že se vysokou pravděpodobností odpovědi na výzvu získané z různých zařízení budou lišit a to ideálně v 50%.[8] Obrázek 1.1 přirovnává jedinečnost PUFu k unikátnosti lidských otisků prstů.



Obrázek 1.1: Jedinečnost PUFu. Po aplikaci stejné výzvy (*Challenges*) na dva různé PUFy (*PUF A* a *PUF B*) získáme odlišné odpovědi (*Response A* $\neq$ *Response B*).[3]

Identifikovatelnost

Identifikovatelnost (*Identifiability*) PUFu představuje spojení dvou předchozích vlastností, reprodukovatelnosti a jedinečnosti. Vyjadřuje, že PUF lze identifikovat na základě jeho odpovědí. Odpovědi jednoho PUFu na stejnou výzvu jsou si tedy podobné, resp. se liší od odpovědí jiných PUFů.

Fyzická neklonovatelnost

Fyzická neklonovatelnost (*Physical Unclonability*) označuje hlavní princip PUF. Technicky je velice obtížné vyrobit dvě zcela identická zařízení, která by generovala dvě stejné PUF odpovědi. Je tedy jednoduché vytvořit PUF s náhodnými hodnotami, ale složité (tj. v zadaném čase, prostoru a ceně nereálné) vytvořit specifický PUF s danými hodnotami. Bezpečnostní výhodou PUFu je tedy, že máme jistotu, že orginální výrobce dodává všechna zařízení s unikátním PUFem. Tato vlastnost se označuje jako „odolnost proti pozměnění výrobcem“ (*manifacturer resistance*).

Nepředvídatelnost

Nepředvídatelnost (*Unpredictability*) PUFu zajišťuje, že ani v případě odposlechnutí omezené množiny dvojic výzev a odpovědí daného PUFu, potencionální útočník nedokáže zkonstruovat algoritmus pro získání odpovědi pro náhodnou výzvu tohoto PUFu.

Matematická neklonovatelnost

Matematická neklonovatelnost (*Mathematical Unclonability*) je rozšířením předchozí vlastnosti PUFu, nepředvídatelnosti. Zatímco v případě nepředvídatelnosti měl potencionální útočník pouze limitovaný přístup k (náhodným) dvojicím výzev a odpovědí PUFu, zde pracujeme s předpokladem, že útočník má fyzický přístup k PUFu a tedy může získat tolik párů výzev a odpovědí, kolik

[3] Převzato z [14].

je schopen uložit. Pokud ani v tomto případě neexistuje matematický aparát, který umožňuje vytvořit takový algoritmus, který dokáže predikovat odpověď PUF na zadanou výzvu, tvrdíme, že daný PUF je matematicky neklonovatelný. Nutnou podmínkou matematické neklonovatelnosti je tak velký počet párů výzev a odpovědí, aby nebylo možné je všechny uložit a vyhledávat v nich, například ve formě tabulky. Matematická neklonovatelnost PUFu tedy implikuje jeho nepředvídatelnost.

Jednosměrnost

Jednosměrnost (*One-wayness*) znamená, že na základě daného PUFu a jeho (náhodné) odpovědi nelze zkonstruovat příslušnou výzvu. Nutnou podmínkou jednosměrnosti je velký počet párů výzev a odpovědí, aby nebylo možné je všechny uložit a vyhledávat v nich, například ve formě tabulky.

Detekovatelnost manipulace

Detekovatelnost manipulace (*Tamper-evidence*) se týká fyzické manipulace útočníka se zařízením obsahujícím PUF za účelem modifikovat jeho chování. Tato vlastnost značí, že jakákoliv malá fyzická úprava PUFu povede k výrazně odlišným odpovědím.

1.3 Typy PUF

Od představení prvního PUFu Pappuem v roce 2002 bylo navrženo mnoho typů PUFu. V této kapitole popisujeme členění PUFů dle počtu párů výzev a odpovědí a jejich zabezpečení, dále dle typu použitého materiálu zdroje náhodných a unikátních odpovědí a nakonec dle umístění PUFu. Podrobněji se zaměřujeme hlavně na *Intrinsic Memory-based* PUFy, podobné SRAM PUFu, který využíváme v praktické části práce.

V závislosti na počtu párů výzev a odpovědí a jejich zabezpečení lze PUF klasifikovat jako silný (*Strong*), kontrolovaný (*Controlled*) nebo slabý (*Weak*).[14]

1. Silný PUF: Jako silný označujeme PUF s velkým množstvím párů výzev a odpovědí. Počet párů výzev a odpovědí roste exponenciálně v závislosti na počtu bitů výzvy. Zároveň platí, že páry výzev a odpovědí jsou veřejně přístupné a tedy není možné je všechny uložit a vyhledávat v nich, například formou tabulky. Vzhledem k velkému počtu párů a odpovědí je tento PUF vhodný k autentizaci zařízení. Příkladem silného PUFu je Arbiter PUF nebo Bistable Ring PUF.

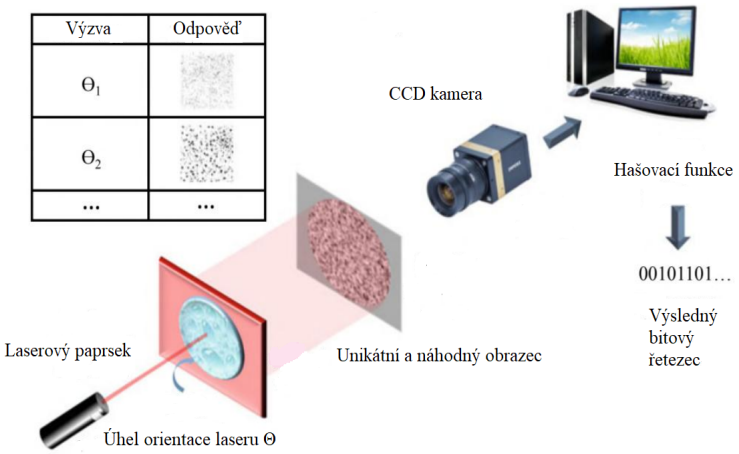
2. Kontrolovaný PUF: Jako kontrolovaný označujeme silný PUF, jehož páry výzev a odpovědí nejsou veřejně přístupné a jsou navíc zabezpečené například pomocí hashovací funkce (viz Gassend et al v [15]).

3. Slabý PUF: Jako slabý označujeme PUF s malým množstvím párů výzev a odpovědí (v extrémním případě pouze jedním párem), které nejsou veřejně přístupné. Vhodný ke generování kryptografických klíčů. Známý také jako *Physically Obfuscated Key*.[16] Příkladem slabého PUFu je SRAM PUF, který je založen na čtení obsahu paměti po zapnutí zařízení a kde výzvu může představovat adresa v paměti.[17]

Maes et al v [7] člení PUFy dle typu použitého materiálu zdroje náhodných a unikátních odpovědí na elektronické (*Electronic*) a neelektronické (*Non-electronic*).

1. Elektronické PUFy: Generují odpovědi na základě měření odchylek elektrických analogových signálů (například změn v napětí tranzistorů).[18] Mezi elektronické PUFy řadíme například SRAM PUF, Latch PUF, Butterfly PUF nebo Flip-flop PUF (podrobněji o nich pojednáváme níže).

2. Neelektronické PUFy: Zahrnují nejstarší konstrukce PUFu a často bývají založeny na optických efektech. Mezi neelektronické PUFy řadíme například Paper PUF představený v [19], CD PUF představený v [20] nebo Optical PUF představený v [9] (viz obrázek 1.2).



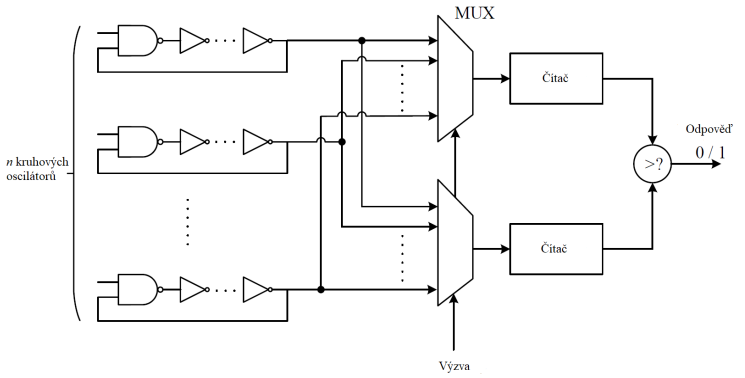
Obrázek 1.2: Princip Optical PUF spočívá v osvícení speciálního materiálu (naplněného velkým množstvím částic, které rozptylují světlo) helium-neonovým laserovým paprskem. Výsledkem tohoto procesu je unikátní a náhodný obrazec. PUF výzvu zde reprezentuje přesná pozice a úhel použitého laseru, zatímco PUF odpověď je výsledkem digitálního zpracování a kvantifikace (zakódování do bitového řetězce) promítnutého obrazce.[21].[4]

Guajardo et al v [23] dále člení PUFy dle umístění na vnitřní (*Intrinsic*) a vnější (*Non-intrinsic*). Vnitřní PUF je pevnou součástí integrovaného

[4] Převzato a upraveno z [22].

obvodu a jeho vyhodnocení probíhá interně uvnitř daného obvodu. Zdroj entropie tohoto PUFu vzniká díky nekontrolovatelným vlivům během výroby zařízení. Vnitřní PUFy je snadné zkonstruovat, jelikož není třeba vyrábět žádné speciální obvody navíc. Vnitřní PUFy dále dělíme dle zdroje náhodnosti na *Delay-based* a *Memory-Based*.[7]

1. Delay-based PUF: Zdrojem entropie jsou malé náhodné odchylky ve zpoždění hradel a jejich propojení, způsobené nekontrolovatelnými vlivy během výroby a vnějšími podmínkami. Příkladem Delay-based PUFu je například Arbiter PUF představený v [24] nebo Ring Oscillator PUF představený v [10] (viz obrázek 1.3).



Obrázek 1.3: Princip Ring Oscillator PUF spočívá v měření frekvencí oscilujících obvodů. PUF se skládá z n kruhových oscilátorů. Kruhový oscilátor (*Ring Oscillator*) je většinou složený z lichého počtu invertorů. PUF porovnává frekvenci dvou zvolených oscilátorů a v závislosti na tom, který z nich je rychlejší, generuje na výstup logickou 1 nebo 0. K získání PUF odpovědi ve formě bitového řetězce tedy porovnáváme dvojice kruhových oscilátorů z náhodně zvolené množiny (tato množina reprezentuje PUF výzvu).[21][5]

2. Memory-based PUF: Zdrojem entropie je nestabilita v bistabilních klopných obvodech, které tvoří paměťové buňky, způsobená nekontrolovatelnými vlivy během výroby a vnějšími podmínkami. Mezi Memory-based PUFy řadíme například SRAM PUF, Latch PUF, Butterfly PUF nebo Buskeeper PUF.

1.3.1 Latch PUF

Su et al v roce 2007 představil v [25] koncept *Memory-based* PUFu nazvaný Latch PUF. Latch PUF je tvořen bistabilním klopným obvodem (*latch*), složeným ze dvou křížově propojených logických hradel NOR.

[5] Převzato a upraveno z [14].

Obrázek 1.4a ilustruje princip Latch PUFu. Pokud je signál *Reset* nastaven na hodnotu logické 1, pak se Latch PUF nachází v nestabilním stavu, který poté konverguje ke stabilnímu stavu, jehož hodnota se odvíjí od vnitřních fyzikálních vlastností a odchylek mezi hradly NOR.[14]

Na rozdíl od paměťových buněk SRAM PUF, které se nacházejí v nestabilním stavu, dokud nejsou nastaveny, paměťové buňky Latch PUF naopak zůstávají ve stabilním stavu, dokud není nastaven signál *Reset*.[4] Dále Latch PUF není závislý na neinicializovaných hodnotách paměťových buněk po zapnutí zařízení a je tedy více flexibilní, protože dokáže generovat náhodné a unikátní odpovědi kdykoliv je signál *Reset* přiveden do hodnoty logické 1. Není tedy třeba si PUF odpověďi pamatovat, jelikož mohou být kdykoliv znovu vygenerovány.[8]

1.3.2 Butterfly PUF

Kumar et al v roce 2008 představil v [26] koncept *Memory-based* PUFu nazvaný Butterfly PUF. Butterfly PUF je tvořen dvojicí křížově propojených bistabilních klopných obvodů.

Obrázek 1.4b ilustruje princip Butterfly PUFu. Pokud jsou signály *Preset* a *Clear* nastaveny pomocí budícího signálu najednou na hodnotu logické 1, pak se Butterfly PUF nachází v nestabilním stavu. Po uplynutí určitého počtu hodinových cyklů jsou signály nastaveny na hodnotu logické 0 a ustálený obvod poté konverguje ke stabilnímu stavu, jehož hodnota se odvíjí od vnitřních fyzikálních vlastností a odchylek mezi klopnými obvody.[14]

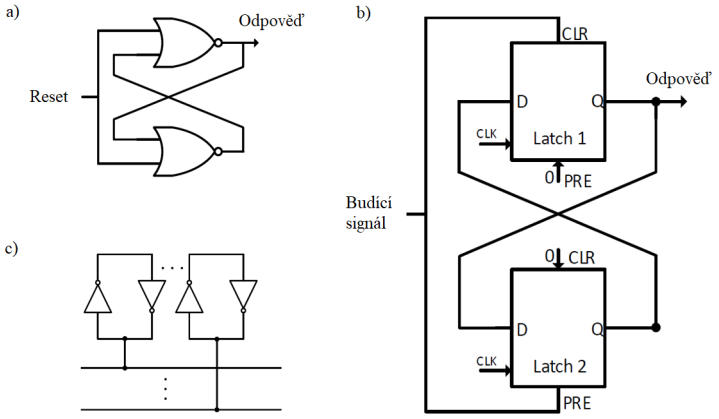
Butterfly PUF kompenzuje nevýhody implementace SRAM PUF na FPGA. Na některých FPGA totiž nelze SRAM PUF implementovat, jelikož buňky SRAM jsou po přivedení napájení automaticky inicializovány na hodnotu logické 0.[4] Stejně jako Latch PUF, ani Butterfly PUF nevyžaduje ke generování PUF odpovědí opakovaně zapínat zařízení.[7] Nevýhodou Butterfly PUFu je složitá konstrukce, protože pro správné fungování je nutné dosáhnout optimální symetrie propojení výstupů Q a vstupů D mezi oběma klopnými obvody a také budícím signálem.[8]

1.3.3 Buskeeper PUF

Simons et al v roce 2012 představil v [27] koncept *Memory-based* PUFu nazvaný Buskeeper PUF. Buskeeper PUF je tvořen tzv. *bus keeper* buňkami. Bus keeper buňka se skládá z dvojice křížově propojených invertorů napojených na sběrnici.

Obrázek 1.4c ilustruje princip Buskeeper PUFu. Bus keeper buňka v sobě uchovává poslední hodnotu sběrnice. Podobně jako v případě buněk SRAM, hodnoty bus keeper buněk se odvíjí od fyzikálních vlastností zařízení, které vznikají vlivem náhodných a nekontrolovatelných vlivů během výrobního pro-

cesu. Výhodou Buskeeper PUFu je malá velikost buskeeper buněk, v porovnání s například bistabilními klopnými obvody.[7]



Obrázek 1.4: Konstrukce *Memory-based* PUFů. a) Latch PUF, b) Butterfly PUF, c) Buskeeper PUF.[6]

1.4 Aplikace PUF

V této kapitole popisujeme tři základní aplikace PUFu, identifikaci a autentizaci zařízení a generování kryptografických klíčů.

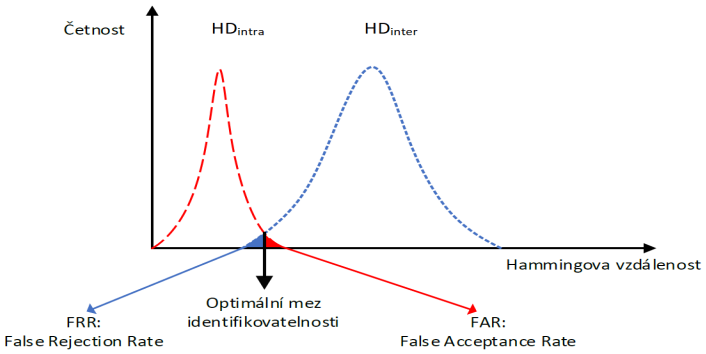
1.4.1 Identifikace zařízení

Identifikace zařízení je velice snadný způsob aplikace PUFu, protože nevyžaduje žádné další úpravy získaných PUF odpovědí. Proces identifikace dělíme na dvě fáze, *inicializaci* a samotnou *identifikaci*. Během inicializace opakovaně (náhodně) generujeme páry výzev a odpovědí PUFu a ukládáme je v databázi. Během identifikace je náhodně zvolena výzva uložená v databázi a pokud aktuální odpověď PUFu na tuto výzvu odpovídá odpovědi uložené v databázi, identifikace zařízení proběhla úspěšně. Po použití se daná výzva a odpověď z databáze vymažou, aby se předešlo útoku pomocí odposlechnutí dat.[7]

Vzhledem k poměrně vysoké chybovosti PUF odpovědí se toleruje určitá odchylka od uložených odpovědí v databázi. Porovnání získané odpovědi s odpovědí uloženou v databázi se provádí pomocí výpočtu Hammingovy vzdálenosti. Odpovědi z jednoho PUFu na opakovanou stejnou výzvu na jednom zařízení musí být velmi podobné, ale z různých zařízení dostatečně odlišné.[28] Tedy ideální průměrná Hammingova vzdálenost mezi odpověďmi PUFů různých zařízení je 50%.

[6]Převzato a upraveno z [14].

Na obrázku 1.5 vidíme, že maximální hodnota Hammingovy vzdálenosti pro určení úspěšné identifikace, se zjistí pomocí histogramu Hammingovy vzdálenosti mezi odpověďmi PUFu stejného zařízení (značíme jako HD_{intra}) a histogramu Hammingovy vzdálenosti odpovědí PUFů různých zařízení (značíme jako HD_{inter}). Pokud se tyto dva histogramy nepřekrývají, pak se mez identifikovatelnosti nachází v prostoru mezi oběma histogramy. V případě, že se ale histogramy částečně překrývají, pak se optimální mez identifikovatelnosti nachází na průsečíku těchto histogramů (tedy v oblasti minimálního chybného přijetí (*False Acceptance Rate*) i chybného odmítnutí (*False Rejection Rate*)).[8]



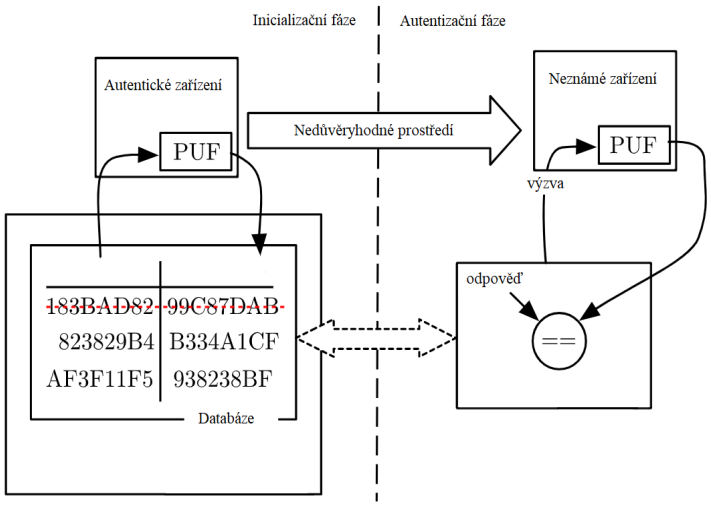
Obrázek 1.5: Princip nalezení optimální meze pro identifikaci zařízení.[7].

1.4.2 Autentizace zařízení

Proces autentizace pomocí PUFu spočívá v ověření pravosti identifikovaného zařízení. Schéma 1.6 ilustruje princip autentizace zařízení. Proces autentizace dělíme na dvě fáze, *inicializaci* a samotnou *autentizaci*. Během inicializace opakovaně (náhodně) generujeme páry výzev a odpovědí PUFu a ukládáme je v databázi spolu s identifikátorem konkrétního zařízení. Během autentizace se nejprve zařízení identifikuje pomocí identifikátoru. Poté je mu zaslána náhodně zvolená výzva a pokud zařízení na tuto výzvu vygeneruje odpověď, která je stejná jako ta uložená v databázi nebo pokud je rozdíl PUF odpovědí (vyjádřený pomocí Hammingovy vzdálenosti) menší než zvolená optimální mez, je zařízení úspěšně autentizováno.[17]

Po použití se daná výzva a odpověď z databáze vymažou, aby se předešlo útoku pomocí odposlechnutí dat.[7]

[7]Převzato a upraveno z [8].
[8]Převzato a upraveno z [29].



Obrázek 1.6: Princip autentizace zařízení.[8]

1.4.3 Generování kryptografických klíčů

PUF lze využít také k jednoduchému generování náhodných a nepředvídatelných kryptografických klíčů. Výhodou této metody je, že se klíče generují za běhu a není nutné je tedy ukládat do nevolatilní paměti, kterou je často složité a drahé zabezpečit. Na rozdíl od identifikace zařízení a autentizace je zde ale podmínkou zcela stabilní PUF odpověď. Pokud by získaný klíč nebyl při opakovaném čtení identický, pak by dešifrovaná data byla nečitelná. K dosažení identických PUF odpovědí nezávisle na počtu vyčítání lze využít například samoopravných kódů.[7] Existují ovšem i další metody získání stabilních odpovědí, např. Taniguchi et al využívá v [30] tzv. *Fuzzy Extractors*.

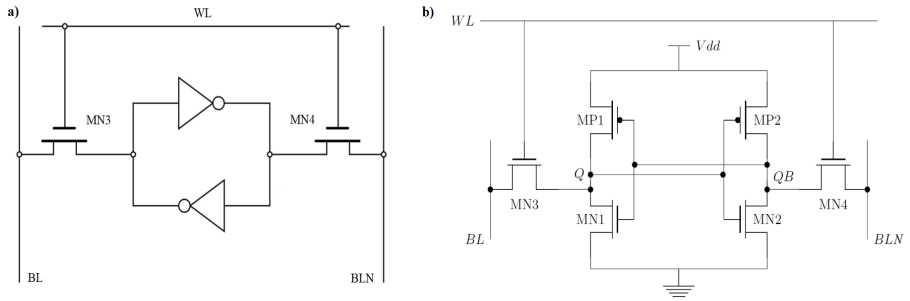
Proces generování kryptografických klíčů dělíme na dvě fáze, *inicializaci* a *rekonstrukci klíče*. Během inicializace vygenerujeme klíč pomocí PUFu a dále také pomocná data, která slouží k opravě PUF odpovědi během rekonstrukce klíče. Pomocná data obsahují informace potřebné pro samoopravný kód, případně konfiguraci PUFu.[8] Z pomocných dat nesmí být možné žádným způsobem získat samotný klíč, mohou být tedy zveřejněna. Inicializační fáze se provede pouze jednou. Během rekonstrukční fáze poté vygenerujeme klíč pomocí kombinace aktuální PUF odpovědi a pomocných dat z inicializační fáze procesu.

Kapitola 2

SRAM PUF

Guajardo a Holcomb et al v [12] a [23] navrhli v roce 2007 koncept PUFu, který využívá obsah polovodičové paměti SRAM (*Static Random-Access Memory*) po zapnutí napájení.

SRAM je paměť volatilní, udržuje tedy uložené hodnoty až do odpojení napájení. SRAM se skládá z dvourozměrné mřížky paměťových buněk schopných uložit jeden bit informace (logickou 0 nebo 1). Paměťové buňky jsou realizovány pomocí bistabilních klopných obvodů (viz 2.1a). V CMOS technologii se tento obvod typicky implementuje pomocí šesti tranzistorů typu MOSFET (viz 2.1b).

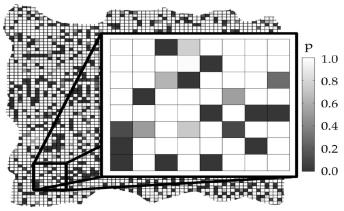


Obrázek 2.1: 6T SRAM buňka.[9] a) SRAM buňka složená ze dvou invertorů. b) Ekvivalentní obvod ze 6 tranzistorů. Dvojice křížově propojených invertorů (*MP1*, *MP2*, *MN1* a *MN2*) tvoří bistabilní klopný obvod, který dokáže přečíst, resp. zapsat jeden bit informace po příslušných datových vodičích *BL* a *BLN*. Přitom musí být otevřený jeden ze čtecích, resp. zapisovacích tranzistorů *MN3* nebo *MN4*. Otevření probíhá pomocí aktivace adresovacího vodiče *WL* pro výběr řádku, který je napojený na tranzistory *MN3* a *MN4*.[14]

[9] Převzato a upraveno z [31] a [14].

Každá paměťová buňka SRAM preferuje po zapnutí napájení určitou hodnotu, buď logickou 0 nebo 1, případně ani jednu z nich a chová se náhodně (viz obrázek 2.2).[7] Preference stavu jednotlivých buněk je náhodná a nemůže být ovlivněna, jelikož je způsobena asymetrií SRAM buněk, vzniklou během výrobního procesu.[17] Buňky, které uchovávají stále stejnou hodnotu považujeme za stabilní a využíváme je jako zdroj náhodnosti PUF. Různé metody výběru stabilních bitů SRAM popisujeme podrobněji v kapitole 4. Pokud nelze selektovat stabilní buňky SRAM ještě před generováním PUF odpovědí (například z důvodu vysoké ceny), pak je možné použít dodatečné metody na snížení chybovosti získaných PUF odpovědí, například korekci samoopravnými kódy.[4]

Odpověď SRAM PUFu tedy tvoří obsah zvoleného úseku paměti po zapnutí napájení, zatímco výzvu může reprezentovat například počáteční adresa daného úseku v paměti.



Obrázek 2.2: Rozložení stability SRAM buněk.[10]

Před použitím konkrétní SRAM ke generování PUFu je nutné zjistit, zda disponuje vlastnostmi nezbytnými pro PUF (uvedenými v kapitole 1.2). Kvalitu získaného PUFu lze analyzovat na základě různých metrik, například HD_{intra} nebo HD_{inter} (podrobněji v kapitole 2.1).[32]

Výhodou SRAM PUFu je snadná implementace, realizovatelná na většině dnešních mikrokontrolérů, bez nutnosti přidávat speciální hardware.[32] Citlivé údaje navíc nezůstávají uložena v zařízení, když je odpojeno od napájení a tím se minimalizuje potenciální bezpečnostní hrozba.[1]

Mezi nevýhody SRAM PUF řadíme vyšší míru chybovosti, například oproti dedikovaným PUF buňkám, které jsou speciálně navrženy tak, měly nejen minimální chybovost, ale i nízkou spotřebu a krátký čas čtení [33].

SRAM PUF představuje jednoduché a levné řešení k zabezpečení nízkonákladových zařízení a proto dnes můžeme nalézt na trhu stále více komerčních produktů, které jej využívají, například Intrinsic-ID [5] nebo LPC5500 od firmy NXP [6].

[10] Převzato z [12].

2.1 Analýza vlastností PUF

Dle [4] a [14] lze kvalitu SRAM PUFu ohodnotit na základě následujících veličin: náhodnosti, spolehlivosti, jednoznačnosti, entropie, pravděpodobnosti chybného přijetí a pravděpodobnosti chybného odmítnutí. V tabulce 2.1 uvádíme souhrn metrik kvality SRAM PUFu, včetně jejich ideálních hodnot.

Náhodnost

Vzhledem k požadované náhodnosti PUF odpovědí by se měl počet paměťových buněk, které mají hodnotu logické 1 rovnat těm, které mají hodnotu logické 0. Náhodnost lze vyhodnotit pomocí Hammingovy váhy následovně:

$$HW_i = \frac{1}{n}\sum_{j=1}^{n} r_{i,j} \times 100\%, \tag{2.1}$$

kde $r_{i,j}$ je j-tý bit v n-bitové PUF odpovědi zařízení i.

Ideální PUF by měl mít náhodnost rovnou 50%.[14]

Reprodukovatelnost

Odpovědi PUF by měli být za každých podmínek identické. Tedy odchylky způsobené například změnou teploty, napětí nebo stárnutím zařízení by měly být zanedbatelné. Bitovou chybovost lze vypočítat pomocí Hammingovy vzdálenosti. Tuto metodu označujeme také jako *Intra-Chip Hamming Distance* (značíme $\mathrm{HD}_{\mathrm{intra}}$). K porovnání lze jako referenční vektor použít například první PUF odpověď nebo majoritu z několika čtení PUF odpovědí.[34] Ideální PUF by měl mít $\mathrm{HD}_{\mathrm{intra}}$ rovno 0%.

[35] definuje $\mathrm{HD}_{\mathrm{intra}}$ následovně. Mějme PUF výzvu C pro zařízení i a n-bitovou PUF odpověď $R_i(n)$, vygenerovanou za určitých teplotních a napěťových podmínek. Aplikujeme-li stejnou PUF výzvu C na stejné zařízení i m-krát za jiných teplotních a napěťových podmínek, získáme n-bitovou odpověď $R'_i(n)$. Pak $\mathrm{HD}_{\mathrm{intra}}$ daného zařízení i je rovno:

$$HD_{\mathrm{intra}} = \frac{1}{m}\sum_{j=1}^{m} \frac{HD(R_i(n), R'_{i,j}(n))}{n} \times 100\% \tag{2.2}$$

Pomocí $\mathrm{HD}_{\mathrm{intra}}$ lze určit reprodukovatelnost PUFu jako:

$$Reprodukovatelnost = 100\% - HD_{intra} \tag{2.3}$$

Jedinečnost

Abychom mohli identifikovat zařízení, jeho PUF odpovědi musí být unikátní. Špatný návrh obvodu nebo chyba ve výrobě ale může zapříčinit tendenci paměťových buněk klonit se buď k hodnotě logické 0 nebo 1. Korelaci mezi

různými PUFy lze vypočítat pomocí Hammingovy vzdálenosti. Tuto metodu označujeme také jako *Inter-Chip Hamming Distance* (značíme $\mathrm{HD}_{\mathrm{inter}}$). Ideální PUF by měl mít $\mathrm{HD}_{\mathrm{inter}}$ rovno 50%.

Maiti et al. v [35] definuje $\mathrm{HD}_{\mathrm{inter}}$ následovně. Aplikujeme-li stejnou výzvu C na dvě různá zařízení i a j ($i \neq j$), získáme dvě n-bitové odpovědi $R_i(n)$ a $R_j(n)$. Pak $\mathrm{HD}_{\mathrm{inter}}$ mezi k zařízeními je rovno:

$$HD_{\mathrm{inter}} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i(n), R_j(n))}{n} \times 100\% \tag{2.4}$$

Entropie

PUF odpovědi by měly mít vysokou entropii, tedy jednotlivé paměťové buňky se nesmějí navzájem nijak ovlivňovat. Korelaci mezi jednotlivými bity lze dle [36] vypočítat pomocí autokorelační funkce:

$$C_a(\tau) = \sum_{i=0}^{N-1} (-1)^{a_i + a_{i+\tau}}, 0 \leq \tau \leq N-1 \tag{2.5}$$

kde $a = \{a\}_i$ je binární sekvence PUF odpovědi a N je počet paměťových buněk. $C_a(\tau) = 0$ ukazuje na nulovou korelaci mezi bity.

SRAM PUFy s vysokou korelací mezi jednotlivými paměťovými buňkami a tedy nízkou entropií, jsou náchylné na útoky hrubou silou.[4]

Pravděpodobnost chybného přijetí

Pravděpodobnost, že PUF A bude identifikován jako PUF B (PUF A $\neq$ PUF B) a PUF A generuje stejný výstup jako PUF B, označujeme jako tzv. pravděpodobnost chybného přijetí (*False Acceptance Rate*).[37] Pravděpodobnost chybného přijetí p_{mis} vypočítáme dle [14] jako:

$$p_{mis} = \sum_{i=0}^{\epsilon} \binom{n}{i} p_{inter}^{i} (1 - p_{inter})^{n-i} \tag{2.6}$$

kde p_{inter} označuje pravděpodobnost jedinečnosti bitu vypočítanou pomocí rovnice 2.4, jako $\frac{HD_{inter}}{100}$, n je počet bitů PUF odpovědi a ϵ je zvolená maximální mez Hammingovy vzdálenosti.[14]

Pravděpodobnost chybného odmítnutí

Pravděpodobnost, že korektní PUF nebude úspěšně identifikován (tedy PUF nedokáže znovu vygenerovat výstup identický s výstupem uloženým v databázi), označujeme jako tzv. pravděpodobnost chybného odmítnutí (*False*

Rejection Rate).[37] Pravděpodobnost chybného odmítnutí p_{reject} vypočítáme dle [14] jako:

$$p_{reject} = 1 - \sum_{i=0}^{\epsilon} \binom{n}{i} p_{intra}^{i} (1 - p_{intra})^{n-i} \tag{2.7}$$

kde p_{intra} označuje pravděpodobnost chybného bitu vypočítanou pomocí rovnice 2.2, jako $\frac{HD_{intra}}{100}$, n je počet bitů PUF odpovědi a ϵ je zvolená maximální mez Hammingovy vzdálenosti.[14]

Tabulka 2.1: Analýza vlastností SRAM PUF

Veličina	Značení	Ideální PUF
Hammingova váha	HW	50%
Chybovost	$\mathrm{HD}_{\mathrm{intra}}$	0%
Korelace mezi různými SRAM PUF	$\mathrm{HD}_{\mathrm{inter}}$	50%
Korelace mezi bity	$C_a(\tau)$	0%
Pravděpodobnost chybného přijetí	FAR	0%
Pravděpodobnost chybného odmítnutí	FRR	0%

Kapitola 3

Samoopravné kódy

Odpovědi SRAM PUF mají obecně vysokou chybovost, ať už z důvodu šumu ve vlastním obvodu, stárnutím zařízení nebo následkem proměnlivých externích podmínek jako například změnou teploty. Yu et al. v [38] uvádí, že chybovost obvodu v proměnlivých podmínkách může být až 25%.

V případě autentizace obvodu pomocí PUFu tato chybovost nevadí, jelikož tam předpokládáme určitou odchylku (tj. zajímá nás, zda tato odchylka, reprezentována Hammingovou vzdáleností, je menší než daná mez). Pokud ovšem chceme získané PUF odpovědi použít jako kryptografické klíče, je nutné je stabilizovat. To se obvykle provádí pomocí samoopravných kódů, které chybné bity ve výstupu PUFu opraví.

Vzhledem k vysokému počtu chybných bitů v PUF odpovědi je třeba dobře zvážit, který samoopravný kód je vhodný k použití. Je nutné brát v úvahu, že některé mikrokontroléry nemají prostředky na implemetaci složitých samoopravných kódů.[34]

Při opravě SRAM PUF odpovědí pomocí samoopravných kódů vytváříme pomocné data, která následně typicky ukládáme do nevolatilní paměti. Dle [4] je nevýhodou využití nevolatilní paměti vyšší cena zařízení (pokud je paměť součástí zařízení) nebo potencionální odhalení citlivých údajů (v případě externí paměti).

Podrobněji se zaměřujeme na BCH samoopravný kód (stejně jako [1]) a repetiční samoopravný kód (stejně jako [2]), protože je využíváme v praktické části práce k opravě PUF odpovědí pro generování kryptografických klíčů.

BCH kód

Binární BCH (*Bose-Chaudhuri-Hocquenghem*) řadíme mezi lineární[11] blokové cyklické[12] kódy. Dle [39] definujeme parametry BCH(n, k) následovně:

[11] Kódová slova tvoří lineární prostor.

[12] Lineární kód, kde cyklickým posuvem kteréhokoliv kódového slova získáme opět kódové slovo.

- $n|(2^m - 1)$ je délka kódového slova, kde m je přirozené číslo a $m > 1$.
- $k = n - r$ je počet informačních bitů v kódovém slově, kde $r \leq m \cdot (d_{min} - 1)$.
- $d_{min} \geq 2$ je tzv. zaručená vzdálenost.

BCH kód(n, k) dokáže objevit $(d_{min} - 1)$ chyb a opravit až $(\frac{d_{min}-1}{2})$ chyb. BCH kódy konstruujeme pomocí konečných těles.[39]

Nevýhodou BCH kódu je jeho velká výpočetní složitost, kvůli které není vhodný pro některé mikrokontroléry. Miller et al. v [40] uvádí, že použití pouze BCH kódu k opravě PUF je také neefektivní v případě vysoké chybovosti (Bit Error Rate - BER) PUF odpovědí. K získání jednoho stabilního bitu klíče je potřeba až 3.68 bitů PUF odpovědi v případě, že BER PUF odpovědi $\sim 6\%$ a až 26.7 bitů PUF odpovědi, pokud BER $\sim 15\%$.[41] Z toho důvodu se často použití BCH kódu kombinuje s dalšími metodami snížení chybovosti PUF odpovědí, jako například různými druhy preselekce (podrobněji uvádíme v kapitole 4).

Praktický postup pro opravu SRAM PUFu pomocí BCH(31,26) a BCH(31,21) navrhl Laban et al. v [1].

Repetiční kód

Repetiční kód RP(n, k, $\frac{n-1}{2}$) se skládá z n-bitového kódového slova liché délky, kde $k = 1$ je informační bit. Parametr $\frac{n-1}{2}$ udává, kolik bitů lze tímto kódem úspěšně opravit.[42]

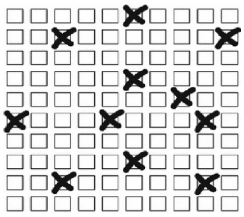
Jedná se o jednoduchý a efektivní samoopravný kód. Zkoumá majoritu (tj. početní převahu) sekvence bitů o liché délce, ze které následně generuje jeden bit výstupu. Aplikací repetičního kódu na SRAM PUF tedy n-krát zredukujeme velikost získaného výstupu za účelem odstranění chyb. Praktický postup pro opravu SRAM PUFu pomocí repetičního kódu navrhl například Bohm et al. v roce 2011. V [34] dosáhl při použití repetičního kódu délky 31 snížení BER odpovědí PUFu z 10% na $6.85 \cdot 10^{-7}\%$.

Výhodou repetičního kódu je jeho schopnost opravit velké počty chyb a jednoduchá implementace. Pomocí repetičního kódu lze opravit až 50% chyb. Naopak jeho nevýhodou je velká redundance a malý informační obsah. Navíc Koeberl et al v [42] zjistili, že repetiční kód snižuje entropii PUF až na 0%, pokud je použit na PUF, který má entropii nižší než 66%. Je proto nutné, aby měl PUF dostatečně vysokou entropii před použitím repetičního kódu.

Kapitola 4

Analýza současných řešení

Odpovědi SRAM PUF jsou typicky poměrně nestabilní a mají vysokou chybovost (Bit Error Rate – BER). Miller et al. v [40] uvádí, že tato chybovost může dosáhnout až 30%. Nicméně pokud bychom chtěli daný SRAM PUF využít například pro generování 128-bitového kryptografického klíče, je nutná chybovost menší než 10^{-15}.[40] Je tedy vhodné použít techniku, která tuto chybovost eliminuje nebo alespoň dostatečně sníží. Technika k výběru stabilních bitů SRAM PUF se označuje jako preselekce bitů. Obrázek 4.1 ilustruje princip výběru stabilních SRAM buněk.



Obrázek 4.1: Princip výběru stabilních buněk SRAM.[13]

Platonov v [4] představuje metodu ohodnocení kvality PUFu na základě stability bitů, kterou označuje jako výtěžnost (*efficiency*) a vyjadřuje ji jako:

$$\epsilon = \frac{počet\ stabilních\ buněk}{počet\ všech\ buněk} \tag{4.1}$$

Díky použití preselekce bitů v inicializační fázi generování PUF odpovědí snížíme nutnost použití dalších komplexních technik (například robustního samoopravného kódu) během rekonstrukce PUF odpovědí. Preselekce stabilních bitů se tedy použije pouze jednou v inicializační fázi a během rekonstrukce PUF odpovědí už nejsou třeba žádné další opakované výpočty.

[13] Převzato ze [4].

Preselekci bitů lze provést buď fyzickou modifikací použitého obvodu nebo aplikací speciálního algoritmu na získané PUF odpovědi. Metody, které modifikují obvod nalezneme například v [33] (*PUF-parallelization* a *PUF-biasing*) nebo v [43] (*PUF-hardening*). V této kapitole se dále zabýváme pouze metodami, které využívají speciální algoritmus k získání PUF odpovědí, tak jako metoda vytvořená v praktické části práce.

Níže představujeme výčet několika nejznámějších metod preselekce stabilních bitů dle [41].

Přímá preselekce

Opakovaně vyčítáme odpovědi PUF v různých teplotních a napěťových podmínkách a nalezené nestabilní bity poté odstraníme (tedy maskujeme) z dalších PUF odpovědí. Nevýhodou přímé preselekce je časová náročnost a malá spolehlivost. Spolehlivost této metody lze zlepšit zvýšením počtu vyčítaných odpovědí.[33] Přímé preslekce využívá například [44].

Nepřímá preselekce

Během nepřímé preselekce je každá jednotlivá buňka PUFu podrobena testu, který nepřímo indikuje, zda je či není stabilní. Buňka, která je označena jako stabilní je poté použita ke generování PUF odpovědí. Výhoda této metody spočívá v kratším čase testování, jelikož není nutné měnit během testování teplotní či napěťové podmínky a dále ve vyšším počtu odhalených nestabilních buněk (oproti přímé preselekci). Nevýhodou nepřímé preselekce je, že může označit některé stabilní buňky jako nestabilní a tak snížit počet buněk využívaných k tvorbě PUFu.

[25] představil nepřímou preselekci stabilních buněk na základě toho, jak rychle se ustálí jejich hodnota po zapnutí napájení. Obrázek 4.2 ilustruje, že jako stabilní jsou označeny pouze ty buňky, jejichž hodnota se ustálí dříve než v čase t_{useful} a které leží nad horní hranicí TH_+ nebo pod spodní hranicí TH_-.

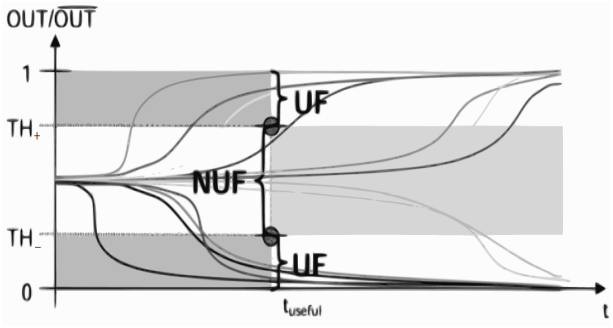
Nepřímé preselekce dále využívají například [40] nebo [41].

Volba na základě majority

Volba na základě majority (*Temporal Majority Voting*) spočívá v opakovaném čtení hodnoty jednotlivých buněk PUFu. Výsledná hodnota dané buňky je poté zvolena na základě majority získaných hodnot (tj. zda převládala během vyčítání logická 0 nebo 1). Nevýhodou volby na základě majority je časová náročnost a malá spolehlivost, pokud se změní teplotní nebo napěťové podmínky. Volby na základě majority využívá například [43].

Koeberl et al. v [45] představil velice podobnou metodu nazvanou *Spatial Majority Voting*, která spočívá v seskupení jednotlivých bitů PUF odpovědi

[14] Převzato a upraveno z [4].



Obrázek 4.2: Nepřímá preselekce. Výběr na základě rychlosti ustálení hodnoty buňky po zapnutí napájení. (UF: stabilní buňky, NUF: nestabilní buňky).[14]

do skupin. Z každé této skupiny se následně volí jeden výstupní bit na základě majority hodnot uvnitř skupiny. Koeberl et al. v [45] uvádí, že se takto podařilo dosáhnout snížení chybovosti PUF odpovědí z 6.5% na 0.3%.

V následující kapitole podrobně popisujeme použitou metodu preselekce v práci [1], ze které vycházíme v praktické části práce. Platonov et al. v [2] ve své práci ke tvorbě SRAM PUF preselekci nevyužívá a proto ho zde dále nezmiňujeme.

4.1 Preselekce dle počáteční hodnoty bitů

Laban et al. v [1] vytvořil vlastní metodu přímé preselekce stabilních bitů pro SRAM PUF speciálně pro 32-bitový mikrokontrolér STM32F050K6 s jádrem ARM Cortex-M0.

Laban et al. v [1] popisuje implementaci preselekce stabilních bitů následovně. Nejprve si vyhradíme 64 bytů (512 bitů) SRAM, ze kterých budeme generovat PUF. K výběru stabilních buněk budeme potřebovat opakovaně vyčítat obsah SRAM po zapnutí napájení. Jelikož je nepraktické mnohokrát ručně vypínat a zapínat zařízení, využíváme zde *standby mód*, který dokáže uvést paměťové buňky do neinicializovaného stavu. K probuzení přípravku ze standby módu využíváme RTC alarm.

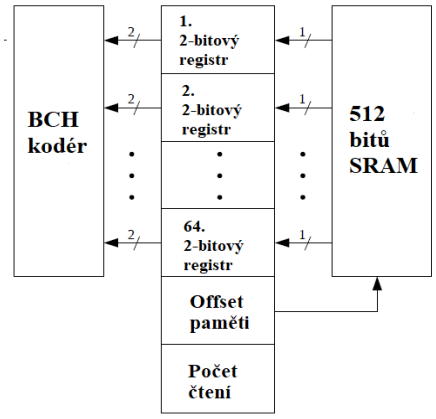
Použitý mikrokontrolér STM32F050K6 poskytuje pět 32-bitových záložních (*backup*) registrů, jejichž obsah se po vstupu zařízení do standby módu nesmaže. Tyto registry využíváme k rozlišení stabilních a nestabilních bitů paměti. První čtyři 32-bitové registry si rozdělíme do 64 menších 2-bitových registrů. Každý z těchto registrů pak obsahuje jednu z následujících informaci o jednom bitu paměťové buňky:

- ’00’b - tuto informaci nevyužíváme

- ’01’b - buňka inicializována na hodnotu logické 0
- ’10’b - buňka inicializována na hodnotu logické 1
- ’11’b - buňka změnila svoji hodnotu oproti své počáteční hodnotě

Do zbylých dvou 32-bitových záložních registrů ukládáme informaci o počtu opakování čtení a offsetu paměti (jelikož čteme 512 bitů paměti po 64 bitových úsecích, těchto offsetů bude celkem 8).

Obrázek 4.3 ilustruje postup preselekce stabilních bitů. Nejprve tedy přečteme prvních 64 bitů SRAM a inicializujeme záložní registry podle získaných hodnot (buď na ’01’b nebo ’10’b). Následně uvedeme zařízení do standby módu a tím znovu inicializujeme obsah paměti. Po probuzení zařízení ze standby módu opakujeme čtení stejných 64 bitů paměti, ale tentokrát porovnáváme přečtené hodnoty s těmi uloženými v záložních registrech. Pokud se hodnota bitu liší od inicializační hodnoty, změníme hodnotu záložního registru na ’11’b. Poté opět uvedeme zařízení do standby módu až dokud nedosáhneme počtu opakování čtení, uloženém ve čtvrtém záložním registru. Následně se posuneme v paměti dál o 64 bitů a celý proces opakujeme (s aktualizovaným offsetem paměti v pátém záložním registru) dokud nezpracujeme všech 512 bitů SRAM. Z bitů, které jsme označili jako stabilní, dále vytváříme kódová slova v BCH kódu.



Obrázek 4.3: Schéma preselekce stabilních bitů pro SRAM PUF dle počáteční hodnoty bitů.[15]

[15] Převzato a upraveno z [1].

Kapitola 5

Návrh PUF

V této kapitole představujeme návrh metody výběru stabilních bitů pro SRAM PUF mikrokontroléru STM32F030R8, skládající se ze dvou částí, preselekce bitů průměrnou stabilitou a preselekce bitů teplotní maskou. Dále popisujeme aplikaci tohoto PUFu ke generování kryptografického klíče, tedy zpracováváme výstup buď BCH nebo repetičním samoopravným kódem.

5.1 Preselekce dle průměrné stability

Tato část je inspirována [1] a [17]. Nejprve 1000krát vyčítáme obsah neinicializované SRAM po zapnutí napájení externím skriptem (viz schéma 5.1). Počet vyčítaných bitů SRAM závisí na následné aplikaci PUFu. V případě identifikace zařízení čteme 60000 bitů (tj. 92% velikosti SRAM, zbylých 8% zabírá samotný program). V případě generování 128-bitového kryptografického klíče čteme 4096 bitů SRAM (pokud rekonstruujeme klíč pomocí repetičního kódu) nebo 512 bitů SRAM (pokud rekonstruujeme klíč pomocí BCH kódu).

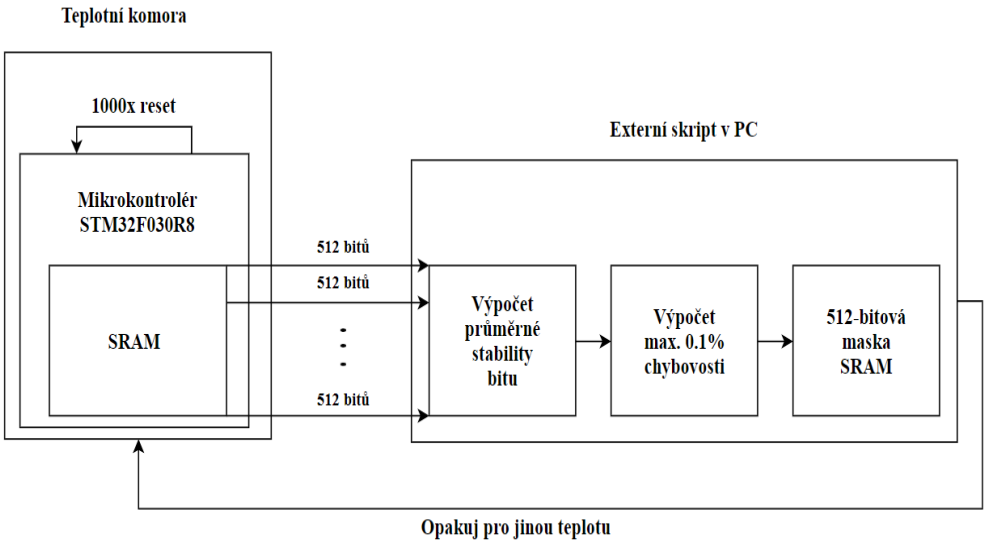
Jelikož je nepraktické mnohokrát ručně vypínat a zapínat zařízení, využíváme zde *standby mód*, který dokáže uvést paměťové buňky do neinicializovaného stavu. K probuzení přípravku ze standby módu využíváme RTC alarm.

Poté, co 1000krát přečteme obsah SRAM, vyhodnotíme pomocí externího skriptu stabilitu každé jednotlivé paměťové buňky. Průměrnou stabilitu i-tého bitu SRAM spočteme vztahem:

$$s_i = \frac{1}{N} \sum_{j=1}^{n} M_j, \tag{5.1}$$

kde n je počet paměťových buněk, N je počet čtení paměti a M_j značí j-tou paměťovou buňku SRAM.

Na základě získané průměrné stability jednotlivých SRAM buněk poté vygenerujeme bitovou masku SRAM tak, že jako logickou 1 označíme paměťovou buňku, která má maximálně 0.1% chybovost a naopak jako logickou 0 označíme



Obrázek 5.1: Preselekce 512 bitů SRAM dle průměrné stability bitů ve více teplotních podmínkách.

buňku, která má větší než 0.1% chybovost. Tato bitová maska SRAM nám tedy udává, které bity jsme označili jako stabilní a budeme s nimi pracovat v dalších krocích algoritmu.

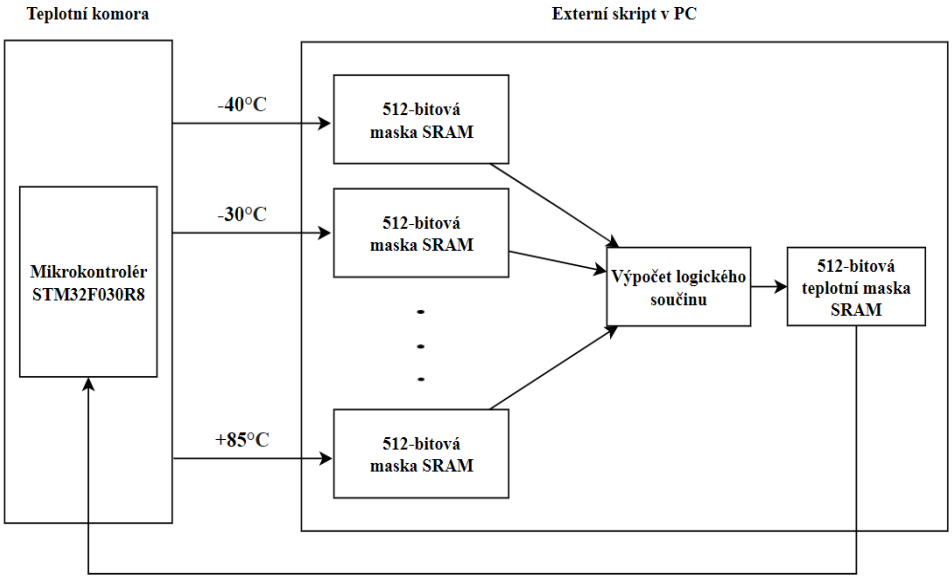
Toto měření provádíme identicky pro třináct zvolených teplotních úseků (v rozsahu od -40 °C do +85 °C). Výsledkem této operace je tedy třináct bitových masek SRAM pro jedno zařízení.

5.2 Preselekce dle teplotní masky

Abychom dosáhli stabilních PUF odpovědí ve všech zvolených teplotních úsecích (v rozsahu od -40 °C do +85 °C), provedeme navíc ještě druhou selekci stabilních bitů před vytvořením kódových slov zvoleného samoopravného kódu.

Vytvoříme teplotní masku SRAM tak, že jako stabilní (tedy jako logickou 1) zde označíme pouze tu paměťovou buňku, která byla vyhodnocena jako stabilní ve všech zadaných teplotních podmínkách. Tedy provedeme operaci logického součinu nad všemi třinácti bitovými maskami SRAM pomocí externího skriptu (viz schéma 5.2).

Výsledkem této operace je jedna bitová maska SRAM, určující stabilní bity nezávislé na externí teplotě. Tuto masku poté přikládáme k programu mikrokontroléru ve formě pole konstant.



Obrázek 5.2: Schéma vytvoření teplotní masky pro 512 bitů SRAM ve třinácti teplotních úsecích.

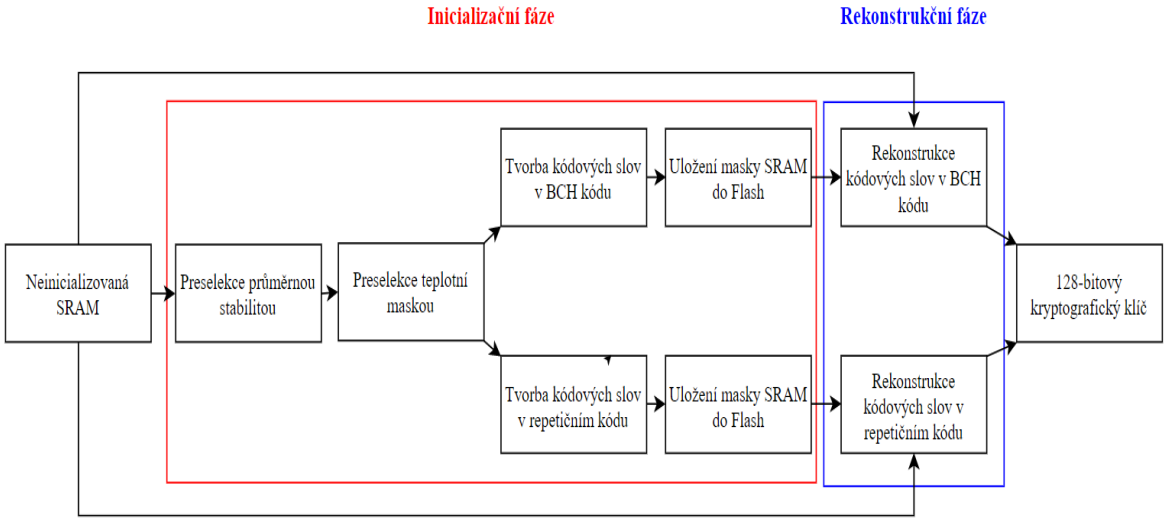
5.3 Generování kryptografických klíčů

Proces generování kryptografického klíče dělíme na dvě fáze, inicializaci a rekonstrukci klíče. Během inicializace vygenerujeme klíč pomocí SRAM PUFu a masky SRAM získané předchozí preselekcí a vytvoříme z něj kódová slova v příslušném samoopravném kódu (zvolili jsme BCH dle [1] nebo repetiční dle [2]). Kódová slova následně uložíme do paměti Flash v podobě masky SRAM. Inicializační fáze se provede pouze jednou. Během rekonstrukční fáze poté vygenerujeme klíč pomocí kombinace aktuální SRAM PUF odpovědi a masky SRAM uložené ve Flash paměti během inicializační fáze. Schéma 5.3 ilustruje celý proces generování klíče.

Oprava SRAM PUF BCH kódem

Tato část je inspirována [1]. Jako další opravný mechanismus PUFu jsme zvolili samoopravný kód BCH(31,26). Tento kód dokáže opravit jednu chybu v odpovědi PUF. Cílem je vytvořit 128-bitový kryptografický klíč generovaný pouze z 512 bitů SRAM, jehož spolehlivost je nezávislá na externích teplotních podmínkách.

Nejprve porovnáme aktuální hodnoty buněk SRAM s hodnotami teplotní masky získané během preselekce. Vytvoříme si pomocný bitový vektor hodnot SRAM pouze z těch buněk, které jsou dle teplotní masky stabilní (neboli je na jejich pozicích hodnota teplotní masky logická 1). Velikost tohoto vektoru



Obrázek 5.3: Schéma generování kryptografického klíče.

závisí na počtu stabilních buněk SRAM (neboli na počtu jedniček teplotní masky) a tedy se mezi různými zařízeními může lišit.

Následně vytvoříme informační část kódového slova ze souvislé sekvence 26 stabilních bitů. Poté hledáme zbylých pět opravných bitů kódového slova mezi dalšími stabilními bity SRAM. Stabilní bity, které byly přeskočeny během hledání těchto pěti bitů vyřadíme z finální masky SRAM uložené do FLASH paměti. Poté, co nalezneme mezi stabilními bity SRAM všechny potřebné bity pro tvorbu jednoho kódového slova, algoritmus opakujeme dokud nedojdeme na konec úseku SRAM ze kterého tvoříme PUF.

Počet kódových slov závisí na celkovém počtu stabilních bitů dané SRAM a také na tom, kolik bitů spotřebujeme na nalezení pěti opravných bitů každého kódového slova, a proto není předem znám a může se lišit mezi různými zařízeními.

Obrázek 5.4 ilustruje příklad inicializační fáze jednoduššího kódu BCH(7,4).

Během rekonstrukční fáze nejprve porovnáme aktuální hodnoty buněk SRAM s hodnotami masky SRAM uložené ve Flash paměti během inicializace. Vytvoříme si pomocný bitový vektor hodnot SRAM pouze z těch buněk, které dle masky SRAM tvoří kódová slova (neboli je na jejich pozicích hodnota masky SRAM logická 1) a tato slova následně dekódujeme. Algoritmus je schopen opravit jednu chybu v informačních bitech kódového slova.

Obrázek 5.5 ilustruje příklad rekonstrukční fáze jednoduššího kódu BCH(7,4).

[16]Převzato a upraveno z [1].
[17]Převzato a upraveno z [1].

	Inicializační fáze															
Stabilita	U	**S**	**S**	U	**S**	U	**S**	S	S	U	U	S	U	S	S	S
SRAM hodnota	0	**1**	**0**	1	**0**	1	**1**	0	1	1	1	1	0	1	1	0
Teplotní maska	0	**1**	**1**	0	**1**	0	**1**	1	1	0	0	1	0	1	1	1
Kódové slovo	x	**1**	**0**	x	**0**	x	**1**	x	1	x	x	1	x	x	x	0
Maska ve FLASH	0	**1**	**1**	0	**1**	0	**1**	0	1	0	0	1	0	0	0	1

Obrázek 5.4: Příklad inicializační fáze jednoduššího kódu BCH(7,4). Kódové slovo se skládá ze čtyř informačních bitů ('1001'b) a tří opravných bitů ('110'b). Informační bity jsou vytvořeny ze souvislé sekvence prvních čtyř stabilních bitů a tři opravné bity jsou dopočítány tak, aby tvořily kódové slovo.[16]

	Rekonstrukční fáze															
SRAM hodnota	1	**1**	**0**	1	**1**	1	**1**	0	1	1	0	1	0	1	0	0
Maska ve FLASH	0	**1**	**1**	0	**1**	0	**1**	0	1	0	0	1	0	0	0	1
Kódové slovo	x	**1**	**0**	x	**1**	x	**1**	x	1	x	x	1	x	x	x	0
Krypt. klíč	x	**1**	**0**	x	**0**	x	**1**	x	x	x	x	x	x	x	x	x

Obrázek 5.5: Příklad rekonstrukční fáze jednoduššího kódu BCH(7,4). Získané chybné kódové slovo ('1011110'b) je opraveno a čtyři bity kryptografického klíče ('1011'b) jsou vygenerovány.[17]

Oprava SRAM PUF repetičním kódem

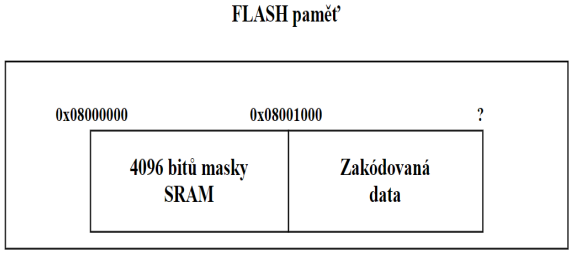
Tato část je inspirována [1] a [2]. Jako alternativní opravný mechanismus PUFu jsme zvolili repetiční samoopravný kód RP(11,1,5). Cílem je vytvořit 128-bitový kryptografický klíč generovaný pouze ze 4096 bitů SRAM, jehož spolehlivost je nezávislá na externích teplotních podmínkách. Velikost paměti, ze které generujeme PUF (4096 bitů) byla zvolena na základě experimentálních výsledků tak, aby bylo zajištěno generování dostatečně dlouhého klíče, pomocí co nejmenšího počtu bitů SRAM (stejně jako v [1]).

Nejprve porovnáme aktuální hodnoty buněk SRAM s hodnotami teplotní masky získané během preselekce. Vytvoříme si pomocný bitový vektor hodnot SRAM pouze z těch buněk, které jsou dle teplotní masky stabilní (neboli je na jejich pozicích hodnota teplotní masky logická 1). Velikost tohoto vektoru závisí na počtu stabilních buněk SRAM (neboli na počtu jedniček teplotní masky) a tedy se mezi různými zařízeními může lišit.

Následně pomocný bitový vektor rozdělíme do úseků o velikosti délky repetičního kódu (tedy délky 11 bitů). Hodnotu prvního bitu každé této podsekvence poté využijeme k vytvoření pomocných dat tak, že provedeme operaci exkluzivní disjunkce všech bitů v daném úseku s touto první hodnotou (viz obrázek 5.7). Tato pomocná data následně uložíme do paměti Flash. Navíc

ale musíme uložit do Flash také informaci o tom, které bity jsme ke tvorbě kódových slov využili, tedy teplotní masku stabilních bitů SRAM, získanou předchozí preselekcí (viz schéma 5.6).

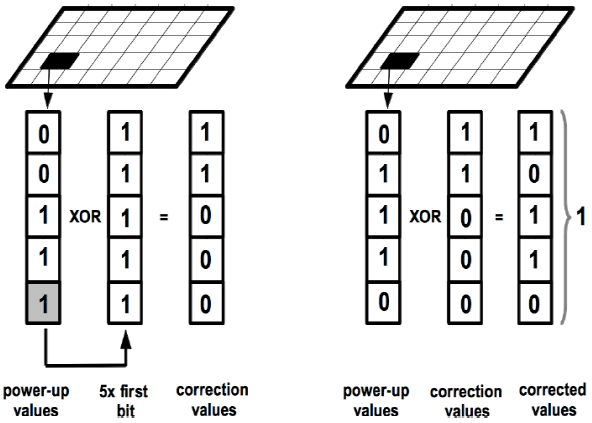
Počet kódových slov závisí na celkovém počtu stabilních bitů dané SRAM a proto není předem znám a může se lišit mezi různými zařízeními.



Obrázek 5.6: Schéma uložení pomocných dat repetičního kódu do paměti Flash. Délka zakódovaných dat závisí na počtu stabilních bitů dané SRAM.

Během rekonstrukční fáze nejprve porovnáme aktuální hodnoty buněk SRAM s hodnotami masky SRAM uložené ve Flash paměti během inicializace. Vytvoříme si pomocný bitový vektor hodnot SRAM pouze z těch buněk, které jsou dle masky SRAM stabilní (neboli je na jejich pozicích hodnota masky SRAM logická 1). Získaný binární vektor poté rozdělíme do úseků o velikosti délky repetičního kódu (tedy délky 11 bitů) a provedeme operaci exkluzivní disjunkce mezi binárním vektorem a zakódovanými pomocnými daty uloženými v paměti Flash. Na základě majority (převažující hodnoty v daném úseku) poté rozhodneme, zda původní hodnota buňky SRAM, přečtená v inicializační fázi (a tedy i výsledný jeden bit kryptografického klíče) byla logická 0 nebo 1. Algoritmus je schopen generovat jeden výsledný bit správně, pokud je méně než 6 bitů v daném úseku chybných.

[18] Převzato a upraveno z [4].



Obrázek 5.7: Inicializační fáze (vlevo) a rekonstrukční fáze (vpravo) repetičního kódu délky 5.[18]

Kapitola 6

Realizace

K realizaci návrhu SRAM PUFu používáme nízkonákladový mikrokontrolér STM32F030R8 s jádrem ARM Cortex-M0 na vývojové desce Nucleo-F030R8. Tento 32-bitový mikrokontrolér obsahuje 8 kB SRAM a 64 kB Flash paměti a pracuje na hodinovém kmitočtu 48 MHz. Podporuje napájecí napětí do 3.6 V. Umožňuje tři režimy běhu za nízké spotřeby (*sleep*, *stop* a *standby*). K probuzení mikrokontroléru z těchto režimů lze využít RTC alarm. Minimální provozní teplota je -40 °C a maximální provozní teplota je +85 °C.[46]

K testování návrhu SRAM PUFu používáme celkem deset mikrokontrolérů, které měříme ve třinácti různých teplotní úsecích (v rozsahu od -40 °C do +85 °C) pomocí teplotní komory Binder MK-56obj (viz obrázek 6.1).

Návrh SRAM PUF implementujeme v programovacím jazyce C a využíváme vývojového prostředí System Workbench for STM32. Celkem jsme vytvořili dva programy. První program slouží k opakovanému vyčtení co největšího obsahu neinicializované SRAM, zatímco druhý program slouží k inicializaci a následné opakované rekonstrukci kryptografického klíče. Program zabírá 512 bitů paměti Flash v případě generování 128-bitového klíče rekonstruovaného pomocí BCH samoopravného kódu a 4096 bitů paměti Flash v případě repetičního samoopravného kódu.

Po nahrání programu do přípravku je nutné odpojit a znovu zapojit mikrokontrolér od PC (připojení k ladícímu programu vývojového prostředí zabraňuje vstupu zařízení do standby módu).

Celkem provádíme tři základní měření. První měření spočívá v opakovaném čtení obsahu neinicializované SRAM pomocí uvedení přípravku do standby módu. Během druhého a třetího měření opakovaně generujeme 128-bitový kryptografický klíč, rekonstruovaný pomocí BCH nebo repetičního samoopravného kódu. Všechna měření provádíme ve zvolených třinácti teplotách.

Měření probíhá tak, že daný mikrokontrolér, umístěný v teplotní komoře, posílá data pomocí sériového rozhraní do PC. V PC je spuštěn skript, který vyčítá příchozí data a ukládá je do souborů. V momentě, kdy skript přečetl předem určené množství dat, změní nastavení teploty teplotní komory a pro-

ces opakuje pro všechny zvolené teploty. Skript je napsán v jazyce Python a využívá knihovnu k ovládání teplotní komory vytvořenou [47]. Skript komunikuje s teplotní komorou pomocí ethernetového rozhraní.

Obrázek 6.1: Teplotní komora Binder MK-56obj.[19]

[19]Převzato z [48].

Kapitola 7

Experimentální výsledky

V této kapitole prezentujeme výsledky testování navrhnuté metody výběru stabilních bitů pro SRAM PUF mikrokontroléru STM32F030R8, skládající se ze dvou částí, preselekce bitů průměrnou stabilitou a preselekce bitů teplotní maskou. Zabýváme se analýzou vytvořeného SRAM PUFu dle metrik prezentovaných v kapitole 2.1.

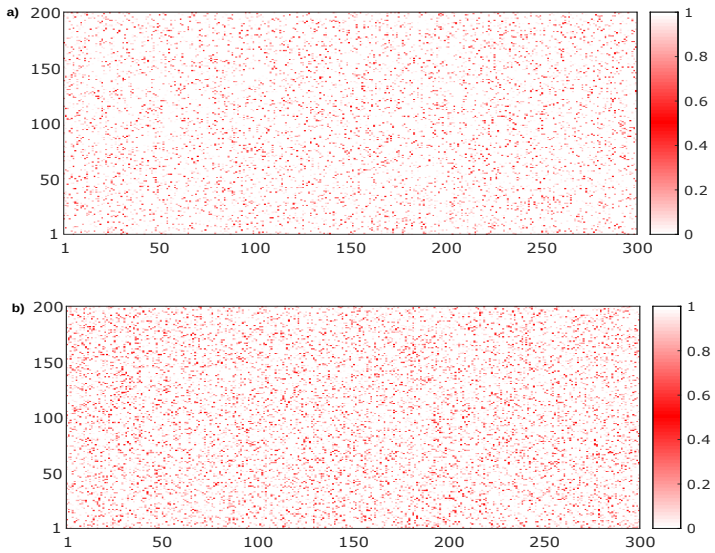
7.1 Preselekce dle průměrné stability

Testování preselekce stabilních bitů pro SRAM PUF dle průměrné stability probíhá následovně. Nejprve 1000krát přečteme 60000 bitů neinicializované SRAM (tj. 92% velikosti SRAM, zbylých 8% zabírá samotný program) deseti přípravků STM32F030R8 ve třinácti různých teplotních podmínkách. Jednotlivé přípravky dále označujeme jako MCU (*Microcontroller Unit*) 1 až 10. Rozsah měřených teplot byl zvolen dle minimální a maximální tolerované provozní teplotě mikrokontroléru STM32F030R8 (tedy od -40 °C do +85 °C).

Pro každý bit spočítáme jeho průměrnou stabilitu pomocí rovnice 5.1. Hodnota průměrné stability 0 a 1 značí zcela stabilní bit, zatímco hodnota 0.5 naopak označuje nejvíce nestabilní bit. Teplotní mapa 7.1 paměti přípravku MCU 1 ilustruje poměr stabilních a nestabilních bitů. Dále pozorujeme, že při vyšší teplotě (+85 °C) přípravek MCU 1 produkuje více nestabilních bitů než při nižší teplotě (-40 °C).

Na základě takto získané průměrné stability jednotlivých bitů poté generujeme bitovou masku SRAM pro každý z deseti přípravků následovně. Nejprve zvolíme procento tolerované chybovosti ve stabilitě buněk použitých pro SRAM PUF. Testovali jsme čtyři hodnoty chybovosti: 10%, 5%, 1% a 0.1%. Následně jako logickou 1 označíme paměťovou buňku, která má maximálně tuto zvolenou chybovost (10%, 5%, 1% nebo 0.1%) a naopak jako logickou 0 označíme buňku, která má chybovost větší.

Graf 7.2 ilustruje, jak se snižuje počet stabilních bitů tvořících bitovou masku paměti, čím více ořezáváme bity nestabilní.

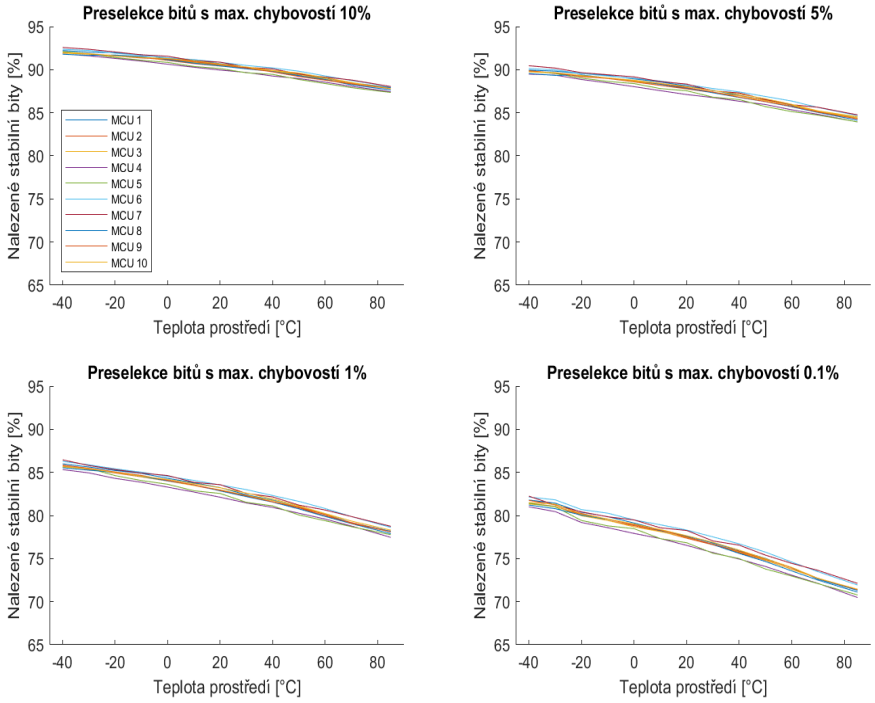


Obrázek 7.1: Poměr stabilních a nestabilních bitů SRAM přípravku MCU 1. Každý pixel reprezentuje jednu paměťovou buňku. Hodnoty 0 a 1 značí zcela stabilní bit (bílá barva), zatímco hodnota 0.5 označuje nejvíce nestabilní bit (červená barva). a) Teplotní mapa SRAM v -40 °C. b) Teplotní mapa SRAM v +85 °C.

Na základě předchozího experimentu v našem návrhu výběru bitů pro SRAM PUF zvolíme maximální povolenou chybovost 0.1%. Graf 7.3 prezentuje výsledky preselekce bitů s maximální povolenou chybovostí 0.1% deseti přípravků ve třinácti teplotních úsecích (viz dodatková tabulka A.1). Můžeme pozorovat, že počet nalezených stabilních bitů se výrazně mění v závislosti na teplotě. Při teplotě -40 °C využíváme průměrně 81.5% měřeného úseku SRAM ke tvorbě bitové masky, zatímco při teplotě +85 °C využíváme průměrně pouze 71.3% měřeného úseku SRAM.

Teplotní mapa 7.4 paměti přípravku MCU 1 ilustruje poměr bitů, které tvoří bitovou masku stabilní paměti a bitů, které byly vyhodnoceny jako nestabilní a dále s nimi nepracujeme. Můžeme pozorovat, že při nižší teplotě (-40 °C) bitová maska obsahuje více jedniček (reprezentujích stabilní bity) než při vyšší teplotě (+85 °C).

Výsledkem prezentované preselekce dle průměrné stability je tedy bitová maska SRAM každého z deseti použitých přípravků pro všech třináct zvolených teplot.



Obrázek 7.2: Výběr bitů pro SRAM PUF s různou maximální povolenou chybovostí stabilních bitů.

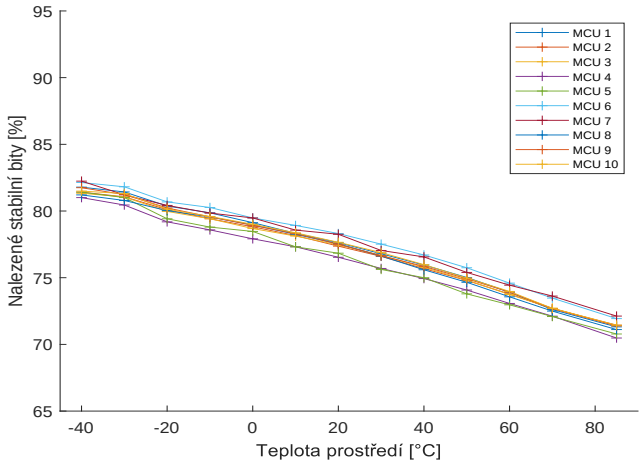
7.2 Preselekce dle teplotní masky

Pro testování preselekce stabilních bitů pro SRAM PUF dle teplotní masky vytvoříme teplotní masku pro každé z deseti použitých zařízení STM32F030R8 jako logický součin jednotlivých bitových masek získaných z předchozí fáze preselekce dle průměrné stability.

Graf 7.5 ilustruje, jak teplotní maska vyrovnává rozdíly v stabilitě bitů SRAM, způsobené vlivem různých teplotních podmínek. Můžeme pozorovat, že ve všech zvolených teplotních úsecích využíváme průměrně 61.6% měřeného úseku SRAM ke tvorbě teplotní masky.

Dále jsme měřili vliv počtu čtení SRAM na množství nalezených stabilních bitů. Testovali jsme čtyři různé počty čtení paměti: 100, 300, 500 a 1000. Graf 7.6 ukazuje, že s rostoucím počtem čtení stoupá i počet nalezených stabilních bitů. Na základě tohoto měření jsme zvolili počet 1000 čtení neinicializované SRAM v první fázi preselekce.

Výsledkem prezentované preselekce dle teplotní masky je tedy jedna teplotní maska stabilních bitů SRAM pro každé z deseti použitých zařízení.



Obrázek 7.3: Závislost počtu nalezených stabilních bitů na změně teploty prostředí.

7.3 Analýza vlastností PUF

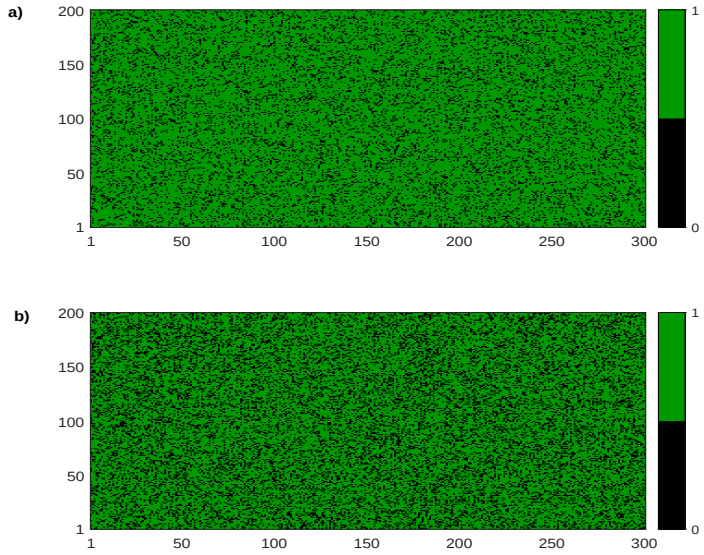
V této kapitole prezentujeme ohodnocení kvality navrženého SRAM PUFu na základě následujících veličin: náhodnosti, spolehlivosti, jednoznačnosti, entropie a pravděpodobnosti chybného přijetí a odmítnutí. Testování provádíme na deseti přípravcích STM32F030R8 ve třinácti různých teplotních podmínkách.

Náhodnost

Náhodnost SRAM PUFu představuje poměr SRAM buněk, které mají hodnotu logické 1 vůči těm, které mají hodnotu logické 0. Náhodnost SRAM PUFu zjišťujeme následovně. Nejprve jednou přečteme daný úsek SRAM (60000 bitů) ve třinácti zvolených teplotách. Pomocí navrhnuté preselekce ze získaných třinácti bitových vektorů paměti vybereme stabilní bity. Z hodnot stabilních bitů poté vyhodnocujeme náhodnost SRAM PUFu na základě výpočtu Hammingovy váhy dle rovnice 2.1.

V tabulce 7.1 prezentujeme pro ilustraci část výsledků měření náhodnosti. Výsledky měření všech deseti přípravků ve třinácti různých teplotních podmínkách uvádíme v dodatkové tabulce A.2.

Získané hodnoty Hammingovy váhy téměř odpovídají 50% ideálního PUFu. Můžeme ale pozorovat drobnou tendenci paměťových buněk klonit se spíše k hodnotě logické nuly, která je výrazná hlavně u nízkých teplot -40 °C a -30 °C. Tato tendence ovšem může být způsobena nedostatečnou délkou setrvání přípravku ve standby módu. Paměť je totiž před vstupem do standby

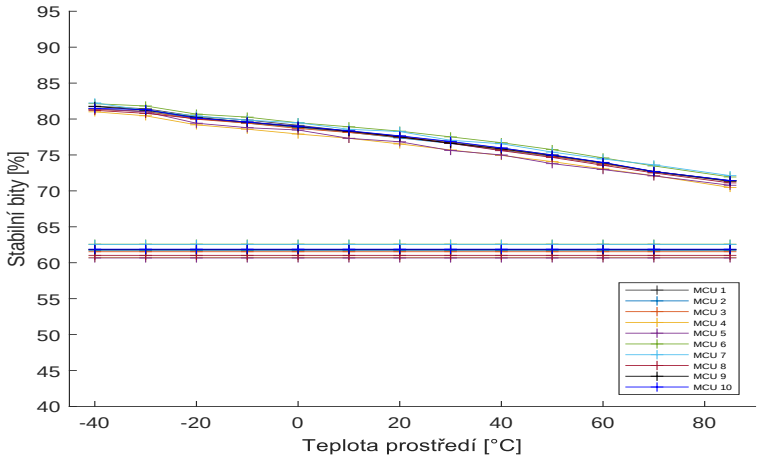


Obrázek 7.4: Bitová maska stabilní SRAM přípravku MCU 1. Hodnota 1 značí stabilní bit, který je součástí bitové masky (zelená barva), zatímco hodnota 0 označuje zbylé a zahozené nestabilní bity (černá barva). a) Teplotní mapa SRAM v -40 °C. b) Teplotní mapa SRAM v +85 °C.

Tabulka 7.1: Měření náhodnosti tří přípravků v deseti teplotních podmínkách. Náhodnost je vyhodnocena pomocí Hammingovy váhy (v %). Ideální PUF by měl mít Hammingovu váhu rovnou 50%.

	Teplota prostředí [°C]									
MCU	**-40**	**-30**	**-10**	**0**	**10**	**30**	**40**	**60**	**70**	**85**
1	49.5	49.67	49.68	49.68	49.68	49.68	49.68	49.68	49.68	49.68
2	50.3	50.31	50.31	50.31	50.31	50.31	50.31	50.31	50.31	50.31
3	49.89	49.98	49.99	49.99	49.99	49.99	49.99	49.99	49.99	49.98
Standby [s]	2	1	1	1	1	1	1	1	1	1

módu inicializována na hodnotu logické 0, abychom zabránili čtení zastaralých hodnot paměti. Na základě tohoto předpokladu jsme provedli další měření náhodnosti deseti přípravků v teplotách -40 °C a -30 °C s delší dobou setrvání přípravku ve standby módu. V tabulce 7.2 prezentujeme část výsledků tohoto měření. Měření všech deseti přípravků v teplotách -40 °C a -30 °C uvádíme v dodatkové tabulce A.3. Výsledky těchto měření potvrdily předpoklad, že se paměť nestihla inicializovat, jelikož prodloužení doby setrvání doby ve standby módu způsobilo zlepšení získaných hodnot Hammingovy váhy.



Obrázek 7.5: Snížení rozdílů ve stabilitě bitů v různých teplotních podmínkách pomocí teplotní masky SRAM. a) Počet stabilních bitů jednotlivých bitových masek SRAM klesá s rostoucí teplotou. b) Počet stabilních bitů teplotní masky SRAM zůstává vyrovnaný v celém teplotním rozsahu.

Tabulka 7.2: Měření náhodnosti tří přípravků ve dvou teplotních podmínkách (v %). Doba setrvání přípravku ve standby módu je prodloužena na 4 s.

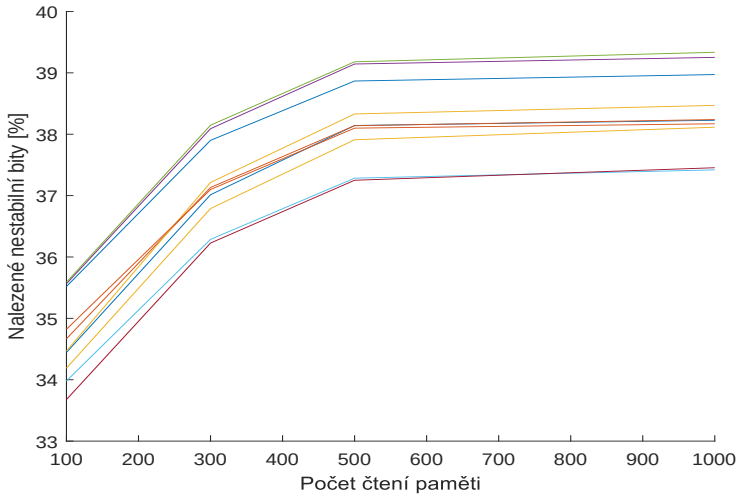
	Teplota prostředí [°C]	
MCU	**-40**	**-30**
1	49.68	49.68
2	50.31	50.31
3	49.99	49.99
Standby mód [s]	4	4

Reprodukovatelnost

Reprodukovatelnost SRAM PUFu představuje identičnost PUF odpovědí daného přípravku v různých externích podmínkách. Reprodukovatelnost SRAM PUFu zjišťujeme následovně. Nejprve jednou přečteme daný úsek SRAM (60000 bitů) ve třinácti zvolených teplotách. Pomocí navrhnuté preselekce ze získaných třinácti bitových vektorů paměti vybereme stabilní bity. Poté určíme referenční vektor vůči kterému budeme porovnávat ostatní PUF odpovědi. Zvolili jsme vektor stabilních bitů SRAM v teplotě 0 °C. Následně vyhodnocujeme HD_{intra} porovnáním referenčního vektoru a PUF odpověďmi stejného zařízení ve dvanácti jiných teplotách dle rovnice 2.2.

V tabulce 7.3 prezentujeme výsledky výpočtu minimální, maximální a průměrné HD_{intra} pro každé z deseti zařízení. Kompletní výsledky HD_{intra} uvádíme v dodatkové tabulce A.4.

Získané hodnoty HD_{intra} téměř odpovídají 0% ideálního PUFu. Můžeme



Obrázek 7.6: Závislost množství nalezených stabilních bitů SRAM na počtu čtení paměti.

Tabulka 7.3: Měření minimální, maximální a průměrné HD_{intra} deseti přípravků ve třinácti teplotních podmínkách (v %). Ideální PUF by měl mít HD_{intra} rovno 0 %.

MCU	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
min HD_{intra}	0	0	0	0	0	0	0	0	0	0
max HD_{intra}	0.003	0.003	0.002	0.002	0	0.002	0	0.005	0	0.002
avg HD_{intra}	0.0002	0.0005	0.0002	0.0002	0	0.0002	0	0.0004	0	0.0002

Tabulka 7.4: Měření reprodukovatelnosti přípravku MCU 1 (v %).

	Teplota prostředí [°C]												
MCU	**-40**	**-30**	**-20**	**-10**	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**85**
1	100	100	100	100	x	100	100	100	100	100	100	100	99.997
2	100	100	100	100	x	100	100	100	99.997	100	100	100	99.997
3	100	100	100	100	x	100	100	100	100	100	100	100	99.998

pozorovat, že nejvíce odlišných bitů se nachází v PUF odpovědích měřených ve vyšších teplotách, především v teplotě +85 °C.

Reprodukovatelnost SRAM PUFu lze zjistit pomocí rovnice 2.3. V tabulce 7.4 prezentujeme výsledky výpočtu reprodukovatelnosti přípravku MCU 1.

Jedinečnost

Jedinečnost SRAM PUFu představuje unikátnost PUF odpovědí každého přípravku. Jedinečnost SRAM PUFu zjišťujeme následovně. Nejprve jednou přečteme daný úsek SRAM (60000 bitů) všech deseti přípravků ve zvolených

Tabulka 7.5: Měření jedinečnosti deseti přípravků ve 20 °C. Jedinečnost je vyhodnocena pomocí HD_{inter} (v %). Ideální PUF by měl mít HD_{inter} rovnou 50%.

MCU	1	2	3	4	5	6	7	8	9	10
1		50.18	49.99	51.42	51.28	50.81	50.325	50.43	49.4	50.55
2			50.06	50.73	50.54	50.84	50.53	50.18	50.57	49.98
3				50.65	50.62	51.14	50.58	50.77	50.02	49.88
4					50.09	51.18	51.07	50.55	50.91	50.98
5						51.56	51.3	50.80	50.75	50.78
6							50.26	50.79	50.25	50.89
7								51.25	51.05	50.65
8									50.47	51.03
9										50.27
10										

teplotách. Pomocí navrhnuté preselekce ze získaných deseti bitových vektorů paměti vybereme stabilní bity. Poté vyhodnocujeme HD_{inter} porovnáním všech deseti stabilních bitových vektorů navzájem dle rovnice 2.4. Vzhledem k tomu, že počet stabilních bitů je na každém zařízení jiný, porovnáváme dvojice různě dlouhých bitových řetezců. K výsledné Hammingově vzdálenosti stejně dlouhých úseků vektorů tedy ještě příčítáme přebytečnou délku delšího z řetězců.

V tabulce 7.5 prezentujeme výsledky měření HD_{inter} deseti přípravků v teplotě 20 °C. V dodatkové tabulce A.5 uvádíme výsledky měření HD_{inter} navíc i v teplotě -40 °C a 85 °C.

Získané hodnoty HD_{inter} téměř odpovídají 50% ideálního PUFu. Můžeme pozorovat, že rozdíly HD_{inter} měřené v různých teplotách jsou zanedbatelné.

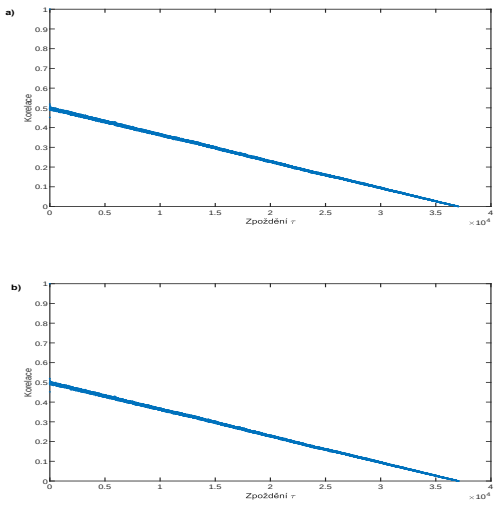
Entropie

Entropie SRAM PUFu představuje nízkou korelaci mezi jednotlivými bity PUF odpovědi. Entropii SRAM PUFu zjišťujeme následovně. Nejprve jednou přečteme daný úsek SRAM (60000 bitů) ve zvolených teplotách. Pomocí navrhnuté preselekce ze získaných bitových vektorů paměti vybereme stabilní bity. Poté vyhodnocujeme korelaci mezi jednotlivými stabilními bity jedné PUF odpovědi dle rovnice 2.5.

V grafu 7.7 prezentujeme výsledek měření korelace mezi jednotlivými stabilními bity přípravku MCU 1 v teplotě -40 °C a 85 °C. Ke korelaci mezi bity dochází pouze při zpoždění $\tau = 0$, jelikož zde bit koreluje sám se sebou. Můžeme pozorovat, že rozdíly v korelaci jednotlivých bitů měřené v různých teplotách jsou zanedbatelné.

Pravděpodobnost chybného přijetí a odmítnutí

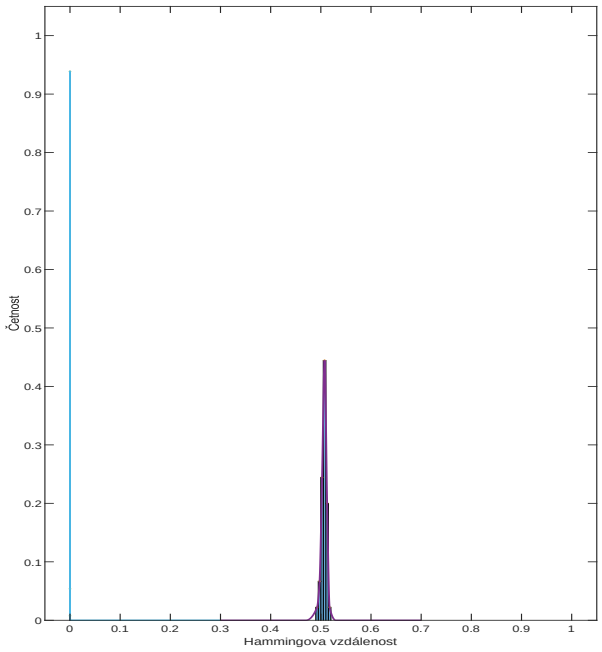
Pravděpodobnost, že PUF A bude identifikován jako PUF B (PUF A $\neq$ B) a PUF A generuje stejný výstup jako PUF B, označujeme jako pravděpodobnost



Obrázek 7.7: Korelace mezi jednotlivými stabilními bity PUF odpovědi přípravku MCU 1. a) Měření korelace v teplotě -40 °C. b) Měření korelace v teplotě 85 °C.

chybného přijetí. Pravděpodobnost, že korektní PUF nebude úspěšně identifikován, označujeme jako pravděpodobnost chybného odmítnutí. Pravděpodobnost chybného přijetí a odmítnutí zjišťujeme následovně. Nejprve určíme optimální mez identifikovatelnosti pomocí předchozích výpočtů $\mathrm{HD}_{\mathrm{intra}}$ a $\mathrm{HD}_{\mathrm{inter}}$ všech deseti přípravků ve třinácti teplotních podmínkách dle grafu 7.8. Jelikož se histogramy navzájem nepřekrývají, je možná bezchybná identifikace přípravku v rozsahu mezi nimi, resp. pokud je Hammingova vzdálenost mezi odpověďmi PUFu ze stejných zařízení ($\mathrm{HD}_{\mathrm{intra}}$) menší než 4.9% a pokud je Hammingova vzdálenost odpovědí PUFů z různých zařízení ($\mathrm{HD}_{\mathrm{inter}}$) větší než 47.1%.

Pravděpodobnost chybného přijetí p_{mis} vypočítáme pomocí rovnice 2.6. Pro zvolené ϵ v rozsahu od 4.9% do 47.1% je výsledek 0%. Pravděpodobnost chybného odmítnutí p_{reject} vypočítáme pomocí rovnice 2.7. Pro zvolené ϵ v rozsahu od 4.9% do 47.1% je výsledek 0%.



Obrázek 7.8: Nalezení optimální meze Hammingovy vzdálenosti pro identifikaci zařízení. Histogram vlevo (resp. vpravo) reprezentuje HD_{intra} (resp. HD_{inter}) deseti přípravků ve třinácti teplotních podmínkách.

Kapitola 8

Analýza výsledků

V této kapitole základě výsledků z předchozí kapitoly prezentujeme aplikaci navrhnutého SRAM PUFu pro identifikaci jednotlivých přípravků a také pro generování stabilních kryptografických klíčů.

8.1 Identifikace zařízení

Tato část je inspirována [2]. Na základě provedených měření v předchozí kapitole můžeme navrhnout identifikaci mikrokontrolérů STM32F030R8 pomocí neinicializovaných hodnot SRAM, nezávislou na externích teplotních podmínkách.

Nejprve představíme návrh identifikace deseti testovaných přípravků. Vzhledem k tomu, že nejvyšší naměřená hodnota HD_{intra} je pouze 0.002% a zároveň nejmenší naměřená hodnota HD_{inter} je 41%, všechny testované přípravky mohou být bezchybně identifikovány.

Identifikaci testovaných přípravků provedeme následovně. Nejprve přečteme zvolený úsek paměti neinicializované SRAM (60000 bitů) ve třinácti teplotních podmínkách. Pomocí navrhnuté preselekce ze získaných bitových vektorů paměti vybereme stabilní bity. Poté porovnáváme stabilní bity jednotlivých deseti přípravků mezi sebou a hledáme nejmenší souvislý úsek (od počátku paměti), který je schopný rozlišit všech deset zařízení ve všech teplotních podmínkách. Na základě prezentovaného postupu jsme zjistili, že všech deset testovaných přípravků lze identifikovat pouze pomocí prvních pěti stabilních bitů (viz tabulka 8.1).

Obecně je vhodné použít k identifikaci přípravků co největší úsek SRAM, aby byla zachována unikátnost každého vektoru stabilních bitů i při použití velkého množství zařízení. V takovém případě ve fázi inicializace uložíme do databáze všechny nalezené stabilní bity SRAM každého přípravku. Počet stabilních bitů závisí na konkrétní SRAM a může se tedy mezi přípravky lišit. (Na základě provedených měření preselekce stabilních bitů v kapitole 7.2 víme, že délka těchto jednotlivých identifikátorů bude průměrně 36000 bitů.) Během

Tabulka 8.1: Identifikace deseti testovaných zařízení. Tabulka uvádí počet rozdílných bitů mezi různými přípravky v teplotě 0 °C.

MCU	1	2	3	4	5	6	7	8	9	10
1	0	3	1	4	1	4	2	3	5	2
2	3	0	2	3	2	1	3	4	2	1
3	1	2	0	5	2	3	1	4	4	1
4	4	3	5	0	3	2	4	1	1	4
5	1	2	2	3	0	3	3	4	4	1
6	4	1	3	2	3	0	4	3	1	2
7	2	3	1	4	3	4	0	3	3	2
8	3	4	4	1	4	3	3	0	2	5
9	5	2	4	1	4	1	3	2	0	3
10	2	1	1	4	1	2	2	5	3	0

samotné identifikace dojde k porovnání identifikátorů uložených v databázi a aktuální PUF odpovědi (tj. vektorem stabilních bitů SRAM) přípravku, který žádá o identifikaci.

Na základě předchozího měření pravděpodobnosti chybného přijetí, odmítnutí a optimální meze identifikovatelnosti v kapitole 7.3 víme, že přípravek bude úspěšně identifikován, pokud je Hammingova vzdálenost mezi všemi identifikátory (kromě jednoho) uloženými v databázi a aktuální PUF odpovědí je větší než 47.1% a zároveň je Hammingova vzdálenost mezi identifikátorem uloženým v databázi a aktuální PUF odpovědí menší než 4.9%. Pokud během identifikace není v databázi nalezen žádný identifikátor, který splňuje oba požadavky, identifikace přípravku je zamítnuta.

8.2 Generování kryptografických klíčů

Na základě provedených měření v předchozí kapitole můžeme implementovat generování 128-bitového kryptografického klíče pomocí SRAM PUFu, nezávislé na externích tepotních podmínkách. K dosažení zcela stabilních PUF odpovědí používáme samoopravný kód BCH(31,26) nebo repetiční kód délky 11.

Generování kryptografického klíče probíhá následovně. Nejprve dle navrhnuté preselekce stabilních bitů opakovaně přečteme zvolený úsek SRAM (512 bitů v případě BCH kódu nebo 4096 bitů v případě repetičního kódu) ve třinácti zvolených teplotách (v rozsahu od -40 °C do 85 °C). Ze získaných třinácti bitových vektorů paměti vybereme stabilní bity nezávislé na externí teplotě.

Proces generování klíče dělíme na inicializaci a rekonstrukci klíče. Během inicializace pro každý z deseti přípravků z vybraných stabilních bitů vygenerujeme kódová slova v příslušném samoopravném kódu a uložíme pomocná data do paměti Flash. Proces inicializace probíhá pouze jednou a to ve 20 °C. Následně 300krát provedeme rekonstrukci klíče (a jeho opravu v příslušném samoopravném kódu) pro každou ze třinácti zvolených teplot. Celkem je tedy rekonstruováno 3900 odpovědí pro každý z deseti přípravků. Všechny odpovědi

Tabulka 8.2: Počet získaných bitů kryptografického klíče opraveného pomocí BCH a repetičního samoopravného kódu.

	Samoopravný kód	
MCU	**BCH(31,26,1)**	**RP(15,1,5)**
1	208	213
2	234	224
3	182	218
4	208	220
5	208	218
6	208	213
7	208	216
8	208	221
9	182	213
10	208	219

byly rekonstruovány se 100% úspěšností.

Vzhledem k tomu, že generujeme kódová slova pouze ze stabilních bitů, jejichž počet se na každém přípravku liší, získáme pro každý přípravek jiný počet bitů. V tabulce 8.2 uvádíme přesný počet získaných bitů klíče pro každý přípravek. Pozorujeme, že v případě BCH kódu jsme získali od nejméně 182 bitů do nejvíce 234 bitů klíče, zatímco v případě repetičního kódu jsme získali od nejméně 213 bitů do nejvíce 224 bitů klíče. V obou případech jsme tedy schopni generovat 128-bitový kryptografický klíč.

Závěr

V práci jsme se zabývali návrhem a implementací PUFu na nízkonákladovém mikrokontroléru STM32F030R8. Představili jsme různé metody výběru bitů pro SRAM PUF a na jejich základě jsme navrhli vlastní metodu vyhodnocení PUF a provedli její implementaci, kterou jsme následně otestovali v různých teplotních podmínkách. Navázali jsme na práce [1] a [2].

V první kapitole jsme představili obecný koncept PUFu. Uvedli jsme přehled vlastností, kterými by měl PUF disponovat. Dále jsme popsali různé možnosti členění PUFů, například na základě počtu párů výzev a odpovědí a jejich zabezpečení. Podrobněji jsme se zaměřili hlavně na tři tzv. *Intrinsic Memory-based* PUFy, podobné SRAM PUFu, který využíváme v praktické části práce. Nakonec jsme uvedli tři základní aplikace PUFu, identifikaci a autentizaci zařízení a generování kryptografických klíčů.

Ve druhé kapitole jsme nejprve popsali princip SRAM pamětí a následně představili jejich využití pro PUF. Následně jsme uvedli možnosti ohodnocení kvality SRAM PUFů na základě následujících veličin: náhodnosti, spolehlivosti, entropie a pravděpodobnosti chybného přijetí a odmítnutí.

Ve třetí kapitole jsme nejprve obecně popsali využití samoopravných kódů pro generování kryptografických klíčů a následně podrobněji představili dva samoopravné kódy, BCH a repetiční, které využíváme v praktické části práce.

Ve čtvrté kapitole jsme představili různé metody výběru bitů pro SRAM PUF. Podrobněji jsme se zabývali metodou výběru bitů Labana et al. v [1].

V páté kapitole jsme uvedli vlastní návrh metody výběru stabilních bitů pro SRAM PUF mikrokontroléru STM32F030R8. Uvedená metoda se skládá ze dvou částí, preselekce bitů dle průměrné stability a preselekce bitů dle teplotní masky. Zvolená metoda preselekce bitů byla navrhnuta tak, abychom dosáhli stabilních PUF odpovědí ve všech třinácti zvolených teplotních podmínkách (v rozsahu od -40 °C do 85 °C).

Nad rámec zadání práce jsme se zde věnovali i návrhu aplikace vytvořeného PUFu pro generování kryptografického klíče. Ke stabilizaci PUF odpovědí jsme použili samoopravný kód BCH(31,26) (stejně jako [1]) a repetiční kód

s délkou slova 11 (stejně jako [2]).

V šesté kapitole jsme popsali realizaci návrhu SRAM PUFu. Představili jsme vlastnosti použitého nízkonákladového mikrokontroléru STM32F030R8. Dále jsme popsali zapojení mikrokontroléru a teplotní komory během měření.

V sedmé kapitole jsme prezentovali výsledky testování navrhnutého SRAM PUFu. Vytvořený PUF byl testován na deseti přípravcích a ve třinácti různých teplotních podmínkách. Měřili jsme celkem 60000 bitů neinicializované SRAM (tj. 92% velikosti SRAM, zbylých 8% zabíral samotný program). Nejprve jsme testovali preselekci dle průměrné stability bitů. Pozorovali jsme, že počet nalezených stabilních bitů se výrazně mění v závislosti na teplotě. Zatímco při teplotě -40 °C jsme jako stabilní označili průměrně 81.5% měřeného úseku SRAM, při teplotě 85 °C jsme jako stabilní označili pouze 71.3% měřeného úseku SRAM. Následně jsme testovali preselekci dle teplotní masky. Teplotní maska vyrovnala rozdíly v nalezených stabilních bitech SRAM, způsobených vlivem různých teplotních podmínek. Po použití preselekce teplotní maskou jsme jako stabilní označili průměrně 61.6% měřeného úseku SRAM. Tyto bity jsme dále využili pro identifikaci jednotlivých přípravků nebo pro generování kryptografického klíče.

Dále jsme se v sedmé kapitole věnovali ohodnocení kvality navrženého SRAM PUFu na základě veličin uvedených v kapitole 2.1. Na základě výpočtu Hammingovy váhy hodnot stabilních bitů jsme zjistili, že je nutné prodloužit dobu setrvání přípravků ve standby módu v teplotách -40 °C a -30 °C ze 2 s na 4 s. Dále jsme testovali například hodnoty metrik HD_{intra} a HD_{inter}, které se obě velice blížili hodnotám ideálního PUFu.

V poslední kapitole jsme na základě výsledků z předchozí kapitoly uvedli aplikaci vytvořeného SRAM PUFu pro identifikaci jednotlivých přípravků. Zjistili jsme, že všech deset testovaných přípravků lze úspěšně identifikovat pouze pomocí prvních pěti stabilních bitů. Na základě měření HD_{intra}, HD_{inter} a optimální meze identifikovatelnosti lze tvrdit, že mikrokontrolér STM32F030-R8 lze identifikovat, pokud je Hammingova vzdálenost mezi všemi identifikátory (kromě jednoho) uloženými v databázi a aktuální PUF odpovědí je větší než 47.1% a zároveň je Hammingova vzdálenost mezi identifikátorem uloženým v databázi a aktuální PUF odpovědí menší než 4.9%.

Dále jsme v poslední kapitole aplikovali BCH a repetiční samoopravný kód na vytvořený PUF a úspěšně generovali stabilní 128-bitové kryptografické klíče, nezávislé na externích teplotních podmínkách.

Na závěr uvádíme možnosti pro budoucí výzkum:

- Testování navrženého SRAM PUFu nejen v různých teplotních, ale také v různých napěťových podmínkách (stejně jako [1]).

- Aplikování jiných samoopravných kódů, použitých při generování kryptografického klíče (například BCH(31,21) dle [1]).

- Testování delší doby setrvání přípravku ve standby módu a její vliv na počet nalezených stabilních bitů.
- Vliv stárnutí přípravků na vytvořený návrh SRAM PUFu.

Literatura

[1] Laban, M.; Drutarovský, M.: Compact Bit Selection Procedure for SRAM PUF Embedded in a Low-cost Microcontroller. *Sborník příspěvků PAD 2019 – elektronická verze. Praha: AMCA spol. s r.o.*, 09 2019: s. 27–30, [cit. 2020-07-01]. Dostupné z: `https://anc.fit.cvut.cz/pad2019/sbornik_files/sbornik-PAD2019.pdf`

[2] Platonov, M.; Hlaváč, J.; Lórencz, R.: Using Power-Up SRAM State of Atmel ATmega1284P Microcontrollers as Physical Unclonable Function for Key Generation and Chip Identification. *Information Security Journal: A Global Perspective*, ročník 22, 11 2013: s. 244–250.

[3] Buček, J.: HW Security – Side channel types, algorithm and technology. Simple Power Analysis. [online], 2019, [cit. 2020-07-2]. Dostupné z: `https://courses.fit.cvut.cz/MI-HWB/media/lectures/hwb3.pdf`

[4] Platonov, M.: *SRAM-Based Physical Unclonable Function on an Atmel ATmega Microcontroller*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[5] IntrinsicID: SRAM PUF: The Secure Silicon Fingerprint – White Paper. [online], 2019, [cit. 2020-07-26]. Dostupné z: `http://go.intrinsic-id.com/securesilicon-fingerprint-lp`

[6] NXP: LPC5500 Series - World's Arm Cortex-M33 based Microcontroller Series for Mass Market, Leveraging 40nm Embedded Flash Technology. [online], 2018, [cit. 2020-07-26]. Dostupné z: `https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc5500-cortex-m33:LPC5500_SERIES`

[7] Maes, R.: *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer Berlin Heidelberg, 2013, ISBN 978-3-642-

41395-7, 49-80 s., [cit. 2020-07-20]. Dostupné z: https://doi.org/10.1007/978-3-642-41395-7_3

[8] Kodýtek, F.: *Fyzicky neklonovatelné funkce na FPGA*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

[9] Pappu, R. S.: *Physical One-Way Functions*. Dizertační práce, Massachusetts Institute of Technology, 2002, [cit. 2020-07-17]. Dostupné z: http://alumni.media.mit.edu/~pappu/pdfs/Pappu-PhD-POWF-2001.pdf

[10] Gassend, B.; Clarke, D.; van Dijk, M.; aj.: Silicon Physical Random Functions. *Association for Computing Machinery*, 2002: str. 148–160, [cit. 2020-07-11]. Dostupné z: https://doi.org/10.1145/586110.586132

[11] Tuyls, P.; Skoric, B.; Stallinga, S.; aj.: Information-Theoretic Security Analysis of Physical Uncloneable Functions. *Lecture Notes in Computer Science*, ročník 3570, 02 2005: s. 141–155.

[12] Holcomb, D. E.; Burleson, W. P.; Fu, K.: Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers*, ročník 58, č. 9, 2009: s. 1198–1210, [cit. 2020-07-19]. Dostupné z: https://spqr.eecs.umich.edu/papers/holcomb-FERNS-IEEE-Computers.pdf

[13] Katzenbeisser, S.; Kocabaş, Ü.; Rožić, V.; aj.: *PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon*. Springer Berlin Heidelberg, 2012, ISBN 978-3-642-33027-8, 283–301 s.

[14] Mispan, M. S.: *Towards reliable and secure physical unclonable functions*. Dizertační práce, University of Southampton, Faculty of Physical Sciences and Engineering, 07 2018.

[15] Gassend, B.; Dijk, M. V.; Clarke, D.; aj.: Controlled Physical Random Functions and Applications. *ACM Transactions on Information and System Security*, 2008, [cit. 2020-07-11]. Dostupné z: https://doi.org/10.1145/1284680.1284683

[16] Joshi, S.; Mohanty, S.; Kougianos, E.: Everything You Wanted to Know about PUFs. *IEEE Potentials*, ročník 36, 11 2017: s. 38–46.

[17] Kodýtek, F.: *Behaviour Analysis and Improvement of the Proposed PUF on FPGA*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[18] Pérez-Jiménez, M.; Bordel Sánchez, B.; Migliorini, A.; aj.: Protecting Private Communications in Cyber-Physical Systems through Physical Unclonable Functions. *Electronics*, ročník 8, 04 2019: str. 390.

[19] Buchanan, J.; Cowburn, R.; Jausovec, A.; aj.: Forgery: 'fingerprinting' documents and packaging. *Nature*, ročník 436, 2005: s. 475–475.

[20] Hammouri, G.; Dana, A.; Sunar, B.: CDs Have Fingerprints Too. *Cryptographic Hardware and Embedded Systems - CHES 2009*, 2009: s. 348–362.

[21] Busch, H.; Sotáková, M.; Katzenbeisser, S.; aj.: The PUF Promise. *International Conference on Trust and Trustworthy Computing, Berlin*, 06 2010: s. 290–297.

[22] Suri, M. (editor): *Applications of Emerging Memory Technology : Beyond Storage*. Springer Singapore, 2020, ISBN 978-981-13-8378-6.

[23] Guajardo, J.; Kumar, S. S.; Schrijen, G.-J.; aj.: FPGA Intrinsic PUFs and Their Use for IP Protection. *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2007*, 2007: s. 63–80.

[24] Lee, J. W.; Daihyun Lim; Gassend, B.; aj.: A technique to build a secret key in integrated circuits for identification and authentication applications. *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, 2004: s. 176–179.

[25] Su, Y.; Holleman, J.; Otis, B.: A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations. *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, 2007: s. 406–611.

[26] Kumar, S. S.; Guajardo, J.; Maes, R.; aj.: Extended abstract: The butterfly PUF protecting IP on every FPGA. *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008: s. 67–70.

[27] Simons, P.; van der Sluis, E.; van der Leest, V.: Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2012: s. 7–12.

[28] Hegr, V.: *Fyzicky neklonovatelné funkce*. Diplomová práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017.

[29] Barbareschi, M.; DeBenedictis, A.; Mazzocca, N.: A PUF-based hardware mutual authentication protocol. *Journal of Parallel and Distributed Computing*, ročník 119, 2018: s. 107–120.

[30] Taniguchi, M.; Shiozaki, M.; Kubo, H.; aj.: A stable key generation from PUF responses with a Fuzzy Extractor for cryptographic authentications. *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, 2013: s. 525–527.

[31] Sedlář, J.: *Testování SRAM pamětí s využitím MBIST*. Diplomová práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky, 2018, [cit. 2020-07-24]. Dostupné z: `https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=172282`

[32] Deutschmann, M.; Iriskic, L.; Lattacher, S.-L.; aj.: Research on the Applications of Physically Unclonable Functions within the Internet of Things. *Technikon Forschungs- und Planungsgesellschaft mbH Burgplatz 3a, 9500 Villach, Austria*, 08 2018.

[33] Böhm, C.; Hofer, M.: *Physical Unclonable Functions in Theory and Practice*. Springer Publishing Company, Incorporated, 2012, ISBN 9781461450399.

[34] Böhm, C.; Hofer, M.; Pribyl, W.: A microcontroller SRAM-PUF. *5th International Conference on Network and System Security*, 2011.

[35] Maiti, A.; Gunreddy, V.; Schaumont, P.: A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. *IACR Cryptology ePrint Archive*, ročník 2011, 01 2011: str. 657.

[36] Golomb, S. W.; Gong, G.; Helleseth, T.; aj.: *Sequences, Subsequences, and Consequences, Interna-tional Workshop, SSC 2007, Los Angeles, CA, USA, May 31 - June 2,2007, Revised Invited Papers*, ročník 4893. Springer-Verlag Berlin Heidelberg, 2007, ISBN 978-3-540-77404-4.

[37] Suh, G. E.; Devadas, S.: *Physical Unclonable Functions for Device Authentication and Secret Key Generation*. 2007 44th ACM/IEEE Design Automation Conference, 2007, 9-14 s., [cit. 2020-07-15]. Dostupné z: `http://people.csail.mit.edu/devadas/pubs/puf-dac07.pdf`

[38] Yu, M.; Devadas, S.: Secure and robust error correction for physical unclonable functions. *IEEE Design Test of Computers*, ročník 27, č. 1, 2010: s. 48–65.

[39] Pluháček, A.: MI-BKO – Dvojkové kódy BCH. [online], 2020, [cit. 2020-07-30]. Dostupné z: `https://courses.fit.cvut.cz/MI-BKO/media/lectures/BKO-1-08-H-BCH.pdf`

[40] Miller, A.; Shifman, Y.; Weizman, Y.; aj.: A Highly Reliable SRAM PUF with a Capacitive Preselection Mechanism and pre-ECC BER of 7.4E-10. *2019 IEEE Custom Integrated Circuits Conference (CICC)*, 2019: s. 1–4.

[41] Shifman, Y.; Miller, A.; Keren, O.; aj.: A Method to Improve Reliability in a 65-nm SRAM PUF Array. *IEEE Solid-State Circuits Letters*, ročník 1, č. 6, 2018: s. 138–141.

[42] Koeberl, P.; Li, J.; Rajan, A.; aj.: Entropy loss in PUF-based key generation schemes: The repetition code pitfall. *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014: s. 44–49.

[43] Mathew, S. K.; Satpathy, S. K.; Anders, M. A.; aj.: 16.2 A 0.19pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014: s. 278–279.

[44] Satpathy, S.; Mathew, S.; Suresh, V.; aj.: A 4-fJ/b Delay-Hardened Physically Unclonable Function Circuit With Selective Bit Destabilization in 14-nm Trigate CMOS. *IEEE Journal of Solid-State Circuits*, 01 2017: s. 1–10.

[45] Koeberl, P.; Li, J.; Wu, W.: *A Spatial Majority Voting Technique to Reduce Error Rate of Physically Unclonable Functions*. Springer International Publishing, 2013, ISBN 978-3-319-03491-1, 36-52 s.

[46] STMicroelectronics: STM32F030x4, STM32F030x6, STM32F030x8, STM32F030xC Datasheet. [online], 2019, [cit. 2020-07-26]. Dostupné z: `https://www.st.com/resource/en/datasheet/stm32f030f4.pdf`

[47] Šádek, K.: *True Random Number Generator on FPGA*. Bakalářská práce, České vysoké učení technické v Praze. Fakulta informačních technologií., 2020.

[48] Binder: Model MK 56. [online], 2019, [cit. 2020-07-4]. Dostupné z: `https://www.binder-world.com/en/products/dynamic-climate-chambers/series-mk/mk-56`

Příloha A

Výsledky měření

V této příloze prezentujeme kompletní výsledky měření stability, náhodnosti, reprodukovatelnosti a jedinečnosti navrženého SRAM PUFu, které kvůli své velikosti nebyly zařazeny do hlavního textu práce.

Tabulka A.1: Výsledky měření preselekce bitů dle průměrné stabity SRAM deseti přípravků (MCU 1 až 10) ve třinácti teplotních podmínkách (v rozsahu od -40 °C do +85 °C) v %.

	Teplota prostředí [°C]												
MCU	**-40**	**-30**	**-20**	**-10**	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**85**
1	81.79	81.43	80.376	79.863	79.128	78.325	77.628	76.796	75.951	75.018	73.913	72.61	71.313
2	81.418	81.061	80.101	79.42	78.818	78.16	77.366	76.6	75.68	74.795	73.77	72.686	71.44
3	81.331	81.056	79.97	79.421	78.685	78.123	77.456	76.675	75.778	74.826	73.743	72.716	71.4
4	81.005	80.45	79.188	78.585	77.915	77.318	76.531	75.695	74.935	74.076	73.07	72.116	70.478
5	81.415	81.008	79.418	78.795	78.47	77.313	76.838	75.62	75.001	73.788	72.966	72.088	70.78
6	82.136	81.816	80.683	80.265	79.481	78.92	78.311	77.516	76.705	75.746	74.593	73.47	71.933
7	82.24	81.215	80.42	79.848	79.48	78.57	78.256	77.043	76.558	75.396	74.44	73.623	72.123
8	81.208	80.79	80.031	79.555	78.956	78.258	77.543	76.623	75.595	74.656	73.566	72.496	71.111
9	81.748	81.23	80.23	79.565	78.926	78.303	77.511	76.648	75.848	74.925	73.921	72.683	71.388
10	81.493	81.31	80.138	79.591	79.006	78.366	77.655	76.885	75.978	74.98	73.97	72.703	71.445
Standby mód [s]	4	4	1	1	1	1	1	1	1	1	1	1	1

Tabulka A.2: Výsledky měření náhodnosti (Hammingovy váhy) stabilních bitů deseti přípravků (MCU 1 až 10) ve třinácti teplotních podmínkách (v rozsahu od -40 °C do +85 °C) v %.

	Teplota prostředí [°C]												
MCU	**-40**	**-30**	**-20**	**-10**	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**85**
1	49.498	49.673	49.684	49.684	49.684	49.684	49.684	49.684	49.684	49.684	49.684	49.684	49.681
2	50.298	50.311	50.311	50.311	50.311	50.311	50.311	50.311	50.308	50.311	50.311	50.311	50.308
3	49.894	49.983	49.986	49.986	49.986	49.986	49.986	49.986	49.986	49.986	49.986	49.986	49.983
4	50.566	50.574	50.572	50.572	50.572	50.572	50.572	50.572	50.569	50.572	50.572	50.572	50.572
5	49.938	49.938	49.938	49.938	49.938	49.938	49.938	49.938	49.938	49.938	49.938	49.938	49.938
6	49.098	49.758	49.785	49.785	49.785	49.782	49.785	49.785	49.785	49.785	49.785	49.785	49.785
7	49.245	49.832	49.882	49.882	49.882	49.882	49.882	49.882	49.882	49.882	49.882	49.882	49.882
8	51.159	51.159	51.159	51.159	51.159	51.159	51.159	51.159	51.159	51.159	51.159	51.159	51.153
9	50.291	50.412	50.436	50.436	50.436	50.436	50.436	50.436	50.436	50.436	50.436	50.436	50.436
10	49.458	49.48	49.48	49.48	49.48	49.48	49.477	49.48	49.48	49.48	49.48	49.48	49.48
Standby [s]	2	1	1	1	1	1	1	1	1	1	1	1	1

Tabulka A.3: Výsledky měření náhodnosti (Hammingovy váhy) stabilních bitů deseti přípravků (MCU 1 až 10) v teplotě -40 °C a 85 °C v %. Doba setrvání přípravku ve standby módu je zde prodloužena na 4 s.

	Teplota prostředí [°C]	
MCU	**-40**	**-30**
1	49.684	49.684
2	50.311	50.311
3	49.986	49.986
4	50.572	50.572
5	49.938	49.938
6	49.785	49.785
7	49.882	49.882
8	51.159	51.159
9	50.436	50.436
10	49.48	49.48
Standby mód [s]	4	4

Tabulka A.4: Výsledky měření HD_{intra} deseti přípravků (MCU 1 až 10) ve třinácti teplotních podmínkách (v rozsahu od -40 °C do +85 °C) v %. Jako referenční vektor k porovnání byl zvolen vektor stabilních bitů v teplotě 0 °C.

	Teplota prostředí [°C]												
MCU	**-40**	**-30**	**-20**	**-10**	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**85**
1	0	0	0	0	x	0	0	0	0	0	0	0	0.003
2	0	0	0	0	x	0	0	0	0.003	0	0	0	0.003
3	0	0	0	0	x	0	0	0	0	0	0	0	0.002
4	0	0	0	0	x	0	0	0	0.002	0	0	0	0
5	0	0	0	0	x	0	0	0	0	0	0	0	0
6	0	0	0	0	x	0.002	0	0	0	0	0	0	0
7	0	0	0	0	x	0	0	0	0	0	0	0	0
8	0	0	0	0	x	0	0	0	0	0	0	0	0.005
9	0	0	0	0	x	0	0	0	0	0	0	0	0
10	0	0	0	0	x	0	0.002	0	0	0	0	0	0
Standby mód [s]	4	4	1	1	1	1	1	1	1	1	1	1	1

Tabulka A.5: Výsledky měření HD_{inter} deseti přípravků (MCU 1 až 10) v teplotě -40 °C (nahoře) a 85 °C (dole) v %.

MCU	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1		50.18	49.99	51.42	51.28	50.81	50.33	50.43	49.40	50.55
2			50.06	50.73	50.54	50.84	50.53	50.18	50.57	49.98
3				50.65	50.62	51.14	50.58	50.77	50.02	49.87
4					50.09	51.18	51.07	50.55	50.91	50.98
5						51.56	51.30	50.80	50.75	50.77
6							50.26	50.79	50.25	50.88
7								51.25	51.05	50.64
8									50.47	51.02
9										50.28
10										

MCU	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1		50.17	49.99	51.42	51.27	50.80	50.32	50.43	49.40	50.56
2			50.06	50.73	50.54	50.84	50.53	50.18	50.57	49.98
3				50.65	50.63	51.14	50.58	50.77	50.02	49.88
4					50.09	51.18	51.07	50.56	50.91	50.98
5						51.56	51.30	50.81	50.75	50.77
6							50.26	50.79	50.25	50.88
7								51.26	51.05	50.64
8									50.47	51.02
9										50.28
10										

Příloha B

Seznam použitých zkratek

BCH Bose, Ray-Chaudhuri, Hocquenghem

BER Bit Error Rate

CMOS Complementary metal-oxide-semiconductor

FAR False-acceptance rate

FPGA Field-programmable gate array

FRR False-rejection rate

HD Hamming distance

MCU Microcontroller Unit

PUF Physical Unclonable Function

SRAM Static random-access memory

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `data` ..... adresář s výsledky měření
- `src` ..... adresář se zdrojovými kódy
  - `auxiliary_materials` ..... zpracování výsledků v Matlabu
  - `C_programs` ..... zdrojové kódy implementace měření
  - `chamber_programs` ..... zdrojové kódy ovládání teplotní komory
  - `scripts` ..... statistické zpracování výsledků
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF