Úvod

Tato práce se zabývá tzv. TEE[1] v AMD, specifičtěji *Secure Memory Encryption* (SME) a *Secure Encrypted Virtualization* (SEV). Dále krátce popisuje TEE u konkurenčních výrobců Intel a ARM. Všechny zmíněné implementace TEE nakonec práce porovnává a analyzuje z hlediska bezpečnosti.

Trusted execution environment

Při studiu jsem vycházel z [9, 11, 24]. TEE představuje prostředí ve kterém TA[2] a TOA[3] provádějící kód mohou mít vysokou úroveň důvěry ve správnost, izolaci a řízní přístupu při správě aktiv okolního chráněného prostředí, protože mohou ignorovat hrozby „neznámého“ zbytku zařízení (tj. REE[4]). To je zajištěno pomocí:

- **Bezpečného startu** – všechen kód vykonaný v rámci TEE musí být vhodně autorizován, kde daná autorizace je ověřená jiným autorizačním kódem počínaje bootováním z ROM (autorizováno svou přítomností). Taková autorizace je vykonávána pomocí certifikátů, neměnností, či držením v izolaci.

- **Izolace běhu** – Pro zajištění izolace obsahu TEE je tato část fyzicky oddělena od zbytku zařízení a v rámci TEE je izolace zajištěna tak, že TA může přímo přistupovat pouze k vlastním aktivům (např. data), nelze přistupovat k aktivům jiných komponent běžících v TEE. TA tedy důvěřuje pouze TOA a nikomu jinému.

Na určitých typech zařízení je třeba zajistit bezpečnou instalaci nového kódu (TA), což lze zajistit pomocí důvěryhodné vzdálené správy, kde komunikace s ní je třeba zabezpečit kryptografickými klíči. Vzdálená správa TEE je navržena tak, že vzdálení manažeři mohou spravovat pouze jejich sadu TA, a nelze nijak kontrolovat ostatní TA v TEE, které nevlastní.

TEE tedy izoluje TA od možného malware v REE a ostatních aplikací v TEE, a to bez významných omezení funkcionality či výkonu. Stává se proto důležitou součástí všech zařízení s rostoucími požadavky na bezpečnost. Nevýhodou je nedostatek standardizace napříč výrobci TEE (AMD, Intel, ARM, ...), a proto vývojáři TA musí často vyvíjet několik verzí TA pro různé TEE, aby vyhovovala požadavkům dané TEE.

Implementace TEE

Zajistit chráněné prostředí pro provádění lze různými způsoby. V této kapitole jsou popsány techniky největších výrobců čipů na světě, používaných od nízko-výkonových vestavěných zařízeních až po vysoko-zátěžové virtualizační servery.

AMD SME a SEV

Tato sekce popisuje implementaci TEE společností AMD založenou na šifrování hlavní paměti (SME). Dále popisuje zabezpečenou virtualizaci pomocí kombinace šifrování a architektury AMD-V (SEV). Při studiu jsem vycházel z [1, 3, 8, 25].

AMD SME

SME je jednoduchý a efektivní mechanismus pro šifrování hlavní paměti. Oproti standardním technikám šifrování je SME přímo integrováno v CPU architektuře, flexibilní a škálovatelné od vestavěných zařízení až po komplexní servery. To vše bez nutnosti jakýchkoli úprav aplikačního SW.

Data jsou už dnes běžně při ukládání na disk šifrována, avšak při načtení do DRAM jsou data uložena v otevřené podobě a náchylná ke kompromitaci. Šifrováním hlavní paměti lze předcházet různým druhům útoků.

Princip Šifrování je prováděno pomocí dedikovaného HW, a sice kontrolérů v tzv. *on-die* paměti. Každý takový kontrolér obsahuje vysokovýkonnostní modul AES (*Advanced encryption standard*), který šifruje data při zápisu do DRAM a dešifruje při čtení z DRAM, jak lze vidět na snímku 1. Šifrování je prováděno pomocí klíče délky 128 bitů v režimu chránícím před útoky typu posunu bloku šifrovaného textu.

[1] *Trusted execution environment* – v překladu chráněným prostředím pro spouštění kódu.
[2] *Trusted application* – v překladu důvěryhodná aplikace.
[3] *Trusted operating system* – v překladu důvěryhodný operační systém.
[4] *Rich operating system execution environment* – v překladu operační systém zbytku zřízení, např. Android™.

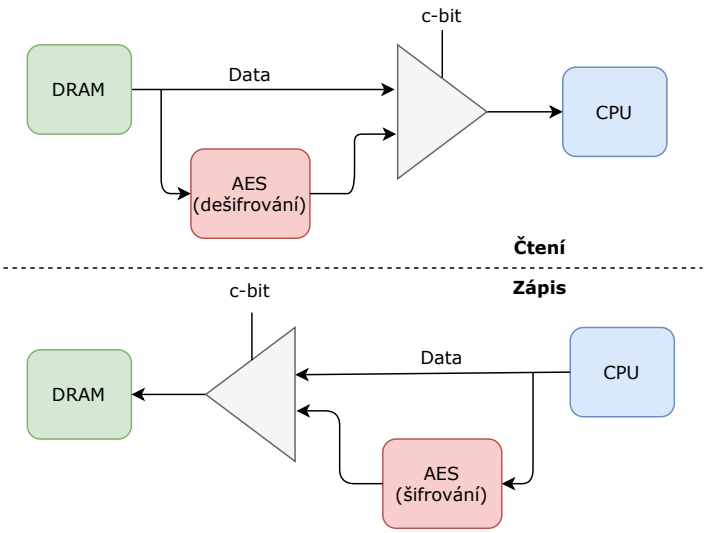
Šifrování a klíče Šifrovací klíč pro modul AES je **náhodně** vygenerován při každém restartu systému. Tento klíč není viditelný pro **žádný** SW běžící na CPU. Klíč je plně ve správě AMD bezpečnostního procesoru, což je 32-bitový mikrořadič (ARM Cortex-A5), který funguje jako dedikovaný bezpečnostní subsystém v rámci AMD SoC (*System on Chip*). Náhodné generování klíče funguje v souladu se standardizací NIST SP 800-90 a klíč je následně uložen v dedikovaných HW registrech a není v otevřené podobě použit mimo SoC. SME nevyžaduje SW běžící na CPU aby se podílel na správě klíče, narozdíl od SEV.

Řízení šifrování Určení, které části paměti budou šifrované, je prováděno OS nebo hypervizorem v rámci SW tabulek stránek. Pro indikaci které stránky mají být šifrovány a které ne je využit fyzický bit č. 47, zvaný jako **c-bit**. Nastavením tohoto bitu na logickou hodnotu 1 dané stránce hypervizorem či OS způsobí, že přístup k této paměti bude (de-)šifrován pomocí AES modulů popsaných výše.

Výkonnost Z pohledu výkonnosti (de-)šifrování paměti pomocí AES modulů způsobuje při přístupu k DRAM malé výkyvy latence a vliv této latence na SW běžící na CPU značně závisí na aktuálním zatížení. Odhaduje se však, že vliv na celkovou výkonnost systému je velmi nízký.

Modely pro použití SME poskytuje dva způsoby, jakým jej lze využít, a sice částečné a úplné šifrování paměti. Částečné šifrování paměti je založeno na výše zmíněném c-bitu a poskytuje OS či hypervizoru flexibilitu v možnosti výběru, které stránky obsahují citlivá data a budou šifrovány a které nikoli. V případě úplného šifrování paměti je c-bit obsolentní (resp. je nastaven vždy na 1) a využívá se zejména v případě vyšší pravděpodobnosti fyzického útoku a tím že je šifrován celý obsah DRAM, tak je zamezeno i útokům typu *cold boot* či DRAM *interface snooping*, atd.

Transparentní SME Aby SME mohlo fungovat, vyžaduje podporu v OS/hypervizoru a sice nastavování c-bitu. Pro systémy, které používají OS/hypervizor bez zmíněné podpory, avšak vyžadující šifrování DRAM, je možné využít TSME. Při použití TSME je veškerá paměť šifrovaná i beze změn v SW.



Obrázek 1: Princip (de-)šifrování při použití SME. Vytvořeno na základě [8].

AMD SEV

SEV kombinuje možnosti šifrování hlavní paměti s existující virtualizační architekturou AMD-V za účelem podpory šifrovaných virtuálních strojů. To chrání VM (*Virtual machine* – v překladu virtuální stroj) nejen proti fyzickým útokům, ale také vůči ostatním VM a hypervizorem samotným. To umožňuje použití v cloudovém prostředí, kde není vyžadován plně důvěryhodný hypervizor a administrátor systému. Stejně jako u SME, není vyžadována žádná změna aplikačního SW.

Princip SEV je rozšíření AMD-V architektury která umožňuje běh několika VM pod jedním hypervizorem. Při použití SEV jsou pomocí HW označena všechna data či kód a sice s VM ACID (štítek) indikující, z které VM data pochází nebo pro kterou VM jsou data určena. Tento štítek je u dat udržován po celou dobu a zaručuje, že přístup k datům či jejich modifikaci může provádět pouze jejich vlastník. Štítek chrání data jen v rámci SoC, zatímco při opuštění SoC jsou data chráněna pomocí šifrování SME (AES 128 bit). Tzn. že při opuštění či vstupu dat z/do SoC, data jsou (de-)šifrována pomocí klíče spojeného s daným štítkem (vlastníkem). Ke každé VM a hypervizoru se tedy váže štítek, kterým se označují data daného vlastníka, a ke štítku se zase váže AES klíč, kterým jsou daná data šifrována mimo SoC. Při přístupu k datům jinou VM/hypervizorem, jiným než je vlastník, uvidí data pouze v šifrované podobě, což poskytuje silnou kryptografickou izolaci mezi VM a hypervizorem a VM

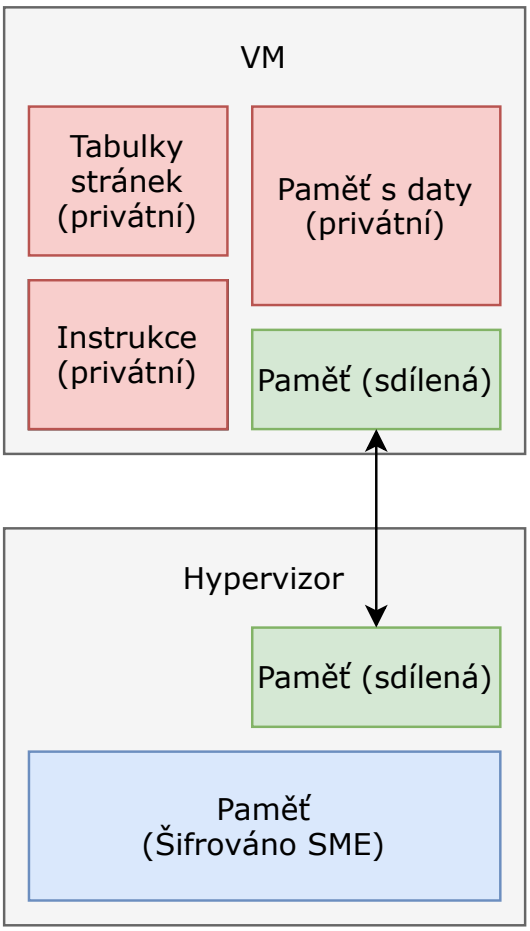
navzájem.

Šifrování a klíče SEV využívá stejné vysokovýkonnostní AES moduly pro (de-)šifrování paměti jako SME popsané v sekci . Klíčovou vlastností SEV je, že VM si mohou vybrat, jaká data budou důvěrná (opět označeno pomocí c-bit na standardních CPU tabulkách stránek) tzn. rozhodnutí, která privátní data dané VM budou šifrované je plně v režii dané VM (vlastníka dat) a šifrování se provádí pomocí klíče pevně svázaného s danou VM a štítkem. Sdílená paměť může být naopak šifrována pomocí klíče hypervizoru. Standardně VM označuje veškeré s kódem a daty jako privátní (k zašifrování) kromě specifických stránek, které slouží pro komunikaci s ostatními VM a hypervizorem, jak lze vidět na snímku 2. SEV HW vyžaduje aby určitá data (tabulky stránek a stránky instrukcí) byly vždy privátní, aby chránil VM.

Bezpečnost Bezpečnost SEV závisí zejména na schopnosti zajistit bezpečnost kryptografických klíčů jednotlivých VM. Proto hypervizor musí spravovat jednotlivé VM a jejich zdroje, avšak nesmí znát jejich klíče. To je zajištěno pomocí firmwaru SEV běžícího v rámci AMD-SP. Daný firmware zajišťuje bezpečné rozhraní pro bezpečnou správu klíčů – to je zajištěno pomocí:

- **Autentizace platformy** – Autentizace platformy předchází nebezpečnému SW vydávat se za legitimní platformu. Autentizace je prováděna pomocí klíče „identity“. Tento klíč je podepsán od AMD a vlastníka platformy, což zajišťuje, že je platforma věrohodná a poskytuje SEV.
- **Atestace zapnuté VM** – Atestace vlastníkům VM zaručuje, že VM se úspěšne zapnula s povoleným SEV režimem. Firmware vlastníkovi VM poskytne podepsané určité komponenty spojené se SEV z dané VM (jako např. počáteční stav paměti), čímž proběhne ověření správného a bezpečného stavu VM předtím, než jsou VM předána nějaká důvěrná data vlastníka.
- **Důvěrnosti dat VM** – Důvěrnost dat VM je zaručena pomocí šifrování paměti klíčem dané VM, který zná pouze zabezpečený firmware SEV. Tento firmware neposkytne žádná důvěrná data (jako klíče) bez správné autentizace, což chrání data VM před hypervizorem, který může být potenciálně nebezpečný či chybný.

SEV také poskytuje možnost migrace SEV chráněných VM, a sice tak, že důvěrná data jsou přenášena v šifrované podobě, jak jsou uložena v DRAM. Poté, co je vzdálená platforma autentizována je pomocí zabezpečeného kanálu odeslán potřebný klíč dané VM.



Obrázek 2: Standardní schéma VM a hypervizoru v SEV, komunikující pomocí sdílené paměti. Vytvořeno na základě [8].

AMD SEV-ES ES[5] je označení další generace SEV, která šifruje obsah všech registrů procesoru po vypnutí VM. Toto brání úniku informací z registrů procesoru komponentám jako je hypervizor a dokáže dokonce detekovat nebezpečné modifikace stavu registrů procesoru.

[5] *Encrypted state* – v překladu šifrovaný stav.

AMD SEV-SNP SNP[6] je označení nové generace SEV. SNP přichází se silnou ochranou integrity paměti založenou na HW, jejímž cílem je předejít nebezpečným útokům ze strany hypervizoru, jako je útok opakováním dat (*replay attack*), re-mapování paměti, atd.

Intel SGX

Při studiu jsem vycházel z [7, 22]. Intel SGX (*Software Guard Extensions*) je sada rozšíření architektury Intel jejímž cílem je poskytnout integritu a důvěrnost.

Princip V SGX se používá tzv. „enkláva“ (zabezpečený kontejner na vzdáleném počítači), který obsahuje pouze důvěrná data pro výpočet a kód provádějící výpočet. Enkláva tedy izoluje data a kód od nedůvěryhodného okolního prostředí (OS, hypervizor, ...), ale zároveň zanechává tradiční vrstvení SW v Intel architektuře, kde OS a hypervizor využívá zdroje počítače. SGX vyčleňuje část paměti zvanou PRM (v překladu paměť rezervovaná procesorem). Procesor chrání PRM před všemi přístupy zvenčí enklávy (včetně OS a hypervizoru). PRM si udržuje tzv. cache stránek enkláv (EPC), kde každá stránka má velikost 4KB a obsahuje kód a data enklávy, která ji vlastní. Přiřazování stránek enklávám zajišťuje nedůvěryhodný systémový SW, který udržuje metadata o každé stránce v EPCM, aby zajistil, že každá stránka je přiřazena maximálně jedné enklávě. Počáteční kód a data enklávy jsou načtena nedůvěryhodným SW, z nechráněné paměti do EPC a SW také přiřazuje danou stránku enklávě. Po načtení všech dat a kódu je enkláva považována za inicializovanou a je v CPU spočten kontrolní součet obsahu a může být provedena atestace zajišťující, že uživatel komunikuje se SW v bezpečném prostředí.

Atestace SGX je stejně jako předchůdci založen zejména na atestaci SW. Atestace poskytuje uživateli jistotu, že komunikuje s určitým SW zabezpečeným uvnitř enklávy na důvěryhodném HW, avšak v potencionálně nebezpečném prostředí (hypervizor na vzdáleném počítači, ...). Atestace je založena na kryptografických podpisech, které certifikují kontrolní součet obsahu/stavu vzdáleného kontejneru. Uživatel vzdáleného výpočetního prostředku nenahraje data na vzdálený kontejner, jehož kontrolní součet nesedí s očekávaným součtem. Vzdálený počítač ověřuje tzv. atestovací klíč (vytvářející podpis) proti schvalovacímu certifikátu (vytvořeného výrobcem onoho důvěryhodného HW). Zmíněný klíč je známý pouze důvěryhodnému HW a je použit pouze pro atestaci.

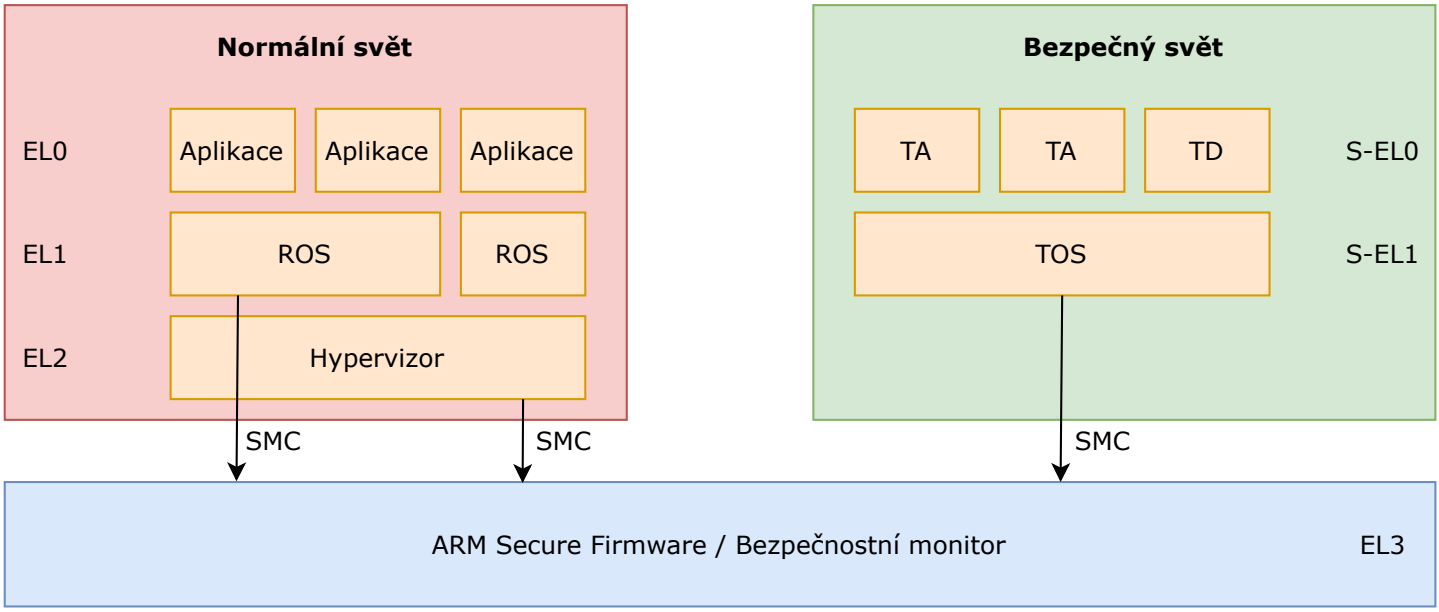
Bezpečnost Řízení programu může do enklávy přistoupit pouze pomocí **speciálních instrukcí**, podobných přepínaní z uživatelského do kernel módu. Spuštění kódu enklávy vždy probíhá v chráněném prostředí a používá překlad adres OS/hypervizoru. Aby neunikala důvěrná data, CPU, který provádí kód enklávy, neobsluhuje přímo přerušení ani výpadky stránek. První provede tzv. asynchronní opuštění enklávy do uživatelského módu (vrstva 3 v architektuře Intel) a až poté obslouží ono přerušení či výpadek. Alokace stránek EPC enklávám je delegována OS/hypervizoru. OS tuto informaci deleguje SGX implementaci pomocí instrukcí z vrstvy 0 (kernel). OS může také přemístit EPC stránky do nedůvěryhodné DRAM a načíst zpět pomocí speciálních instrukcí. SGX používá kryptografické klíče pro zajištění integrity a důvěrnosti stránek zatímco jsou uloženy v DRAM.

Výkonnost Z pohledu výkonnosti SGX používá velké množství informací roztroušené po mnoha zdrojích. Výhodou je, že klíčová část – atestace – se provádí pouze na začátku, avšak při každém přerušení apod. je nutné přepínat režim procesoru. Dalším problémem může být velké množství enkláv nebo velké množství dat zpracovávaných v nich, kdy by bylo potřeba dočasně ukládat stránky do DRAM, což vyžaduje poměrně náročné kryptografické zajištění bezpečnostních cílů a tedy výkonnostní *overhead*.

ARM Trustzone

Při studiu jsem vycházel z [10, 13, 14, 15]. ARM procesory s podporou TrustZone (v překladu zóna důvěry) obsahují architektonické bezpečnostní rozšíření, ve kterém každý fyzický procesor poskytuje dvě virtuální jádra, jedno bezpečné (zvané bezpečný svět) a druhé nezabezpečené (zvané normální svět) a mechanismus pro přepínání kontextu mezi nimi, zvaný mód monitoru.

[6] *Secure nested paging* – v překladu zabezpečené vnořené strankování.



Obrázek 3: Model úrovní privilegií v ARM TrustZone. Vytvořeno na základě [14, 15].

Princip ARM TrustZone přináší změnu v tom jak CPU může pracovat. Když pracuje v „bezpečném“ režimu, může pracovat pouze s podmnožinou periferních zařízení a přistupovat pouze k části HW a paměti zařízení. TA a TD[7] dokáží tuto vlastnost využít a umístit kód a data do zabezpečené části, kde izolace zajišťuje, že k těmto datům nelze přistoupit z normálního světa. Ačkoli nic z normálního světa (ani OS/hypervizor) nemůže přistupovat k zabezpečené paměti, kód běžící v bezpečném světě může volně přistupovat k nezabezpečené paměti.

Privilegia TrustZone také mění standardní model privilegií CPU (zvané EL – *exception levels*), jak lze vidět na snímku 3. Dříve (a v normálním světě) jsou tři úrovně privilegií, a sice EL0 (uživatelský mód), EL1 (mód kernelu) a EL2 (mód hypervizoru). Bezpečný svět zavádí S-EL3 (mód bezpečného monitoru), který má nejvyšší privilegia a řídí celý systém a také mění předchozí módy, a sice následovně:

- S-EL0 – Zde běží neprivilegované TA a TD. Výchozí nastavení je, že TA nemohou komunikovat s ostatními TA, periferiemi, nebo ostatními procesy v bezpečném ani nebezpečném světě. To mlže jedině explicitně povolit TOS (důvěryhodný OS).
- S-EL1 – Zde běží TOS, a u některých implementací zde můžou běžet některé TD.
- S-EL2 – *V bezpečném světě není hypervizor.*
- S-EL3 – Zde běží kód bezpečnostního monitoru – ARM Trusted Firmware (v překladu důvěryhodný firmware ARM) – poskytnutého výrobcem. Provádí změnu kontextu mezi ROS a TOS a poskytuje SMC[8], který lze využít oběma ROS a TOS pomocí speciální instrukce.

Komunikace Komunikaci mezi bezpečným a normálním světem lze zajistit dvěma různými způsoby. První je založen na výše zmíněné SMC. Dále je možné využít sdílené paměti, kdy si obyčejná i zabezpečená aplikaci namapují stejnou fyzickou paměť do svého adresového prostoru. Je potřeba, aby TA označila paměť jako „nezabezpečenou“. Data zapsané do této části poté vidí oba procesy a lze tak efektivně přenést velké množství dat bez nutnosti přepínání kontextu.

Analýza implementací TEE

V této kapitole jsou slovně analyzovány jednotlivé implementace z pohledu bezpečnosti a výkonnosti. Analýza je založena na všech pracích, ze kterých jsem při studiu problematiky vycházel, dále pak na pracích [21, 17, 19, 23].

[7] *Trusted driver* – v překladu důvěryhodný ovladač.
[8] *Secure monitor call handler* – Bezpečné volání monitoru.

Analýza SW implementací První pohled, ze kterého je vhodné analyzovat implementace TEE je, zda jsou implementované v SW či HW. Implementace TEE založené na SW (např. SofTEE, Virtual Ghost a SKEE) mají několik výhod oproti implementacím založených na HW. První z nich je, že jejich využití je vhodnější na použití v různorodých HW prostředích, protože nejsou závislé na speciálním HW. Za druhé, v SW implementacích je značně jednoduší a levnější vytvořit novou verzi – rychlá reakce na zranitelnosti v bezpečnosti je klíčová. A poslední, nerozšiřuje HW TEE o další možné zranitelné části, tzn. že SW a HW implementace TEE mohou být použity dohromady. Nevýhodou SW řešení je zejména tzv. *overhead* způsobený delegováním privilegovaných operací bezpečnostnímu monitoru (kvůli implementaci kernel de-privilegování). Dále pak větší tzv. „plocha útoku“ SW implementace (tzn. více částí, které lze napadnout a využít pro útok) – např. fyzické útoky jako monitorování sběrnice pro odchycení kryptografických klíčů či útoky na DMA, které mohou obejít kernel de-privilegování.

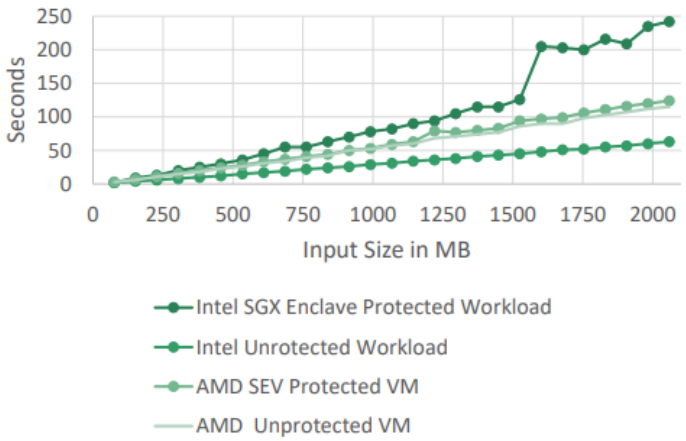
Analýza AMD SME a SEV AMD používá pro šifrování rychlý způsob AES 128 bit v režimu ECB – *electronic codebook*. Režim ECB má známou bezpečnostní díru a sice únik informace ze zašifrovaného textu (stejný vstupní blok otevřeného textu produkuje stejný blok šifrovaného textu). AMD pro tyto účely používá zmíněné kombinování adresy a otevřeného textu (což zaručí, že stejný otevřený text na různých místech paměti bude produkovat i jinou šifru). Na druhou stanu, AMD SME/SEV nepodporuje ochranu integrity šifrované paměti, což snižuje bezpečnost. AMD SEV je navrženo pro použití na veřejných cloudech, kde jsou největším bezpečnostním rizikem útoky napříč VM nebo z hypervizoru. Výhodou je zde to, že ochrana je z pohledu uživatelských aplikací transparentní, a tedy není třeba provádět dodatečné změny aplikací, protože ochrana proti hypervizoru je založena na šifrování obsahu paměti VM se SEV. To je velkou výhodou pro koncové uživatele, protože SEV ochrana v podstatě nevyžaduje žádné úsilí. SEV také dokáže oproti SGX chránit velké kusy paměti s udržením relativně nízkého výkonnostního *overhead*-u. Nevýhodou zde však je, že OS a hypervizor jsou umístěny do důvěryhodné výpočetní báze, což otevírá možnosti pro větší množství útoků a tedy poskytuje nižší úroveň zabezpečení, než SGX/ARM.

Analýza Intel SGX Intel SGX oproti AMD opatrně odděluje důvěryhodné a ne-důvěryhodné prostředí, poskytuje chráněné vstoupení/opuštění enklávy (důvěryhodného prostředí), vynucuje kontrolu přístupu do paměti a poskytuje ochranu integrity paměti. Na druhou stranu ve výkonnostním srovnání – nadbytek práce, který zajištění bezpečnosti přináší oproti nezabezpečenému použití – je značně horší než například AMD. To je způsobeno zejména jiným záměrem použití. Intel SGX je navržen pro malé aplikace závislé na vysoké úrovni bezpečnosti a privátnosti, jako například zabezpečování přihlašování do internetového bankovnictví či zabezpečení manažera hesel v PC/mobilu, které interagují s relativně malým množstvím citlivých dat. Avšak existují i práce, které se snaží aplikovat Intel SGX v aplikacích, které pracují s větším množstvím dat až dokonce do veřejných cloudových řešení. Dalším velmi důležitým faktorem při výběru implementace TEE je, že aplikace, které mají začít využívat Intel SGX potřebují značnou refaktorizaci kódu. Také omezená velikost EPC paměti kde jsou uloženy stránky s citlivými daty způsobuje značné snížení výkonnosti při kopírování stránek do DRAM, které musí být kryptograficky zabezpečeny pro zajištění důvěrnosti a integrity.

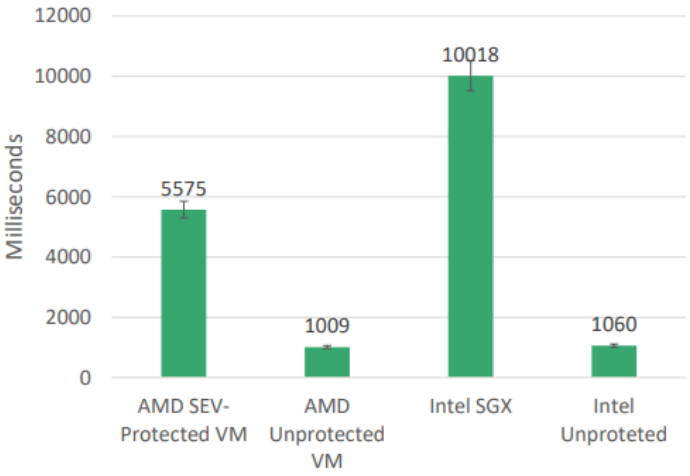
Analýza ARM TrustZone Stejně jako Intel SGX, i ARM TrustZone odděluje důvěryhodné a ne-důvěryhodné prostředí, zde známé jako normální a bezpečný svět. Procesory typu Cortex-A obsahují NS – *non-secure* bit, který udává, v jakém světě zrovna procesor pracuje a poskytuje také bezpečnostní monitor pro komunikaci mezi světy. Aplikace běžící v normálním světě nejsou schopny přistoupit k ničemu v bezpečném světě. V procesorech typu Cortex-M není zabudovaný bezpečnostní monitor pro ušetření času a spotřeby energie a obsahuje pouze jednu TrustZone – tyto procesory jsou cíleny na použití s jedním účelem v IoT. ARM TrustZone poskytuje vysokou míru ochrany, v některých typech útoků dokonce vyšší než Intel SGX, například při re-mappování paměti je ARM zabezpečen, protože tabulky stránek a TLB jsou rozděleny na dvě části v TrustZone a jsou přístupné pouze z bezpečného světa, zatímco u SGX jsou přístupné i z nedůvěryhodného SW. Nevýhodou je, že je potřeba re-faktorizace kódu, a sice která část aplikace poběží v bezpečném a která v normálním světě. Další výhodou je poměrně nízký *overhead* při použití TrustZone a specializace na použití v IoT.

Tabulka 1: Porovnání implementací TEE z pohledu odolnosti proti zranitelnostem. Vytvořeno na základě [1, 7, 19, 23, 17]. Legenda: „?“ značí, že informaci se nepodařilo dohledat. Podrovnější informace o AMD SEV lze dohledat v tabulce 2.

Zranitelnost	**AMD SEV**	**Intel SGX**	**ARM TrustZone**
Mapování paměti (pasivní)	✓	×	✓
Mapování paměti (aktivní)	✓	✓	✓
Útok na DMA	✓	✓	✓
Útok na firmware	✓	✓	✓
Odepření služby	✓	×	✓(slabě)
Vedlejší paměťové kanály	×	×	×
Útok na porty	?	✓	✓
Odposlech sběrnice	?	✓	✓
Útok na čip	×	×	×
Útok změnou napětí	×	×	×



Obrázek 4: Porovnání výkonnosti při větší zátěži na algoritmu *quicksort* s různou velikostí vstupu. Převzato z prezentace [18].



Obrázek 5: Porovnání výkonnosti šifrovacích modulů při zpracování 2.8GB. Převzato z prezentace [18].

Srovnání a útoky na implementace TEE

Tabulka 1 shrnuje několik základních tříd útoků na TEE a podává informaci o tom, které z implementací jsou vůči danému útoku odolné. Na snímcích 4 a 5 lze vidět porovnání výkonnosti Intel SGX (Core i7-6700) a AMD SEV (EPYC-7251).

Útoky na AMD SEV

V tabulce 2 lze dohledat, vůči jakým typům útoků mohou být jednotlivé verze SEV odolné. Na stránce [2] lze nalézt různé zranitelnosti SEV nahlášené zejména bezpečnostními výzkumníky, podrobnosti o nich a poskytnuté záplaty v reakci na ně.

Výzkumníci z univerzity v Grazu prezentovali nástroj pro diferenční analýzu napájení využívající Linuxové rozhraní Running Average Power Limit (RAPL) pro různé útoky založené na postranních kanálech. Rozhraní AMD RAPL umožňuje rozlišovat různé instrukce prováděné na procesoru AMD Ryzen. To by mohlo umožnit podobné útoky na procesory AMD, např. proti AMD SEV-SNP, kde není vyloučena možnost útočníka se nacházet v privilegovaném režimu v kernel prostoru, odkud je možné tento útok provést. V reakci na to, AMD vynucuje privilegovaný přístup k RAPL [2, 16].

Bylo zjištěno, že implementace eliptických křivek v SEV je zranitelná vůči útoku neplatnou křivkou – při startu VM může útočník poslat body křivek malého řádu, které nepatří mezi oficiální NIST křivky a přinutit tím firmware SEV, aby vynásobil

Tabulka 2: Tabulka udávající typy zranitelností a možnost ochrany proti nim u jednotlivých verzí SEV. Převzato z [1].

Zranitelnost	**SEV**	**SEV-ES**	**SEV-SNP**
Důvěrnost			
Paměť VM	✓	✓	✓
Stav registrů VM po ukončení	×	✓	✓
Ochrana DMA	✓	✓	✓
Integrita			
Ochrana proti útoku opakováním (*replay*)	×	×	✓
Ochrana proti korupci dat	×	×	✓
Aliasing paměti	×	×	✓
Re-mapování paměti	×	×	✓
Dostupnost			
Odepření služby VM	×	×	×
Odepření služby hypervizoru	✓	✓	✓
Smíšené			
Rollback (navrácení) TCB (*trusted computing base*)	×	×	✓
Útok pomocí přerušení nebo vyjímky	×	×	✓(volitelné)
Nepřímé otrávení prediktoru	×	×	✓(volitelné)
Zneužití hardware ladících registrů	×	×	✓(volitelné)
Podvržení CPUID informací	×	×	✓(volitelné)
Postranní kanály architektury	×	×	×
Postranní kanály na úrovni tabulek stránek	×	×	×
Sledování výkonnostního čítače	×	×	×

malý bod soukromým DH skalárem firmwaru. Shromážděním dostatečného množství modulárních zbytků může útočník získat kompletní soukromý klíč PDH. S PDH může útočník obnovit klíč relace a získat tajemství virtuálního počítače. AMD vyřešilo tento problém tak, že PSP od určité verze odmítá body, které nejsou na křivkách NIST a vrací chybu `INVALID_CERT` [2, 12, 6].

Při útoku na SEV lze použít např. Fake SEV, který předstírá přítomnost SEV a lze využít poskytovatelem cloudu ke zvýšení počtu běžících VM nebo k přístupu k jakýmkoli datům uvnitř uživatelské VM. Aby se útočník mohl vydávat za certifikovanou AMD SEV platformu, potřebuje jednorázový přístup k systému využívající SEV, a sice za účelem extrakce privátního klíče CEK (Chip Endorsement Key) a odpovídajícího ID platformy. Není nutné aby tento specifický systém (ze kterého byly CEK a ID získány) později hostoval VM, na kterou se bude v rámce FakeSEV útočit. CEK je uložen v PSP (Platform Security Processor, tedy důvěryhodném firmware platformy AMD). Na PSP je však možné zaútočit a získat uložené tajemství, jako je například zmíněný CEK. PSP je uložen v nezdokumentované oblasti UEFI firmware, kde byly odhaleny chyby v mechanismu ověřování certifikátů od PSP OS, které umožňují načítat vlastní komponenty prostřednictvím upravených obrazů UEFI. A to buď fyzicky pomocí SPI programátoru nebo pomocí aktualizačního mechanismu firmware vytvořeného výrobcem čipu. Při aktualizaci obrazu nebyla vyžadována kontrola certifikátu, nebo ji bylo možné vypnout. Tento problém PSP OS byl výrobcem opraven, avšak platforma neposkytuje žádnou ochranu proti použití starších verzí PSP OS, tudíž lze chybu nadále využít. S dostupným CEK a ID platformy poté může provádět podpisy potřebné při atestaci (tzn. vytvořit řetězec důvěry certifikátů), získat správny hash paměti VM a uživatel cloudu nemá šanci přijít na to, že SEV ochrana VM není dostupná. Dalším útokem je Migration Attack, který dokáže získat data běžící VM (používající SEV) z hostujícího systému (např. administrátor jinak důvěryhodné organizace). Prekvizitami tohoto útoku jsou získání CEK+ID platformy a možnost migrace uživatelských VM administračním SW (která je standardně povolena, např. kvůli chybám ve VM). Při migraci je potřeba

opět ověřovat řetězce důvěry a přeposílat certifikáty, přičemž útočník získává přístup k transportním klíčům generovaným v důvěryhodném firmware, pomocí kterých je schopen dešifrovat obsah paměti VM, a to protože zná PDH (Platform Diffie-Hellman klíč), který byl použit k odvození klíčů, které šifrovaly transportní klíče. Posledním útokem je Debug Override Attack, který dokáže získat důvěrná data VM podobným způsobem. V ladícím režimu hypervizor může číst paměť jednotlivých VM, avšak tento režim musí být explicitně povolen vlastníkem VM – to lze obejít opět pomocí přepsání obrazu UEFI vlastním, stejně jako bylo popsáno u Fake SEV. U VM s povoleným SEV je paměť však plně šifrovaná pomocí SME, ale i toto je možné obejít, protože API SEV pro ladění poskytuje dešifrování čtené paměti (tím, že standardně je zapnutí ladícího režimu autentizované a nelze obejít) [4, 17, 18, 1].

Útoky na Intel SGX

Mezi útoky na SGX lze zmínit například Plundervolt, kdy výzkumníci dokázali provádět specificky načasované poruchy paměti při provádění kódu v enklávě, což má za následek únik tajemství z SGX. Útok lze provést i na dálku, ale vyžaduje přístup k privilegované kontrole napětí a frekvenci procesoru. Dále například Prime+Probe útok, který dokáže získat klíče RSA z SGX enklávy za použití určitých instrukcí a zneužívající postranních kanálů DRAM a vyrovnávací paměti. Posledním prezentovaným útokem je SGAxe, který umožňuje útočníkovi přístup k soukromým klíčům CPU použitým pro vzdálenou atestaci. Tzn., že útočník může obejít bezpečnostní opatření společnosti Intel k porušení důvěrnosti enkláv SGX tím, že se může vydávat za legitimní SW ověřený společností Intel [20, 7, 22].

Útoky na ARM TrustZone

Jeden z útoků na ARM TrustZone je například Downgrade. Tento útok využívá chyb v TA, které mohou být klidně zaplátovány novou verzí, která tuto chybu odstraňuje, protože umoňuje načtení staré (chybné) verze TA. K provedení útoku je nutné získat privilegovaný přístup a pouze nahradit novou verzi aplikace za starou. Řešením tohoto problému je používat (generovat) jiné klíče pro každou verzi aplikace. V TrustZone jsou všechny řádky v cache doplněny o NS bit. ARMageddon využívá Prime+Probe útok na postranní kanály cache k extrakci privátních dat z bezpečného světa do normálního světa. Dalším útokem je např. TruSpy, který také využívá Prime+Probe a dokáže získat přímo AES klíče z kompromitovaného OS v normálním světě [5, 19].

Shrnutí

Cílem této práce bylo vytvořit přehledovou studii o TEE a následná krátká analýza a srovnání nejznámějších implementací TEE.

Z dostupných informací je Intel SGX vhodný pro aplikace pracující s malým množstvím velmi důvěrných dat. Nehodí se naopak pro komplexní programy, které by musely být značně refaktorovány.

Naproti tomu AMD SEV cílí na komplexní cloudové služby a infrastruktury, pracující s velkým množstvím dat v nedůvěryhodném prostředí. Neposkytuje tedy tak vysokou míru zabezpečení jako SGX, ale oproti tomu pracuje značně rychleji.

Nakonec byl popsán ARM TrustZone, cílící především na jednoúčelová zařízení (Cortex-M) jako je IoT, či mobilní a přenosná zařízení (Cortex-A). Ten také vyžaduje úpravy, a sice jaký kód má běžet v normálním a jaký v bezpečném světě.

Z nastudovaných materiálů plyne, že vývoj v této oblasti se posunuje mílovými kroky a relativně rychle reaguje na chyby v daných implementacích nalezené.

U AMD lze vidět blízká kooperace s výzkumníky, kteří nahlašují nalezené chyby a rychlá reakce AMD pro zajištění ochrany jejich partnerů pomocí záplat. Jak lze vidět, zvyšující se verze AMD SEV značně zvyšují ochranu vůči různým druhům zranitelností a lze předpokládat, že AMD bude v tomto pokračovat i v dalších letech a více redukovat možnost útoku na AMD SEV chráněné virtuální stroje.

Reference

[1] AMD. Amd sev-snp: Strengthening vm isolation with integrity protection and more. [online], https://www.amd.com/en/processors/amd-secure-encrypted-virtualization, 2020.

[2] AMD. Amd product security. [online], https://www.amd.com/en/corporate/product-security, 2021.

[3] AMD. Amd secure encrypted virtualization (sev). [online], https://developer.amd.com/sev, 2021.

[4] Robert Buhren, Christian Werling, and Jean-Pierre Seifert. Insecure until proven updated: Analyzing AMD sev's remote attestation. *CoRR*, abs/1908.11680, 2019.

[5] Yue Chen, Yulong Zhang, Zhi Wang, and Tao Wei. Downgrade attack on trustzone. 07 2017.

[6] Cfir Cohen. Amd-sev: Platform dh key recovery via invalid curve attack (cve-2019-9836). [online], https://seclists.org/fulldisclosure/2019/Jun/46, 2019.

[7] V. Costan and S. Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016:86, 2016.

[8] Tom Woller David Kaplan, Jeremy Powell. Amd memory encryption. [online], http://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf, 2016.

[9] Don Felton. What is a trusted execution environment (tee)? [online], https://www.trustonic.com/technical-articles/what-is-a-trusted-execution-environment-tee/, 2019.

[10] Don Felton. What is trustzone? [online], https://www.trustonic.com/technical-articles/what-is-trustzone, 2019.

[11] Inc. GlobalPlatform. Introduction to trusted execution environments. [online], https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf, 2019.

[12] Cfir Cohen Google Security Research. Amd secure encrypted virtualization (sev) key recovery. [online], https://packetstormsecurity.com/files/153436/AMD-Secure-Encrypted-Virtualization-SEV-Key-Recovery.html, 2019.

[13] Joffrey Guilbon. Introduction to trusted execution environment: Arm's trustzone. [online], https://blog.quarkslab.com/introduction-to-trusted-execution-environment-arms-trustzone.html, 2018.

[14] Azeria Labs. Trusted execution environments and arm trustzone. [online], https://azeria-labs.com/trusted-execution-environments-tee-and-trustzone, 2020.

[15] Genode Labs. An exploration of arm trustzone technology. [online], https://genode.org/documentation/articles/trustzone, 2021.

[16] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021.

[17] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. A comparison study of intel sgx and amd memory encryption technology. pages 1–8, 06 2018.

[18] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. A comparison study of intel sgx and amd memory encryption technology. [online], https://caslab.csl.yale.edu/workshops/hasp2018/HASP18_a9-mofrad_slides.pdf, 2018.

[19] M. A. Mukhtar, Muhammad Khurram Bhatti, and G. Gogniat. Architectures for security: A comparative analysis of hardware security features in intel sgx and arm trustzone. *2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE)*, pages 299–304, 2019.

[20] A. Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. A survey of published attacks on intel sgx. *ArXiv*, abs/2006.13598, 2020.

[21] Jim Salter. Intel promises full memory encryption in upcoming cpus. [online], https://arstechnica.com/gadgets/2020/02/intel-promises-full-memory-encryption-in-upcoming-cpus, 2020.

[22] Software Guard Extensions. Software guard extensions — Wikipedia, the free encyclopedia. [online], https://en.wikipedia.org/wiki/Software_Guard_Extensions, 2021.

[23] TechDesign. How the security mechanism of microcontrollers secure iot devices. [online], https://blog.techdesign.com/how-security-mechanism-of-microcontrollers-secure-iot-devices, 2020.

[24] Trusted execution environment. Trusted execution environment — Wikipedia, the free encyclopedia. [online], https://en.wikipedia.org/wiki/Trusted_execution_environment, 2021.

[25] Wikichip. Secure memory encryption (sme) - x86. [online], https://en.wikichip.org/wiki/x86/sme, 2021.