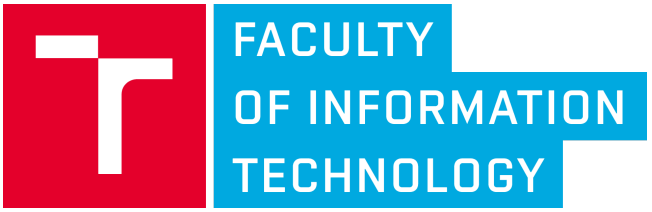

Efektivně zabezpečený inteligentní senzor

Technická dokumentace

Michal Kula, Svetozár Nosko, Ivan Homoliak

Vysoké učení technické v Brně
Fakulta informačních Technologií
Brno 2022

T A
Č R

Program Centra kompetence

Tento dokument byl vytvořen s finanční podporou TA ČR v rámci výzkumného programu TN01000077/07 (Národní centrum kompetence pro Kyberbezpečnost).

1. Obsah

2. Úvod

Jedním z nejčastějších problémů moderních technologií je vždy zabezpečení a autenticita dat, ať už se jedná o naše osobní údaje posílané na vzdálený server (např. instant messaging) nebo citlivější (tajná) data uvnitř společnosti. Avšak stále více se dává pozornost Internetu všeho (IoT), kde síť mikrokontrolérů komunikuje v rámci lokální sítě, kde společně vytvářejí ekosystém. Jedním z nejběžnějších příkladů je chytrý dům (Smart Home koncept), kde každé zařízení poskytuje nějakou funkčnost uživateli (např. ovládání světla, topení, ventilace). Informace z tohoto systému jsou většinou v lokální síti, ale může k nim být také přistupováno vzdáleně, což představuje významnou slabinu pro uživatele. Proto musí být data samotná a přístup k těmto datům dostatečně chráněny, jinak není možné důvěřovat informacím, které dostáváme nebo ještě hůře, ztratili bychom kontrolu nad naším vlastním systémem.

Tento funkční vzorek se zaměřuje na první z těchto dvou problémů a k dosažení tohoto cíle musí být data vytvořená na mikrokontroléru (a tedy získaná ze senzoru) kryptograficky podepsána předtím, než opustí zařízení. Až dosud se zpracování realizovalo bez zabezpečení na hlavním zařízení (typicky HUB nebo cloud), kde jsou všechny mikrokontroléry připojeny a odesílají zaznamenaná data, což představovalo riziko zneužití (např. podvrhnutí dat). Toto je/bylo způsobeno především kvůli nedostatečným výpočetním a paměťovým zdrojům na senzorických zařízeních, takže kryptografické algoritmy nebylo možné efektivně a rychle zpracovat. Dnes mnoho nových mikrokontrolérů poskytuje hardwarovou akceleraci pro kryptografické algoritmy, čímž významně zlepšuje zabezpečení zařízení.

3. Návrh

Důležitou součástí řešení je prôzkum existujících algoritmů, hardwarových bloků, softwarových komponentů a zjištění, zda by některý z nich mohl být použit (popřípadě upraven) pro naše potřeby. Před výběrem kryptografického algoritmu nebo jakéhokoli softwaru je znát, jaká omezení a požadavky bude mít daný hardware.

3.1. Hardware

Vzhledem k tomu, že kryptografické algoritmy jsou často náchylné k různým útokům (např. útoky postranními kanály), je nezbytné, aby před jejich bezpečným použitím řádně implementovat a ověřit. To by vyžadovalo procesor se specifickou instrukční sadou a paměť, do které se program a data načítají. Vzhledem k aktuálním možnostem se jako velmi perspektivní ukazují platformy založené na architektuře, kde je procesor spolu s FPGA[1] propojen sběrnicí na jednom čipu. Mezi nejznámější patří Xilinx Zynq, Altera SoC FPGA (dnes už Intel), Lattice ECP5 nebo Microsemi SmartFusion2. Vzhledem k dostupnosti, vlastnostem a SW podpoře byla zvolena platforma Xilinx Zynq.

[1] FPGA (Field-Programmable Gate Array) je typ programovatelného logického zařízení, které lze nakonfigurovat pro různé účely. FPGA se skládají z mnoha jednotlivých logických bloků, které lze propojit pro tvorbu složitějších obvodů. Tyto logické bloky mohou být nakonfigurovány pro různé funkce, jako je logická operace, komparace nebo paměť. FPGA jsou často používány pro aplikace, kde potřebujete velkou paralelizaci nebo rychlou reakci na změny vstupů. Jsou také využívány pro aplikace, kde potřebujete silnou kontrolu nad hardwarovou realizací

Z pohledu návrhu/implementace algoritmů v FPGA byl zvolen přístup, kde výpočet otisku (SHA256) je akcelerován řetězením (tzv. “pipelining”). Pro výpočet samotného podpisu je využit přístup, kde byl vysyntetizován řídicí soft-procesor, který umožňuje běh jednoduchých programů.

Jedním z nejznámějších FPGA soft-procesorů je Microblaze[2], který lze snadno syntetizovat do FPGA. Dále je k dispozici kompilátor jazyka C, což přináší mnoho dalších možností, o kterých bude řeč později. Další velkou výhodou jsou různé možnosti konfigurace. Vzhledem k tomu, že většina kryptografických algoritmů se skládá převážně z matematických operací (a instrukcí), nabízí se možnost použít celočíselnou násobičku nebo „Barrel shifter“ v rámci Microblaze, kde tyto moduly mohou potenciálně přinést zlepšení výkonu (analýzu výkonu viz část 4). Bezpečnost kryptografie mimo jiné závisí také na zdroji náhodných dat. Generátor náhodných čísel (RNG) může být implementován buď softwarově, nebo hardwarově, kde v případě generátoru náhodných čísel (tzv. zdroj entropie), který generátor inicializuje, musí být implementován hardwarově. Vzhledem k tomu, že Microblaze postrádá jakýkoli další hardware pro získávání entropie, je třeba zvolit jiný přístup. Jedním z různých řešení je použití analogově-digitálního převodníku (ADC), který může číst data ze snímače a akumulovat jejich hodnoty po určitou dobu. Nakonec je zapotřebí realizovat komunikaci s vnějším světem, který lze jednoduše provést pomocí sady registrů která funguje na výměnu dat s Xilinx Zynq ARM CPU.

3.2. Software

Jak již bylo zmíněno, Microblaze ekosystém je včetně kompilátoru jazyka C, který umožňuje dobře integrovat existující kryptografický algoritmus, generátor náhodných čísel a další software pro komunikaci s externími systémy. Výkon a paměť tohoto procesoru jsou však velmi omezené a řešení musí být pečlivě implementováno právě s ohledem na tato omezení. U tohoto projektu jsme byli zaměřeni na co nejmenší spotřebu paměti a prostředků při postačujícím výkonu.

Nejdůležitější částí postupu byla volba algoritmu a velikost klíče, protože obojí výrazně ovlivňuje výkonnost a využití paměti. Nejběžnější algoritmy pro kryptografické podpisy jsou Rivest-Shamir-Adleman (RSA). a algoritmus digitálního podpisu (DSA), přesněji jeho verze s eliptickou křivkou (ECDSA). Na základě studie[3] je výkonnost generování podpisu ECDSA rychlejší než RSA při použití kratší délky klíče, což také výrazně snižuje spotřebu paměti. Současná minimální doporučená velikost klíče[4]pro bezpečnou eliptickou křivku podpisů je 256 bitů.

Existuje mnoho existujících implementací ECDSA v knihovnách třetích stran, například OpenSSL, Boring SSL. nebo MbedTLS, ale většina z nich má vysoké paměťové nároky a nebylo by ji možné realizovat v Microblaze. MbedTLS je však velmi dobře konfigurovatelná a její velikost bylomožno minimalizovat tak, aby se vešla do paměti 64 KB. což by bylo pro naše potřeby ideální. MbedTLS a jeho implementace ECDSA vyžadují přítomnost generátoru náhodných čísel a jeho nasazení pro generování skutečně náhodných podpisů. Z mnoha možností se ukázalo, že generátor náhodných čísel PCG je nejvhodnější volbou, protože má skvělou statistickou kvalitu, velkou periodu a hlavně je výpočetně rychlý a využívá velmi malou část paměti.

---

[2] MicroBlaze je softwarový procesor vytvořený společností Xilinx. Je to malý, modulární a flexibilní 32bitový procesor, který lze implementovat pomocí FPGA nebo CPLD. MicroBlaze má velmi nízkou spotřebu energie a může být použit pro mnoho různých aplikací, jako jsou automatizace, zabezpečení, zpracování signálu a kontrola motoru. Má podporu pro mnoho operačních systémů, včetně FreeRTOS, Linuxu a Petalinuxu.

[3] http://www.cari-info.org/Actes-CARI-2020/21–S4CS-12.pdf

[4] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

4. Implementace

Za účelem rychlého vývoje byla použita vývojová deska Xilinx Zynq ZC702 (viz. obrázek 1). Xilinx Zynq ZC702 je systém na čipu obsahující integrovaný mikroprocesor a FPGA. Zynq ZC702 se skládá z dvou částí: procesoru ARM Cortex-A9 a FPGA Xilinx 7 serie. Deska obsahuje 1GB DDR3 RAM, 16MB flash paměti, 10/100/1000 Ethernet, USB, HDMI, JTAG, a další rozhraní. Může být použita pro aplikace v oblasti průmyslu, automobilového průmyslu, počítačového vidění, komunikací a dalších.

Hardwarová i softwarová implementace byla provedena v prostředí Vivado Design Suite spolu s aplikací Vitis IDE verze 2020.1. V průběhu procesu implementace byl také použit Linux pro realizaci ovládací aplikace a odesílání dat na cloud.

Obrázek 1: Fotodokumentace vzhledu funkčního vzorku. Na obrázku je funkční vzorek zabezpečeného senzoru založen na platformě Xilinx Zynq-7000 SoC ZC702 Evaluation Kit.

4.1. Hardware (FPGA)

Výslední návrh FPGA sestává vyjma MicroBlaze z hardwarových bloků, které realizují komunikaci s Linuxem, zdroj entropie:

- AXI Interconnect realizuje přístup k periferiím z Microblaze a Zynq ARM.
- Bloková paměť RAM (BRAM) pro ukládání vstupních dat a odesílání výstupních dat.

- Analogově-digitální převodník (ADC) s teplotním čidlem pro sběr náhodné entropie.
- AXI Timer pro vyhodnocování doby běhu kódu různých konfigurací Microblaze (viz část 4).

Po provedení řádného vyhodnocení se konečná konfigurace systému Microblaze lišila od jeho výchozí konfigurace zvýšením jeho vstupního taktu na 166 MHz, přidáním barrel shifter, celočíselné násobičky a zapnutím funkce cache pro predikci větvení (jedna z optimalizací pro vykonávaní instrukcí v MicroBlaze). Tato konfigurace se ukázala jako ideální pro naše potřeby, protože výrazně zvýšila výkonnost provádění kódu ECDSA (viz část 4).

4.2. Software

Následně po návrhu byl vytvořen prototyp a ověřen na testovacích datech, byl pro něj vytvořen dizajn v FPGA, návrh byl syntetizován a exportován pro použití v prostředí Vitis IDE. V prostředí Vitis IDE byla implementována softwarová část.

.Jakmile byl návrh ověřen v prostředí Vivado Design Suite, byl vysyntetizován a exportován pro použití v prostředí Vitis IDE, kde byla implementována softwarová část tohoto projektu. Exportovaný projekt byl použit k vytvoření projektu platformy nakonfigurované pro Microblaze a byla vytvořena prázdná aplikace.

Především bylo důležité zjistit, kolik paměťového prostoru (prostor pro kód) knihovna MbedTLS skutečně využívá. Po výběru všech potřebných komponent včetně ECDSA, SHA256 a eliptické křivky v konfigurační hlavičce MbedTLS a zkopírování všech potřebných souborů do aplikace byla výsledná velikost binárky uspokojivá (viz část Vyhodnocení). Poté byl přidán kód pro vložení privátního klíče a generování podpisů ECDSA.

Jakmile byla shromážděna všechna potřebná data a stanovena konečná konfigurace systému Microblaze, byl integrován generátor PCG spolu s vlastním řešením pro sběr entropie. Toto řešení využívalo ADC převodník přidaný během návrhu a každý údaj o teplotě byl použit k počítání výskytů jednotkových bitů. Pokud byl výskyt jedniček větší než počet výskytů nul, byl přičten k výsledné entropii, jinak byla hodnota nulového bitu zachována. Po každém počítání výskytů byla celá hodnota posunuta o jeden bit doleva, dokud nebyly shromážděny všechny požadované bity. Entropie pak byla použita k inicializaci generátoru náhodných čísel. Struktura uchovávající aktuální stav generátoru a funkce zpětného volání pro generování dalšího stavu byly předány podpisové funkci ECDSA, která je v případě potřeby může použít.

Výkonnostní zlepšení systému Microblaze oproti jeho výchozí konfiguraci bylo výrazné, ale přesto byla přidána další vrstva, aby se snížila četnost provádění algoritmu podpisu. Algoritmus Merkeleova stromu byl implementován za účelem agregace hashovaných dat senzorů do binární stromové struktury, kde jsou dva podřízené uzly spojovány pomocí hashování s jejich rodičovským uzlem, dokud není dosaženo kořenového uzlu. Výpočet hodnoty hash kořenového uzlu se provádí buď na vyžádání, nebo když je strom plný a nelze do něj uložit další data. Kořenový hash je poté podepsán a spolu s podpisem předán do výstupního registru, který signalizuje subsystému Linux, že jsou k dispozici nová data.

Na adrese kořenového uzlu je pak podepsán a spolu s podpisem, identifikací transakce a počtem listů v aktuálním stromu předán do výstupního bufferu (Transaction queue). Tento buffer je na straně ARM namapován jako registrové pole o velikosti 128B. Vyčtením dat z nejnižší adresy dojde k přechodu na

další položku fronty. FPGA dále pomocí registrů publikuje Public key elektronického podpisu a typ algoritmu použitého pro výpočet hash a podpisu. Pro účely časového ukotvení dat je z ARM do FPGA části také pomocí registrů předáván hash aktuálního časového razítka, který je následně zanesen do hlavičky dat ze senzoru ještě před výpočtem jeho hashe.

5. Vyhodnocení

Fáze vyhodnocení probíhala iterativně, kde docházelo k modifikacím nastavení procesoru Microblaze a jeho vnitřních komponent, taktovací frekvence FPGA.

5.1. Výkon

Pro kompilaci software byl využit dodaný kompilátor, který je součástí vývojového prostředí Vitis. Konkrétně je o kompilátor jazyka C/C++, který využívá překladač GCC a následně generuje kód pro instrukční sadu Microblaze. Software pro podepisování byl kompilován s optimalizacemi (O2). Následující tabulky ukazují různé konfigurace a jejich příslušné průměrné časy vykonávaní pro algoritmus ECDSA s křivkou SECP256K1 pomocí knihovny MbedTLS.

Frekvence MicroBlaze [MHz]	Doba generování podpisu [ms]
50	1560
100	792
166	474

Tabulka 1: Časy generování podpisu ECDSA pro křivku SECP256K1 s různými vstupními taktovacími frekvencemi.

Modul MIcroblaze	Doba generování podpisu [ms]
32-bit integer multiplier (MUL32)	225
64-bit integer multiplier (MUL64)	487
MUL32 and Basic FPU	234
MUL32 and Extended FPU	252
MUL32 and Barrel Shifter (BSH)	94
MUL32, BSH and Integer Divider (IDIV)	94
MUL32, BSH and Machine Status Register (MSR)	96
MUL32, BSH and Branch Target Cache (BTC)	89
MUL32, BSH, BTC, Data and Instruction caches	91

Tabulka 2: Doba generování podpisu ECDSA s dalšími funkcemi (vstupní hodiny 166 MHz).

Podle tabulky 1 je taktovací frekvence v lineárním vztahu k době provádění (zdvojnásobení vstupní taktovací frekvence z 50 na 100 MHz zkrátilo dobu provádění téměř o polovinu). To je očekávané chování, protože s rychlejším taktem je procesor schopen zpracovat více instrukcí. Vliv přidání různých funkcí do Microblaze je uveden v tabulce 2. Algoritmus ECDSA je založen na matematickém konceptu modulární exponenciace a očekávaně se při zapnutí celočíselné násobičky zvýší výkon. Na druhé straně absence operací s plovoucí desetinnou čárkou nemá žáden účinek – taky jde o očekávané chování. Další významný dopad na výkon mělo přidání Barrel Shifter a také drobné zlepšení zapnutím cache pro predikci větvení. Všechny ostatní funkce neměly na dobu provádění podpisového algoritmu ECDSA žádný vliv. Knihovna MbedTLS podporuje více eliptických křivek dostupných pro použití v ECDSA. Jediné schůdné možnosti jsou 256bitové klíče, protože jsou dostatečně bezpečné, aby se daly použít, ale nezabírají tolik místa v paměti jako klíče větších velikostí. Hodnocení výkonu různých eliptických křivek je uvedeno v následující tabulce.

Eliptická křivka	Doba generování podpisu [ms]
SECP256K1	89
SECP256R1	446
BP256R1	535

Tabulka 3: Srovnání různých podporovaných eliptických křivek.

Kromě toho byl vyhodnocen také algoritmus SHA256, který se používá v implementaci Merkelova stromu. Software byl zkompilován se zapnutou optimalizací na velikost kódu a Microblaze byl nakonfigurován pro nejlepší výkon ECDSA (takt 166 MHz, MUL32, BSH a BTC). Všechna data jsou uvedena v následující tabulce.

Algoritmus	Doba generování (μs)
SHA256	41
Merkleův strom s 8 SHA256 otisky	558
Merkleův strom s 16 SHA256 otisky	1189

Tabulka 4: Doba vykonávaní algoritmu SHA256 a Merkleova stromu.

5.2. Velikost kódu

Po vyhodnocení výkonu byla další důležitou částí optimalizace velikosti kódu. Knihovna MbedTLS byla minimalizována ponecháním pouze nezbytných funkcí. Ovšem velký vliv na paměťový prostor má volba úrovně optimalizace kompilátoru, což je ukázáno následující tabulce.

Úroveň optimalizace	Velikost kódu (B)	Doba generování podpisu [ms]
-O0	67280	198
-O1	50000	97
-O2	58936	89
-O3	68048	71
-Os	44824	101

Jak je vidět, úroveň optimalizace -O3 přináší nejlepší výkon, ale je největší spotřeba prostoru pro kód, v tomhle případě však nebyly implementovány všechny softwarové komponenty. Merkelův strom, PCG RNG a logika pro generování dat entropie zvýšily celkovou velikost balíčku ze 44824 bajtů na 56592 bajtů

6. Závěr

Cílem tohoto funkčního vzorku bylo navrhnout, implementovat a otestovat bezpečný ekosystém pro vytváření dat ze senzorů na FPGA. Je využit soft-core procesor Microblaze, který lze snadno instanciovat, upravit a syntetizovat v prostředí Vivado Design Suite. Navíc je k němu dodáván také kompilátor jazyka C, což poskytuje možnost využít existující implementace podpisových algoritmů a případně je spustit v FPGA.

V mnoha aplikacích je algoritmus ECDSA již řadu let široce používán pro jeho vlastnosti jako je bezpečnost, výkon a spotřeba paměti.

Po provedení vyhodnocení ECDSA v Microblaze lze konstatovat, že je možné generovat podpisy dostatečně efektivně, aby mohly být použity v aplikacích v reálném čase. Použití Merkelova stromu snížilo potřebné množství podpisů ECDSA, což mělo velký vliv na použitelnost tohoto systému. Všechna vylepšení, která byla provedena v průběhu implementace, zvýšila výkon 15krát, přičemž celková velikost softwarového balíčku zůstala pod 64 KB.

Přestože byl projekt úspěšný, je možné v budoucnu provést některá další vylepšení. Po provedení důkladného profilování ECDSA lze vytvořit vlastní blok pro hardwarovou akceleraci speciálně pro časově nejnáročnější část kódu. To může být potenciálně užitečné v aplikacích se senzory produkujících řádově více datových paketů jako jsou například radary nebo obecně senzory v průmyslových aplikacích.