Název projektu: Nástroje pro verifikaci bezpečnosti kryptografických zařízení s využitím AI

Identifikační kód: VJ02010010

Odborná zpráva - Etapa 10

Období: 06/2023 - 12/2023

Příjemce:	Vysoké učení technické v Brně Antonínská 548/1 Brno 60190
Jméno hlavního řešitele:	doc. Ing. Jan Hajný, Ph.D. Tel.: +420541146961 Email: hajny@vut.cz
Další účastník 1:	Masarykova univerzita Žerotínovo náměstí 617/9 Brno 602 00
Jméno řešitele:	doc. RNDr. Petr Švenda, Ph.D. Tel.: +420549491878 Email: svenda@fi.muni.cz
Další účastník 2:	České vysoké učení technické v Praze Jugoslávských partyzánů 1580/3 Praha 160 00
Jméno řešitele:	Dr-Ing. Martin Novotný, Ph.D. Tel.: +420224358715 Email: martin.novotny@fit.cvut.cz

Obsah

Číslo a název aktivity: Etapa 10	
Období řešení:	06/2023 - 12/2023
Cíl aktivity:	Návrh a implementace sady otevřených nástrojů pro analýzu stávajících certifikačních reportů (mapování certifikovaných zařízení na zveřejněné zranitelnosti pro podporu automatizovaného analytického zpracování s využitím strojového učení (AI)). Implementace SW modulu pro analýzu implementací kryptografických algoritmů.
Krátký popis řešení:	Volba a analýza vhodných nástrojů a kryptografických primitiv s aplikačním garantem. Identifikace scénářů využití. Návrh, implementace a vyhodnocení první iterace nástrojů.
Řešitel zodpovědný za realizaci:	Petr Švenda (Masarykova univerzita).
Stav plnění aktivity:	Splněno
Výstupy:	1x shrnující zpráva (CZ), 1x příloha článek popisující mapování certifikátů zařízení a zranitelností (v recenzním řízení, EN), 3x GitHub repositář nástroje pyecsca, dcp-glv, JCMathLib.
Výsledky:	V rámci této etapy došlo k přípravě a zaslání článku analyzujícího certifikační reporty do recenzního řízení (Computers&Security), v době odevzdání zprávy zatím nemáme výsledné vyjádření.
Doložení splnění aktivity:	Odborná zpráva – Etapa 10
Shrnutí:	Všechny podmínky pro přechod do další aktivity byly splněny.

1 Úvod

V rámci etapy #10 (07/2023 - 12/2023) jsme navrhli a implementovali automatické párování zranitelností z databáze NIST NVD vůči zařízením certifikovaným v rámci Common Criteria a FIPS140 včetně vyhodnocení vlivu úrovně certifikace na četnost a závažnost nalezených zranitelností.
Pro analýzu kryptografických implementací na bázi ECC bez přístupu ke zdrojovému kódu (black-box, např. čipové karty a jednočipy) jsme dále rozvíjeli projekt pyecsca, který využívá postranních kanálů pro detekci rozsáhlého množství možných implementačních rozhodnutí na pozorované chování cílové implementace.
Dále proběhla analýza existujících implementací na čipových kartách (status-keycard, JCFROST, JCMathLib) pro detekci úniku informací postranními kanály. Detekované úniky byly analyzovány z hlediska praktického využití zranitelnosti a byly navrženy úpravy kódu snižující únik informací. Cílem prací je vyvinout metodiku podpořenou praktickými nástroji pro detekci, opravu a ověření přiměřené odolnosti testované implementace vůči útokům postranními kanály.
Tato zpráva shrnuje v českém jazyce nejdůležitější zjištění, pro plný rozsah analýz doporučujeme přílohy v anglickém jazyce. Část prací úzce souvisí i s výsledky dosaženými v rámci etapy #9, pro omezení překryvu popisu nástrojů uvádíme potřebné informace na jednom místě a doporučujeme tedy přečíst i související výsledky etapy #9.

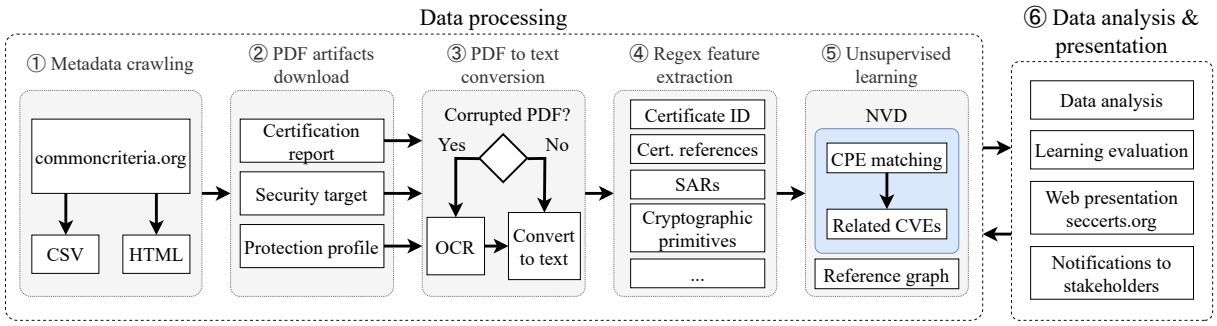
Veškeré plánované činnosti v rámci Etapy 10 byly úspěšně dokončeny a průběžné výsledky byly prezentovány aplikačnímu garantovi (Národní úřad pro kybernetickou a informační bezpečnost (NÚKIB)) v těchto termínech:

- **22.5.2023, Brno.** Předběžné představení plánů chystaných v rámci etapy.
- **18.9.2023 dopoledne, Praha.** Kontrolní den, prezentace dosažených výsledků.
- **18.9.2023 odpoledne, Praha.** Rozšířená diskuze formou workshopu.
- **6.12.2023, Praha.** Představení výsledků v oblasti šifrovaných disků, analýzy úniku časovým postranním kanálem knihovny pro čipové karty a reverzní inženýrství implementací kryptografie eliptických křivek. Diskuze prací plánovaných pro 2024.

2 Návrh a implementace sady otevřených nástrojů pro analýzu stávajících certifikačních reportů. Mapování certifikovaných zařízení na zveřejněné zranitelnosti pro podporu automatizovaného analytického zpracování s využitím strojového učení

2.1 Nástroj *seccerts* pro analýzu certifikatů a zranitelností

V rámci této sekce se zaměřujeme na analýzu certifikačního schématu Common Criteria. Zařízení kritická pro zajištění informační bezpečnosti procházejí v rámci tohoto schématu nákladným certifikačním procesem, jehož vliv na samotnou bezpečnost hodnoceného produktu je nejasný. Zároveň není možné snadno zjistit, která certifikovaná zařízení trpí na existující zranitelnosti, a to kvůli vysokému množství artefaktů souvisejících s certifikací a nejasným vztahem mezi samotnými certifikovanými produkty. V této sekci se snažíme výše zmíněný problém řešit automatizovanou analýzou certifikačních artefaktů.



Obrázek 1: Systémový diagram nástroje sec-certs

V rámci projektu jsme navrhli a implementovali nástroj, který umožňuje plně automatické zpracování pdf, html, a csv dokumentů z portálu `commoncriteriaportal.org`. Celý proces se sestává z několika kroků přehledně zobrazených na obrázku 1. Takto získáváme rozsáhlé informace o všech certifikovaných zařízeních. Zároveň jsme navrhli a natrénovali model strojového učení bez učitele, který automaticky propojuje databázi certifikovaných zařízení s databází existujících zranitelností (National Vulnerability Database) se značnou přesností. Tímto je možno získat konkrétní vhled do vlivu certifikačního procesu na samotnou bezpečnost produktů, jakož je možné sledovat konkrétní zařízení pro možné zranitelnosti.

2.1.1 Zranitelnosti v ekosystému certifikovaných zařízení

Výše uvedená metodika nám pomohla identifikovat druhy zranitelností (dle taxonomie Common Weakness Enumeration), která postihují certifikovaná zařízení nejčastěji. Podařilo se nám ukázat,

že typy zranitelností postihující certifikovaná zařízení se nikterak neliší od typů zranitelností, která zasahují open-source produkty. Zároveň se nám podařilo zmapovat, kdy během životního cyklu produktu dochází ke zveřejňování zranitelností vzhledem k datu certifikace. Tato pozorování jsou zachycena v tabulce 1 a na obrázku 2.

CWE-ID	CWE name	# CVEs
CWE-119	Buffer overflow	892
CWE-20	Improper Input Validation	487
CWE-200	Sensitive information exposure	349
CWE-264	Access control error	316
CWE-787	Out-of-bounds Write	297
CWE-125	Out-of-bounds Read	208
CWE-399	Resource Management Errors	180
CWE-79	Cross-site Scripting	148
CWE-416	Use After Free	122
CWE-362	Race Condition	115

Tabulka 1: Nejčastější typy zranitelností (CWE (Common Weakness Enumeration)) identifikované ve 3114 zranitelnostech (CVE (Common Vulnerabilities and Exposures)) ovlivňujících 362 certifikovaných zařízení.

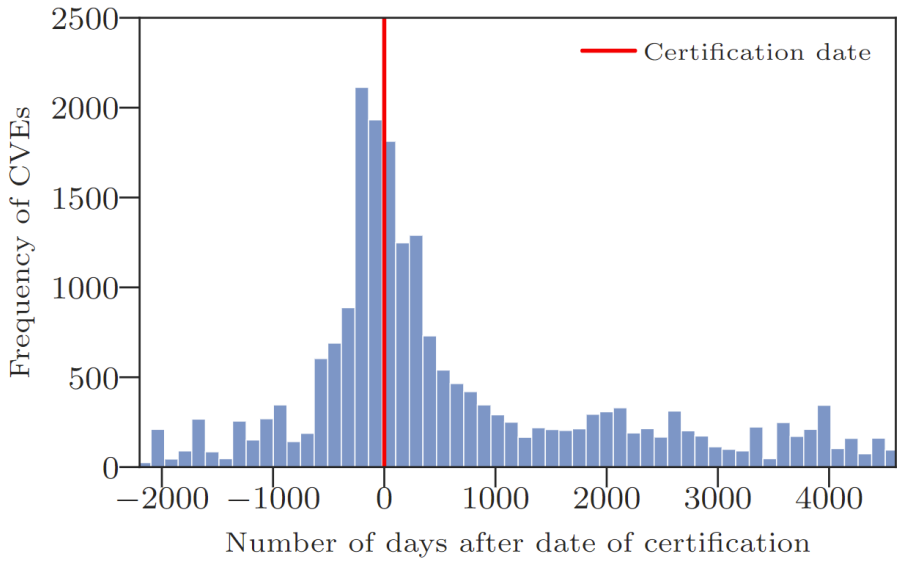
Bližší pohled na vztahy mezi závažností zranitelností a implementovanými bezpečnostní požadavky (SAR, Security Assurance Requirements) v rámci schématu Common Criteria umožnil identifikovat bezpečnostní opatření s nejvyšším pozitivním dopadem na bezpečnost produktu. Námi provedená korelační analýza ukazuje na aspekty certifikace, které jsou asociovány s menším počtem zranitelností, respektive s jejich menší závažností. To v důsledku ukazuje, které aspekty certifikace mohou zlepšit bezpečnost certifikovaného produktu. Tyto jsou uvedeny v tabulce 2.

2.1.2 Zpřístupnění nástroje

Námi vyvíjený nástroj je již on-line dostupný ze `https://seccerts.org`, případně z repozitáře na GitHubu. Zároveň je již aktivně nasazen úřadem NÚKIB, kde pomáhá analytikům (Josef Pospíšil). V rámci projektu probíhá pravidelná aktualizace sbíraných dat. Náš nástroj každý týden posbírá všechny dostupné artefakty certifikačního schématu Common Criteria a sleduje tak vývoj ekosystému v čase.

2.1.3 Zpracování přirozeného jazyka

V současnosti se zaměřujeme na hlubší automatizované pochopení obsahu certifikačních artefaktů metodami zpracování přirozeného jazyka (NLP). To nám umožní automaticky porozumět



Obrázek 2: Data zveřejnění zranitelností ve vztahu k datu certifikace postiženého zařízení.

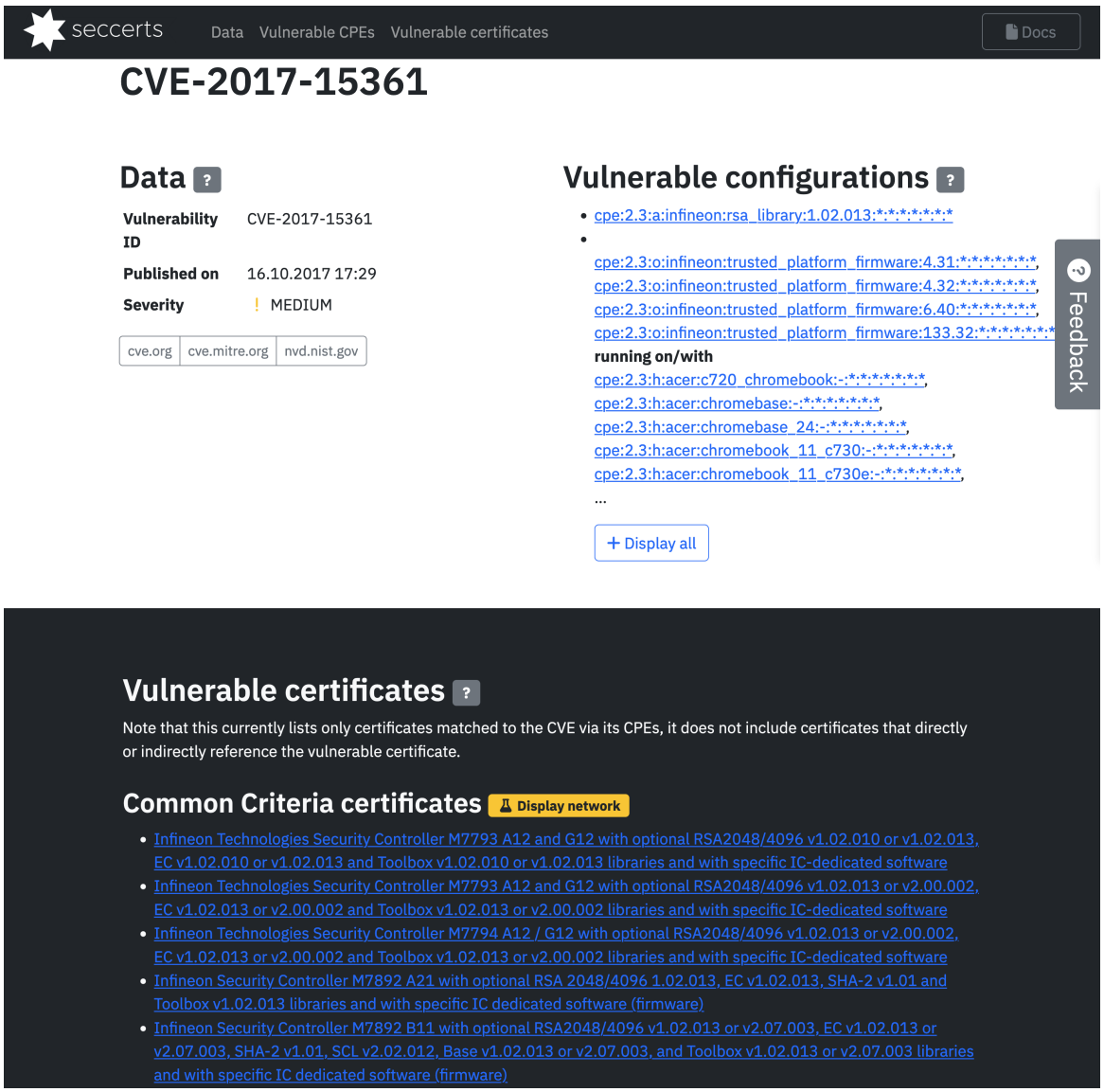
významu referencí mezi certifikáty, což přinese hluboký vhled do ekosystému certifikovaných zařízení a umožní identifikovat kritické zařízení v tomto ekosystému, co se celkového dopadu na bezpečnost týče. Výše uvedené také umožní rekonstruovat úplný graf certifikovaných zařízení a jejich vzájemných referencí. To povede ke zlepšené schopnosti monitorovat existující zranitelnosti. Analytik pracující s nástrojem bude moci expertní analýzou ověřit, zda zranitelnost nalezená v certifikovaném zařízení zasahuje nějaká další zařízení, která na zranitelné zařízení odkazují ve svých certifikačních dokumentech. Zároveň pracujeme na vývoji systému pro identifikaci segmentů dokumentů, které jsou relevantní vzhledem k libovolným uživatelským dotazům. Tím bude možno dosáhnout funkcionality obdobné s velkými jazykovými modely (LLM, například Chat-GPT), avšak ve značně specifické doméně certifikačních dokumentů.

2.1.4 Ukázka operačního využití

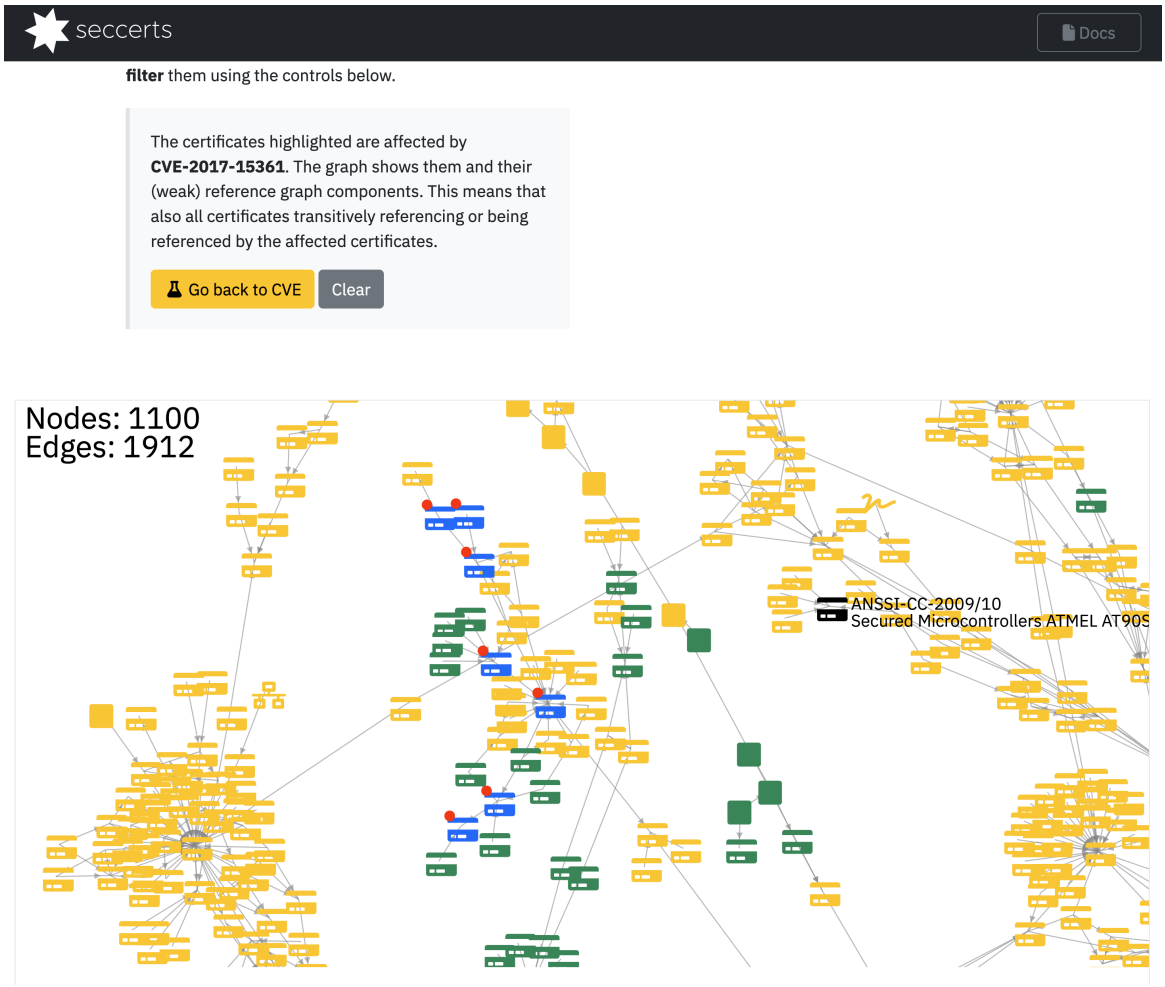
Níže uvádíme příklad využití našeho nástroje z pohledu analytika monitorujícího certifikovaná zařízení na existující zranitelnosti. Z webové adresy `seccerts.org/vuln` je možné prozkoumat certifikáty, které trpí na konkrétní zveřejněnou zranitelnost, viz obrázek 3. Dále je možné tato certifikovaná zařízení zobrazit v kontextu sítě všech zařízení a prozkoumat tak možný dopad zranitelnosti na ostatní části ekosystému, viz obrázek 4.

2.1.5 Publikační činnost

Vytvořený nástroj pravidelně prezentujeme odborné veřejnosti (workshop pro NÚKIB 05/2023, workshop RIA Estonsko 05/2023 a 06/2023, prezentace NÚKIB 09/2023) a sbíráme zpětnou vazbu a nápady pro další rozšíření nástroje. Dosažené výsledky jsme sepsali jako časopisecký článek s názvem, *sec-certs: Examining the security certification practice for better vulnerability mitigation*, který je nyní v průběhu recenzního řízení.



Obrázek 3: Ukázka hledání zranitelných certifikátů pro konkrétní zranitelnost na webové prezentaci `seccerts.org/vuln`.



Obrázek 4: Ukázka zobrazení grafu certifikovaných zařízení postižených konkrétní zranitelností (zvýrazněno červenou tečkou) v kontextu sítě všech certifikovaných zařízení.

variable	*number of CVEs*		*Average CVE base score*		*variable characteristics*	
	ρ	p-value	ρ	p-value	support	domain range
EAL	-0.02	1.64e-01	-0.28	4.24e-05	2927	14
ALC_CMS	-0.07	7.34e-05	-0.35	2.55e-07	2583	5
ALC_CMC	-0.08	3.54e-05	-0.35	3.05e-07	2578	5
AVA_VAN	-0.08	1.29e-05	-0.34	6.47e-07	2606	4
ATE_COV	-0.03	8.50e-02	-0.33	8.41e-06	2419	3
ATE_IND	-0.05	1.51e-03	-0.29	1.54e-06	3321	3
ADV_FSP	-0.06	4.19e-04	-0.24	6.00e-05	3371	6
ASE_OBJ	-0.07	1.33e-03	-0.14	4.46e-02	2080	2
ASE_REQ	-0.07	2.94e-04	-0.12	6.01e-02	2120	2
ALC_FLR	0.06	9.94e-01	0.14	9.52e-01	1684	3

Tabulka 2: Korelační koeficienty (počítáno metodou Spearman's rank correlations) mezi jednotlivými SARy (případně EAL) a počtem, respektive průměrnou závažností zranitelností. Sloupec "support"indikuje velikost datové sady použité pro výpočet. Zelená barva indikuje existující korelaci na p-hodnotě < 0.01.

3 SW moduly pro analýzu implementací kryptografických algoritmů

V rámci této sekce poskytujeme popis nástrojů a databází se zaměřením na bezpečnost implementací vyvíjených nebo rozvíjených v rámci prací na projektu v letošním roce. Uvedený popis pokrývá základní principy fungování nástrojů a ukázky produkovaných výstupů. Pro detailnější informace odkazujeme na příslušnou dokumentaci, reporty a publikované články.
Pokrýváme následující nástroje:

- **pyecsca**: útoky postranním kanálem na implementace kryptografie eliptických křivek se zaměřením na reverzní inženýrství black-box implementací (sekce 3.1).
- **JCProfilerNext**: analýza rychlosti běhu, paměťové náročnosti a používaných balíků JavaCard aplikací. Analýza probíhá přímo na cílové kartě, umožňuje detekovat části časově náročného kódu (rychlostní optimalizace) a detekci časově nekonstantních částí kódu (sekce 3.2).

V rámci projektu jsou nástroje vyvíjeny či upravovány dle potřeb aplikačního garanta a průběžně konzultovány na osobních setkáních.

3.1 *pyecsca*: reverse-engineering black-box implementací kryptografie eliptických křivek

Kvůli složité a vrstvené povaze kryptografie eliptických křivek (ECC) existuje mnoho možností implementace, ze kterých si vývojář může vybrat, aby získal funkční a interoperabilní implementaci. Mezi možnosti implementace patří například křivkový model, souřadnicový systém, sčítací vzorce, skalární multiplikátor nebo detaily nižší úrovně, jako je algoritmus násobení nad konečnými tělesy.
Útoky postranními kanály na ECC často předpokládají útočníka typu white-box, který má podrobné znalosti o detailech implementace. To vytváří mezeru mezi teoreticky možnými útoky a skutečným útočníkem, který má k cíli často přístup pouze jako black-box. Pokud však dokáže útočník potřebné detaily implementace získat (například s využitím reverzního inženýrství), může docházet k praktickým útokům i na certifikované implementace ECC, jak ukazuje například [5].
Pro potřeby analýz jsme vyvinuli sadu otevřených nástrojů pro automatizované reverzní inženýrství black-box implementací kryptografie eliptických křivek nazvanou `pyecsca`. Sada obsahuje tři kategorie metod reverzního inženýrství založených na využití:

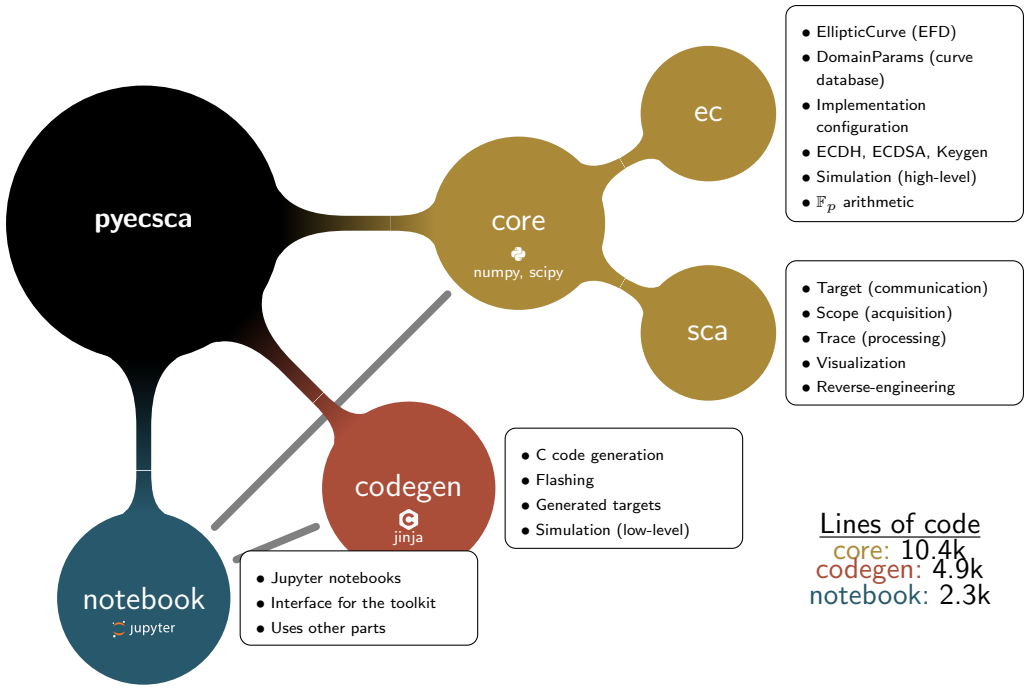
1. **Funkčnosti útoků**. Metody založené na útocích fungují tak, že otáčejí útoky postranním kanálem. Místo aby předpokládaly znalost implementace a zaměřovaly se na zisk tajného klíče, tak používají známou hodnotu (tajného) klíče a zaměřují se na zisk informace o implementaci.

2. **Strukturálních vlastností**. Strukturální metody využívají strukturu a vlastnosti přítomné ve stopách postranního kanálu cíle k reverznímu inženýrství implementace.

3. **Behaviorálních vlastností**. Behaviorální metody pozorují chování implementace — chyby nebo chybné výstupy za neočekávaných či okrajových podmínek, aby z nich vyvodily potřebné informace o implementaci.

I když je nástroj pyecsca zaměřený na reverse-engineering, obsahuje širokou funkcionalitu na útoky postranními kanály a práci s eliptickými křivkami:

- Výčet milionů možných konfigurací implementace ECC.
- Simulace (na úrovni operací na konečných tělesech) a provádění generování klíčů, ECDH a ECDSA.
- Generování C kódu (pro ARM mikrokontrolery, či x86 zařízení) implementace ECC v libovolné konfiguraci.
- Odběrová analýza pomocí osciloskopů PicoScope/ChipWhisperer.
- Možnosti zpracování získaných odběrových křivek, např. zpracování signálu, filtrování, průměrování, řezání, zarovnání.



Obrázek 5: Přehled architektury **pyecsca** toolkitu.

- Komunikace přes PCSC/LEIA s implementací ECC na čipové kartě.

V letošním roce došlo hned k několika rozšířením nástroje pyecsca:

- Přidání GPU-akcelerovaných implementací algoritmů na analýzu postranních kanálu (jako například počítání korelačního koeficientu apod.) výrazně zrychlující celý proces.
- Napojení na CPU simulátor Rainbow (`https://github.com/Ledger-Donjon/rainbow`), což umožňuje simulovat odběrovou analýzu na úrovni jednotlivých CPU instrukcí i s výběrem modelu úniku informace (leakage model).
- Přidání nových skalárních multiplikátorů a tedy zvětšení prostoru pokrytých možných implementací.
- Přidání několika automatizovaných technik na reverzní inženýrství implementací, na základě článku [6].

Více informací i návody na použití toolkitu jsou dostupné na `https://neuromancer.sk/pyecsca/`. Výstupem tohoto výzkumu je i článek, který aktuálně připravujeme pro zaslání k recenznímu řízení bezpečnostní konference.

3.1.1 GPU-akcelerovaná analýza

Nástroj pyecsca obsahuje GPU-akcelerované implementace základních algoritmů na analýzu postranních kanálů: `sum`, `average`, `variance`, `standard_deviation` a `pearson_correlation`. Algoritmy `sum`, `average`, `variance` a `standard_deviation` počítají paralelně na odběrových křivkách

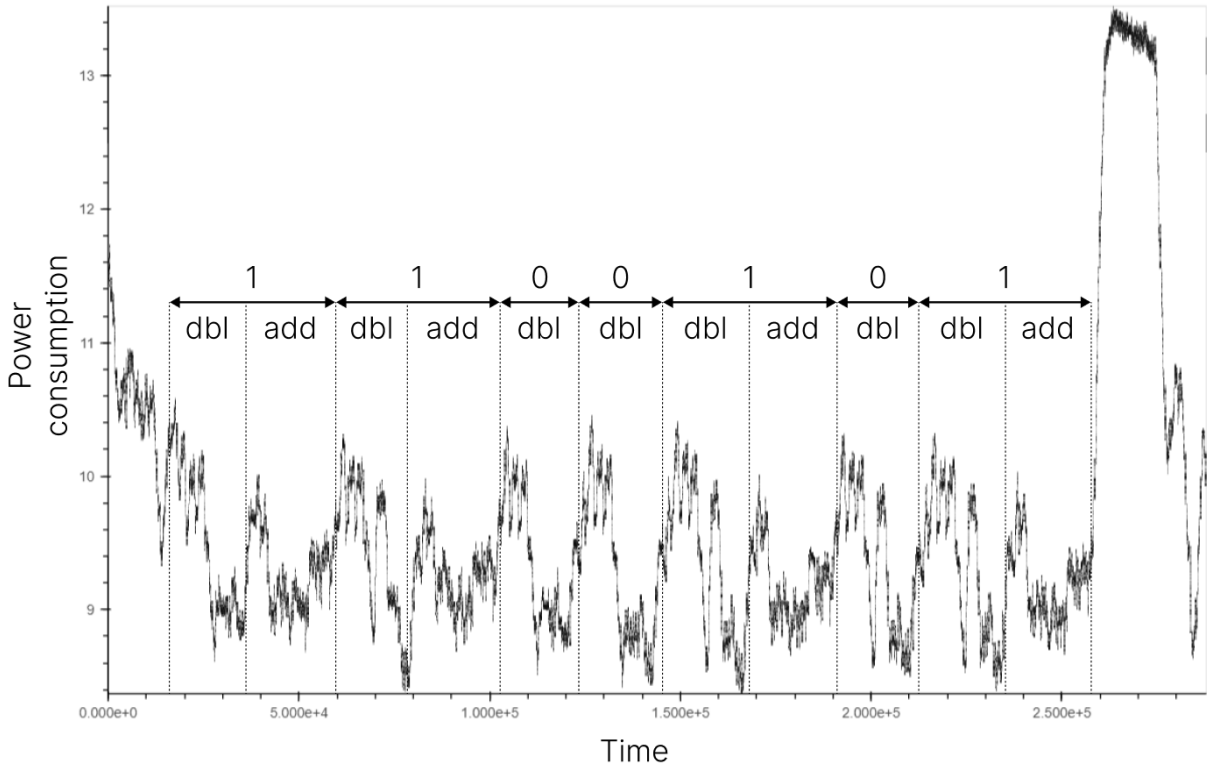
po vzorcích a vyprodukují jednu výslednou odběrovou křivku. Algoritmus `pearson_correlation` slouží při útocích korelační analýzou (CPA). Jeho vstupem je vektor očekávaných mezivýsledků (případně na ně aplikovaného modelu jako Hammingova váha) a dataset odběrových křivek. Algoritmus následně paralelně spočítá korelační koeficienty a vyprodukuje korelační křivku. Algoritmy jsou implementovány pomocí Numba frameworku a podporují tak všechna zařízení kompatibilní s CUDA rozhraním pro GPU.
Při rychlosti moderních CUDA zařízení (a tedy grafických karet) je u zmíněných algoritmů hlavním omezením velikost paměti grafické karty a rychlost, kterou se data do ní dostanou. Toto omezení jsme adresovali implementací kouskování a streamování zpracovávaných dat. Dataset odběrových křivek může být větší než velikost paměti na grafické kartě, algoritmus odběrové křivky automaticky rozdělí na části, které se do paměti vejdou, a následně spojí jejich zpracované výsledky.

3.1.2 CPU simulace

Rainbow (`https://github.com/Ledger-Donjon/rainbow`) je CPU simulátor založený na Unicorn frameworku, který se zaměřuje na postranní kanály. S jeho pomocí jsme do nástroje pyecsca přidali simulaci ARM Cortex-M4 mikro-kontrolérů s vygenerovanou C implementací ECC. Uživatel si tak může vybrat konfiguraci implementace ECC, nechat si ji zkompilovat, vybrat leakage model a následně simulovat její provádění s přesností jednotlivých instrukcí. Na obrázku 6 je vidět jedna simulovaná odběrová křivka skalárního násobení, při které byl použit leakage model Hammingovy váhy registrů.

3.1.3 Zero-value-point útoky

Charakter útoků postranními kanály na ECC závisí na použitém algoritmu pro skalární násobení a případně na použité eliptické křivce. Toho využívá nástroj `pyecsca` pro rozpoznání použitých algoritmů. Ukazuje se však, že volba algoritmů a křivky ovlivňuje nejenom průběh ale dokonce i úspěšnost některých útoků postranními kanály jako jsou útoky nulovým bodem (ZVP/RPA-/EPA) [1]. U těchto útoků útočník musí vyřešit matematický problém, jehož obtížnost se odvíjí od implementovaných algoritmů a křivky.
V rámci našeho výzkumu ECC algoritmů jsme proto analyzovali formule pro základní operace na eliptické křivce vzhledem k útokům nulovým bodem. Klasifikovali jsme každou volbu formule a eliptické křivky (přes 14 000 známých kombinací) pro skalární násobení jako zranitelnou nebo odolnou podle odpovídající obtížnosti útoku nulovým bodem. Například kombinace křivky P-256 a formule navrhnuté autory Renes, Costello, Batina pro bezpečné ECC implementace [4] se ukazuje jako zranitelná. Oproti tomu Curve25519 je odolná při jakékoliv volbě formulí. Naše klasifikace je prováděna automaticky a lze ji doplnit o jiné křivky nebo formule. Výstup naší klasifikace je k dispozici v repozitáři `https://github.com/crocs-muni/dcp-glv`. Tuto klasifikaci lze použít v rámci analýzy ECC implementací z hlediska postranních kanálů nebo může pomoci při návrhu těchto implementací.



Obrázek 6: Simulovaná odběrová křivka skalárního násobení se sedmibitovým skalárem.

Při práci na útocích nulovým bodem jsme vytvořili nový typ těchto útoků na skalární násobení používající Bitcoin křivku, ZCash křivku a jiné jim podobné. Jádrem této práce byl teoretický výzkum matematických vlastností těchto křivek, který má ale praktický dopad pro útočníka používajícího nulové body. Ukázali jsme, že útočník dokáže získat dvakrát více bitů soukromého klíče použitím útoku nulového bodu na tyto křivky než na jiné křivky. Výstupem tohoto výzkumu a popsané klasifikace formulí je článek, který připravujeme k zaslání do recenzního řízení bezpečnostní konference.

3.1.4 Další informace

Detailní informace a návody na použití toolkitu pyecsca jsou dostupné na `https://neuromancer.sk/pyecsca/`, zdrojové kódy v repozitáři `https://github.com/J08nY/pyecsca`. Klasifikaci křivek naleznete v repozitáři `https://github.com/crocs-muni/dcp-glv`.

3.2 *JCProfilerNext*: analýza časových závislostí JavaCard aplikací

Návrh a funkčnost nástroje JCProfilerNext byla detailně popsána v roční zprávě pro rok 2022. V roce 2023 bylo doplněno rozšíření pro statickou analýzu JavaCard projektů a pokročilejší vizualizaci získaných časových a paměťových měření.
Nástroj byl také v roce 2023 využit pro analýzu ekosystému JavaCard appletů pro kryptografické čipové karty (viz. zpráva Etapa 9) s výsledky a související analýzou publikovanou v článku *The adoption rate of JavaCard features by certified products and open-source projects* prezentovaném na konferenci CARDIS 2023 [8].
Na základě konzultací s aplikačním garantem byl nástroj v roce 2023 využit pro analýzu úniku informace postranními kanály dvou JavaCard appletů – status-keycard (kryptoměnová peněženka) a JCFROST (vícestranné podepisování využívající knihovnu JCMathLib). Analýza slouží jako demonstrace využití nástroje na reálných aplikacích i demonstrace proveditelných úprav kódu snižující únik informace postranními kanály.

Příklad 1: Detekce úniku informace při derivaci klíče status-keycard

JavaCard applet *status-keycard* [2] umožňuje hierarchické odvození klíčů za účelem podpory standardu BIP32 [7]. Odvození podřízeného klíče je dle BIP32 typu *soukromý podřízený klíč ze soukromého rodičovského klíče.*

3.2.1 Model útočníka pro analýzu implementace status-keycard

Pro analýzu uvažujeme tyto standardní předpoklady:

- útočník nemá přímý přístup k hlavnímu kořenovému soukromému klíči (root key) ani k odvozeným podřízeným soukromým klíčům uloženým na kartě,
- útočník nemá přímý přístup k řetězovému kódu (chaincode) a nemůže jej měnit,
- veřejný klíč z hlavního klíčového páru nelze upravovat,
- útočník **může** pro derivaci poskytnout libovolnou derivační cestu.

Cíle útočníka (seřazeny sestupně podle míry závažnosti) jsou následující:

- hodnota hlavního kořenového soukromého klíče (root private key) pro odvození hierarchie,
- hodnota kořenového řetězového kódu (root chaincode),
- jeden z podřízených soukromých klíčů (derived private key) pro odvození částečné hierarchie.

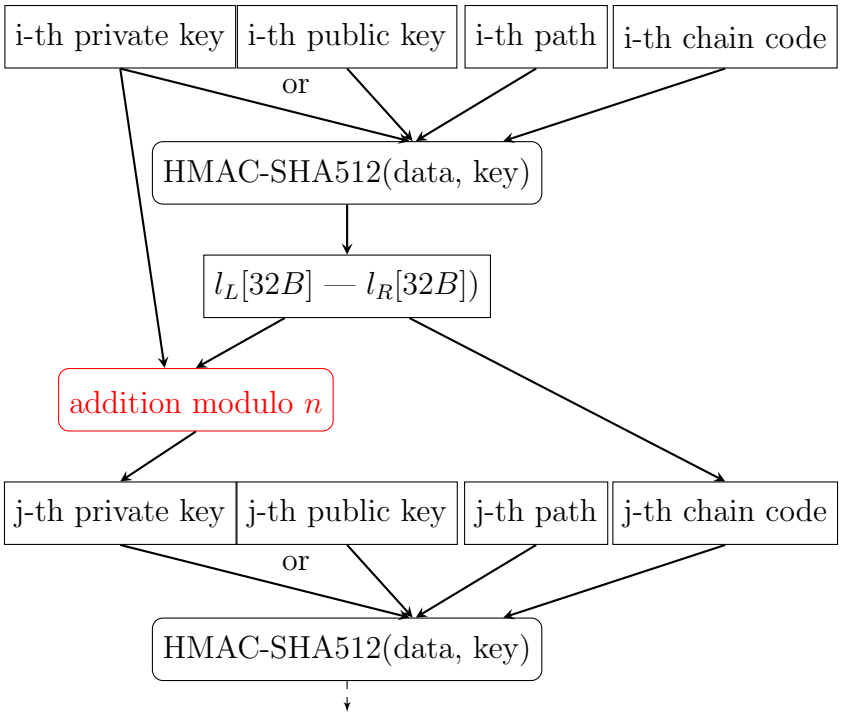
V této sekci zvažujeme pouze izolovanou implementaci pro odvození soukromého podřízeného klíče. Kromě této derivace applet *status-keycard* také obsahuje zabezpečený kanál, správu dat atd.

3.2.2 Proces odvození soukromého klíče

Proces odvození *soukromého podřízeného klíče ze soukromého rodičovského klíče* předpokládá předem nastavený kořenový klíčový pár, který se používá jako kořen pro odvození hierarchie klíčů. Jako doplňující entropie se využívá 32bajtový řetězový kód, který je stejně jako klíčový pár udržován tajný. Odvození konkrétního podřízeného klíče je specifikováno derivační cestou [7]. Proces vypadá následovně (soukromý rodičovský klíč je označen jako k_{par}, veřejný rodičovský klíč jako K_{par}):

1. příprava dat pro HMAC funkci, data obsahují veřejný i soukromý rodičovský klíč,
2. výpočet HMAC-SHA512 nad `data` s řetězovým kódem jako klíčem, výsledkem je `hmaced-data`,
3. rozdělení `hmaced-data` na dvě 32bajtové pole l_L and l_R,
4. podřízený klíč vypočítán jako $(l_L + k_{par}) \mod n$, kde n značí řád bázového bodu křivky,
5. l_R je nový odvozený řetězový kód. [7]

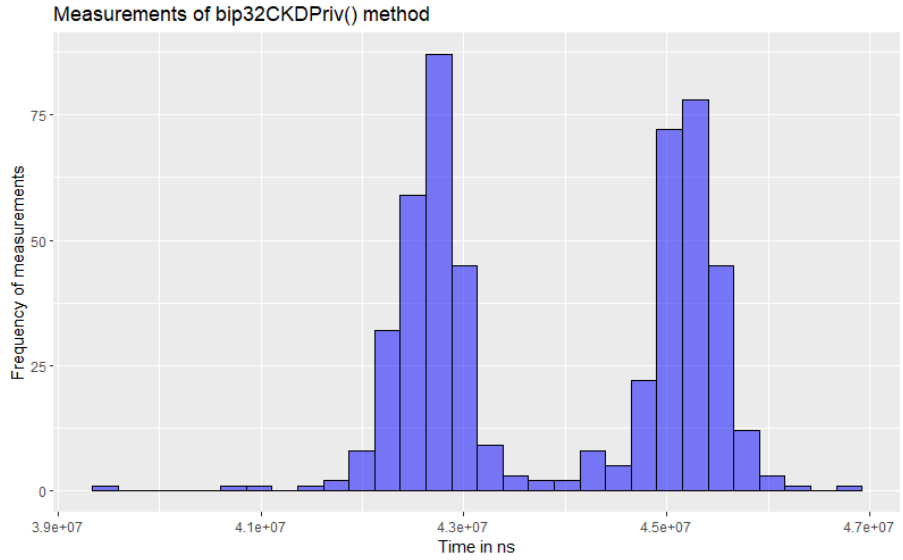
Popsaný proces je ilustrován v následujícím grafu na obrázku 7.



Obrázek 7: Proces derivace soukromého klíče dle BIP32.

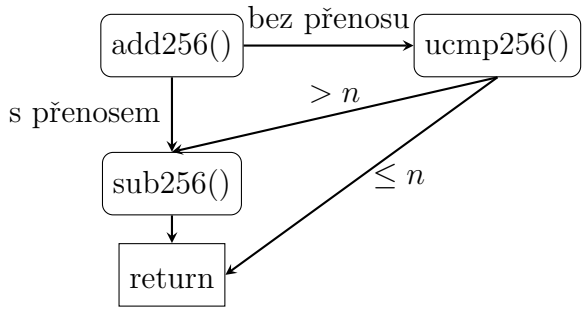
3.2.3 Pozorované časové úniky v implementaci *status-keycard*

S využitím nástroje JCProfilerNext jsme analyzovali metodu apletu *status-keycard* implementující derivaci privátního klíče dle zadané derivační cesty a tajných dat uložených na kartě. Měření času metody `bip32CKDPriv()` s předem nastaveným náhodným kořenovým klíčovým párem a řetězovým kódem vykazuje významné časové rozdíly pro lišící se derivační cesty. Variace času je znázorněna na obrázku 8.



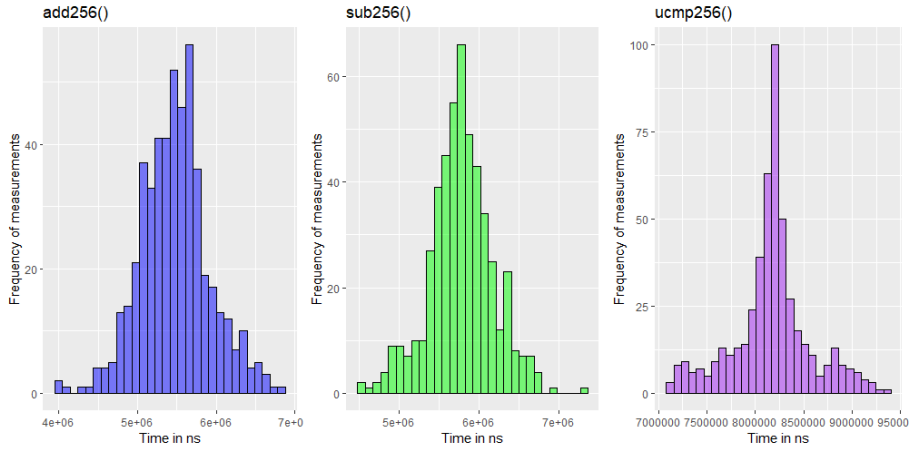
Obrázek 8: Měření metody `bip32CKDPriv()`

Metoda `bip32CKDPriv()` se skládá ze tří hlavních částí – příprava dat pro HMAC, výpočet HMAC a sčítání soukromého klíče s výsledkem operace HMAC modulo řád křivky. Časové měření pro konkrétní řádky kódu v `bip32CKDPriv()` metodě a také ruční posouzení kódu ukazují, že problematickou částí kódu s výraznými rozdíly v čase je implementace modulo sčítání, operace označená červeně na obrázku 7.
Metoda interně používá operace sčítání, porovnání a odčítání. Sekvence provedených operací se liší na základě přenosu na nejvyšším bajtu při sčítání. Průchod metodou je ilustrován na následujícím obrázku 9.



Obrázek 9: Sekvence operací v metodě `addm256()`

Měření ukazuje, že čas zpracování je nezávislý na vstupních datech pro dané operace sčítání, odčítání a porovnání (viz obrázek). Odchylky v čase jsou dány pouze pořadím prováděných operací uvnitř metody `addm256()`.



Obrázek 10: Měření pro dílčí metody zavolané v rámci vykonávání metody `addm256()`.

Podle implementace jednotlivých vnitřních operací lze soudit, že provedení operací sčítání a odčítání zabere méně času než dokončení operace porovnání. Samotné pořadí provedených operací uvnitř `addm256()` metody pak umožňuje rozlišit tři případy:

1. sčítání vede k přenosu na nejvyšším bajtu a provádí se odčítání, tedy $\texttt{hmaced-data} + k_{par} \geq 2^{32}$,
2. sčítání nevede k přenosu na nejvyšším bajtu, při porovnání je součet menší než n nebo rovný n, a odčítání se neprovádí, tedy $\texttt{hmaced-data} + k_{par} \leq n$,
3. sčítání nevede k přenosu na nejvyšším bajtu, při porovnání je součet větší než n, a provádí se odčítání, tedy $n < \texttt{hmaced-data} + k_{par} < 2^{32}$.

Pravděpodobnost případu, kdy se provedou všechny tři operace (sčítání, porovnání i odčítání), je minimální (protože $\texttt{hmaced-data} + k_{par}$ musí splňovat podmínku $n < \texttt{hmaced-data} + k_{par} < 2^{32}$). Můžeme tedy určit přítomnost přetečení (přenosu na nejvyšším bajtu), které se vyskytlo během sčítání výsledku operace HMAC s privátním klíčem, pomocí analýzy doby provádění metody `addm256()`.
Na obrázku 8 značí levá kumulace v histogramu případ, kdy se provede sčítání a odčítání. Naopak kumulace vpravo odpovídá scénáři kombinujícímu operace sčítání a porovnání.

3.2.4 Možnost zneužití časové odchylky při odvození soukromého klíče

Jak bylo diskutováno v předchozí sekci, můžeme rozlišit dvě situace, které nastanou při sčítání se soukromým klíčem použitým k derivaci. Získaná informace z časových odchylek není dostatečná pro praktické provedení útoku na derivaci klíče, neboť únik informace nelze amplifikovat opakováním spouštěním a měřením operace. Použití stejných vstupních dat vede k detekci stejného úniku informace, naopak při použití odlišných vstupních dat je i výsledný (útočníkovi neznámý) privátní klíč odlišný a uniklá informace pro předchozí měření není relevantní.

Příklad 2: Detekce a analýza úniku informací v appletu JCFROST využívajícím knihovnu JCMathLib

JCFROST applet je open-source implementace prahového podpisového schématu FROST v jazyce JavaCard, která se opírá pouze o veřejné rozhraní JavaCard API. Metoda pro podpis v JCFROST appletu se skládá z práce s veřejnými i tajnými daty. Pro výpočty jsou použity metody z JCMathLib, které ale nemusí být časově konstantní a mohou tak do výpočtu zanášet závislost na tajných datech. Výskyt takové časové nekonstantnosti se snažíme 1) detekovat a 2) využít k posouzení míry úniku informace o tajných datech a jejího možného zneužití.
Každý účastník i je na začátku vybaven tajným podílem (secret share) s_i. Tajné podíly byly vygenerovány pomocí Shamirova protokolu pro prahové sdílení tajemství (Shamir Secret Sharing) z tajné hodnoty s. Veřejný klíč odpovídající podílu s_i je označen jako PK_i, skupinový veřejný klíč jako PK.
Jednorázové hodnoty (nonce) jsou čerstvě generovány pro každý podpis. Jsou tajné a nesmí být zveřejněny. Každý účastník generuje dvě různé takové hodnoty – provazující (binding) a maskovací (hiding). K nim odpovídající body na křivce (commitments) jsou veřejné a jsou následně sdíleny se všemi ostatními účastníky.

3.2.5 Model útočníka při analýze implementace JCFROST

Předpokládáme, že nastavení tajného podílu probíhá v bezpečném prostředí skrze zabezpečený kanál, který má zabránit zveřejnění citlivých informací potenciálním útočníkům.
Předpoklady:

- útočník nemá přístup k tajnému klíči s,
- útočník nemá přístup k tajnému podílu jiného účastníka s_i,
- útočník nemůže přímo získat maskovací a provazující jednorázové hodnoty,
- útočník má přístup ke všem veřejným datům,
- útočník může manipulovat se zprávou určenou k podpisu,
- útočník může manipulovat s veřejnými daty ostatních účastníků, které jsou zasílány kartě,
- útočník může měřit dobu trvání operace na straně hostitelského zařízení (např. časovač mezi odesláním a přijetím paketu startujícím operaci), nemůže však měřit dobu trvání dílčích části operace na kartě.

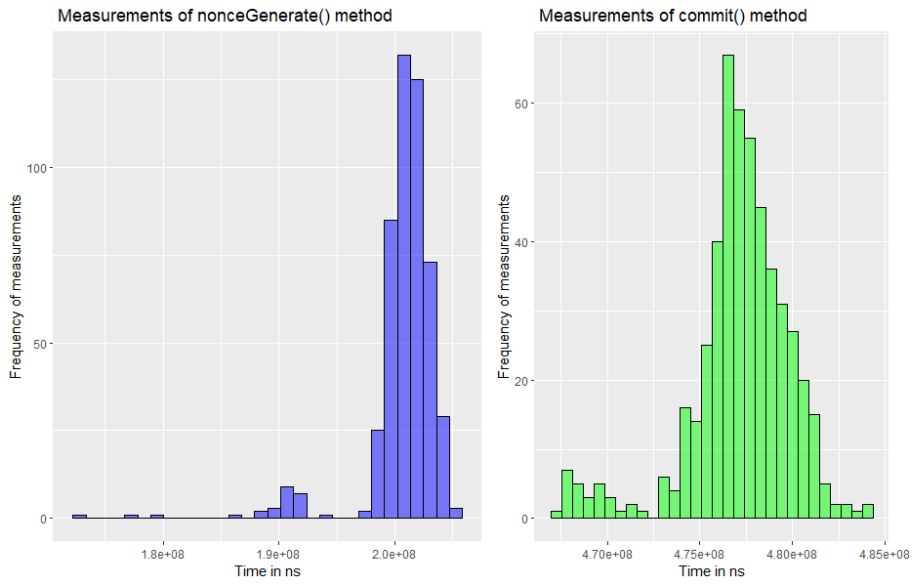
Cílem útočníka je získání tajného podílu účastníka s_i. Získání počtu t z n tajných podílů by mělo za následek schopnost rekonstruovat tajemství s.

3.2.6 Úpravy nástroje JCProfilerNext pro měření appletu JCFROST

Při profilování pomocí nástroje JCProfilerNext je nutné nastavit jednotlivé části datového paketu (APDU), který má být zaslán na kartu, neboť applet JCFROST vyžaduje více po sobě následujících kroků (sekvenci APDU) k vytvoření finálního podpisu. Nástroj JCProfilerNext jsme upravili pro účely testování JCFROST tak, aby byl schopen zajistit všechny potřebné kroky pro profilování jednotlivých operací.
JCFROST definuje několik fází výpočtu – *setup, init, commit, commitment, sign, reset*. Z tohoto důvodu byla metoda `profileImpl()` třídy `TimeProfiler` JCProfilerNext upravena tak, aby podporovala jednotlivé fáze práce s JCFROST a bylo tak možné zvlášť profilovat *commit* a *sign* operace. Pro profilování je možné zafixovat data, se kterými JCFROST pracuje (tajný podíl, nonce, veřejná data).

3.2.7 Operace *commit*

Součástí operace *commit* je vygenerování tajných dat maskovací a provazující jednorázové hodnoty. Odpovídající veřejné části (commitments) jsou sdíleny s ostatními účastníky. V této fázi se již počítá s validně nastaveným tajným podílem na kartě.
Vytvoření maskovací a provazující jednorázové hodnoty zahrnuje generování náhodného bajtového řetězce, který je spolu s tajným podílem od karty dále zahashován (viz [3]). Jako poslední operace se provádí modulo řádu křivky.

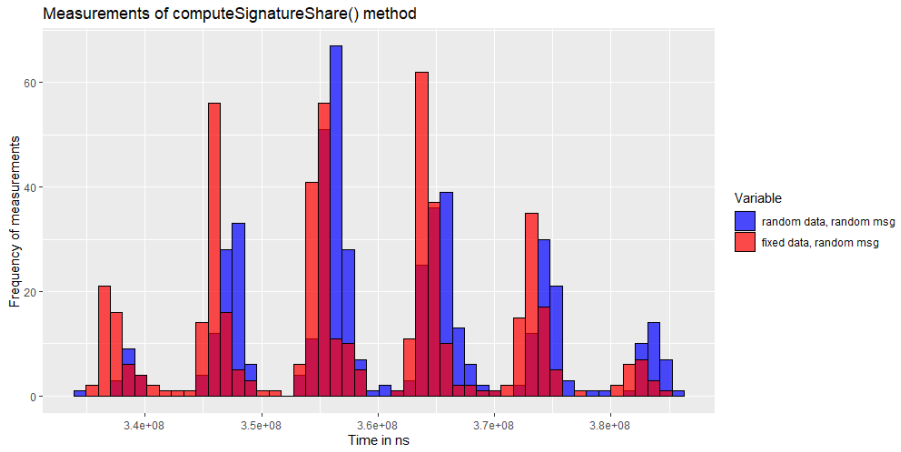


Obrázek 11: Měření metod `nonceGenerate()` a `commit()`

Obrázek 11 obsahuje histogramy časových měření metody `nonceGenerate()` a `commit()` ze třídy `FrostSession`. Měření bylo prováděno v 500 kolech, každé s náhodným tajným podílem, náhodným skupinovým veřejným klíčem a náhodnými jednorázovými hodnotami. Histogram vy-

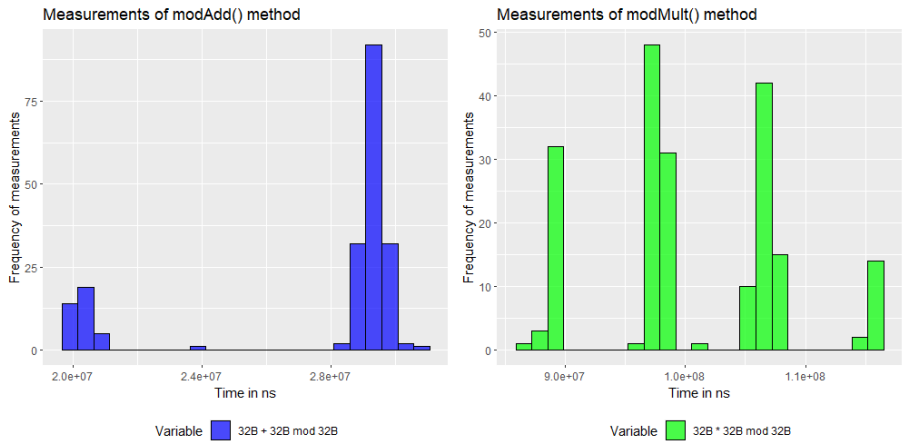
kazuje patrné rozdíly v čase vytváření jednorázové hodnoty. Viditelná odchylka je pravděpodobně způsobena variabilitou v čase provádění operace modulo.

3.2.8 Operace *sign*



Obrázek 12: Měření metody `computeSignatureShare()` s fixními a náhodnými vstupními daty.

Metoda `computeSignatureShare()` přímo vytváří část podpisu zprávy a obsahuje převážně metody z knihovny JCMathLib pro práci s velkými čísly (třída `BigNat`). Graf na obrázku 12 ilustruje viditelné rozdíly pro podpisy náhodných zpráv, používající náhodně generovaná tajná data.

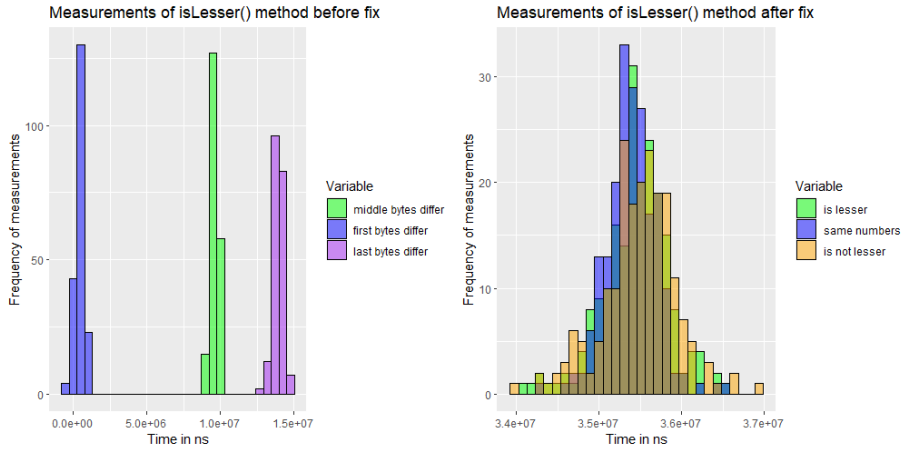


Obrázek 13: Měření metod `modAdd()` a `modMult()`

Časová variabilita při výpočtu části podpisu vzniká použitím metod `modAdd()` a `modMult()` implementujících operace sčítání a násobení. Implementace obou metod obsahují větvení, brzké návraty z funkcí, jsou závislé na délce zpracovávaných čísel, atp. Časové úniky jsou prezentovány na obrázku 13.
Pro jednotlivé časové skupiny viditelné na obrázku 12 následně můžeme zkoumat odpovídající kód a extrahovat další informace o tajných datech na kartě.

3.2.9 Demonstrace proveditelných úprav kódu JCMathLib snižující únik informace postranními kanály

Čas provádění původní implementace (obrázek 14) závisí na velikosti offsetů čísel, které porovnává. Z časového úniku lze také vyčíst, na kterém bajtu se porovnávaná čísla liší. Tato závislost byla odstraněna tím, že byl z funkce odstraněn brzký návrat. Následně bylo rozšířeno porovnávání jednotlivých bajtů v číslech až na velikost alokovaného pole, aby z času nebylo možné poznat offset čísla. Podmíněné větvení bylo v metodě upraveno tak, aby se kód uvnitř spouštěl v každém průchodu. Výsledek úprav je vidět na následujícím obrázku 14. V rámci navazujících prací je plánováno vytváření katalogu možných úprav kódu aplikace tak, aby byl snížen únik informace postranními kanály.



Obrázek 14: Metoda `isLesser()` před úpravou (výrazný časový rozdíl operace v závislosti na vstupních datech) a po opravě (doba trvání operace nezávisí na vstupních datech).

3.2.10 Další informace

Nástroj pro analýzu JCProfilerNext je dostupný v repozitáři `https://github.com/lzaoral/JCProfilerNext`. Analyzované applety status-keycard, JCMathLib a JCFROST jsou dostupné v repozitářích `https://github.com/status-im/status-keycard`, `https://github.com/OpenCryptoProject/JCMathLib` a `https://github.com/crocs-muni/JCFROST`.

4 Závěr

Veškeré plánované činnosti v rámci Etapy 10 byly úspěšně dokončeny a proběhla prezentace výsledků aplikačnímu garantovi v rámci tří osobních setkání a workshopů (květen, září a prosinec 2023). V rámci Etapy 10 byly prováděny práce na třech různých nástrojích a jejich využití pro analýzu certifikačních reportů a vlastností kryptografických implementací útoků postranními kanály.

Nástroj *seccerts* nyní obsahuje možnost automatického párování mezi certifikovanými produkty dle Common Criteria či FIPS140 a databází zranitelností NIST NVD. Párování zranitelností na produkt také umožnilo provést řadu zajímavých analýz vlivu bezpečnostních certifikací na výslednou bezpečnost produktu (hodnocenou metrikou výskytu zranitelností).

Nástroj *pyecsca* poskytuje velmi rozsáhlou databázi možných implementací operací kryptografie eliptických křivek a poskytuje základní možnosti vyhodnocení těchto implementací včetně podpory pro GPU. V navazujících etapách budeme pracovat na rozvoji těchto analýz.

Nástroj *JCProfilerNext* nyní umožňuje statickou i dynamickou analýzu JavaCard appletů a jeho využití bylo demonstrováno na detekci časově nekonstantního chování implementace projektu *status-keycard* a projektu JCFROST využívající knihovny JCMathLib. U knihovny JCMathLib bylo provedena i iniciální úprava zdrojového kódu pro snížení úniku informace časovým kanálem a její praktické ověření. V navazujících etapách budeme pokračovat v těchto úpravách i na základě zpětné vazby od aplikačního garanta.

Reference

[1] Akishita, Toru and Takagi, Tsuyoshi. *Zero-Value Point Attacks on Elliptic Curve Cryptosystem*. In Colin Boyd and Wenbo Mao, editors, Information Security, pp. 218–233. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. ISBN 978-3-540-39981-0.

[2] developers, Keycard. *Keycard applet*. `https://github.com/status-im/status-keycard`.

[3] Komlo, Chelsea and Goldberg, Ian. *FROST: flexible round-optimized Schnorr threshold signatures*. In Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27, pp. 34–65. Springer, 2021.

[4] Renes, Joost, Costello, Craig, and Batina, Lejla. *Complete Addition Formulas for Prime Order Elliptic Curves*. In Proceedings, Part I, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9665, p. 403–428. Berlin, Heidelberg: Springer-Verlag, 2016. ISBN 9783662498897.

[5] Roche, Thomas, Lomné, Victor, Mutschler, Camille, and Imbert, Laurent. *A Side Journey To Titan*. In Michael Bailey and Rachel Greenstadt, editors, 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, pp. 231–248. USENIX Association, 2021.
URL `https://www.usenix.org/conference/usenixsecurity21/presentation/roche`

[6] Sedlacek, Vladimir, Chi-Domínguez, Jesús-Javier, Jancar, Jan, and Brumley, Billy Bob. *A Formula for Disaster: A Unified Approach to Elliptic Curve Special-Point-Based Attacks*. In Mehdi Tibouchi and Huaxiong Wang, editors, Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I, vol. 13090 of *Lecture Notes in Computer Science*, pp. 130–159. Springer, 2021. doi: 10.1007/978-3-030-92062-3_5.
URL `https://doi.org/10.1007/978-3-030-92062-3_5`

[7] Wuille, Pieter. *BIP32: Hierarchical Deterministic Wallets*, 2012. `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`.

[8] Zaoral, Lukas, Dufka, Antonin, and Svenda, Petr. *The adoption rate of JavaCard features by certified products and open-source projects*. In Proceedings of the 22nd Smart Card Research and Advanced Application Conference. Springer, 2023.

Příloha: Analýza rizik

Tabulka 3: Analýza rizik relevantních k Etapě 10.

Riziko	Možný dopad rizika	Skutečný dopad rizika	Datum rizika	Opatření pro minimalizaci/eliminaci
Nemožnost transformace zdrojového kódu pro dosažení úplné časové konstantnosti	Stále zranitelná implementace	Menší doba trvání praktického útoku	-	Studium dalších typů úpravy zdrojových kódů, možnost zapojení randomizačních technik namísto časově konstantních.
Omezení dostupnosti potřebných nízkoúrovňových primitiv na čipových kartách potřebných pro další rozvoj knihovny JCMathLib	Omezení nebo nemožnost implementovat kryptoprimitivum vyšší úrovně	Snížený výkon implementace kryptoprimitiva	-	Průběžné testování na kartách různého typu, zavedení profilů karet umožňující využívat alternativní (pomalejší) implementace, návrh nových možností implementace
Nedostatečné rozlišení časomíry na straně PC pro nástroj JCProfiler-Next	Nemožnost přesného měření doby trvání operací	Snížená přesnost měření	-	Vývoj podpory přesného měření času s využitím externího osciloskopu