Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare

TEZĂ DE DOCTORAT
REZUMAT

Infrastructură cibernetică de încredere

Conducător Științific:
Prof. Dr. Ing. Răzvan-Victor Rughiniș

Autor:
Ing. Florin-Alexandru Stancu

București, 2022

Cuprins

Capitolul 1

Introducere

Calculatoarele moderne sunt suficient de puternice și au mai mult decât suficientă memorie pentru a susține eficient mai multe aplicații în regim multitasking. Programele sunt construite din milioane de linii de cod (de exemplu, mediile grafice, browserele web) care, din cauza acestei complexități, face dificilă (chiar irealizabilă) testarea și validarea împotriva vulnerabilităților de securitate. În plus, sistemul de operare (OS) este, de obicei, însărcinat cu protecția memoriei între aplicațiile din spațiul utilizatorului și de a furniza mecanisme pentru hardware / accesul la rețea, comunicarea între procese etc. Însă un sistem de operare modern este, de asemenea, o bucată mare de software (de exemplu, Linux are acum ajuns la peste 27 milioane de linii de cod [1]) care poate conține vulnerabilități de escaladare a privilegiilor. Termenul de Trusted Computing Base (TCB) a fost inventat pentru a descrie componentele necesare pentru a fi securizate (de exemplu, hardware, firmware, hipervizor, kernel de sistem de operare) pentru întregul sistem să fie de încredere. În mod ideal, TCB trebuie să fie menținut cât mai mic posibil.

Pentru a aborda această problemă, au început să apară tehnologii de execuție de încredere bazate pe hardware [2], care permit programelor să ruleze în contexte de execuție sigure, izolate chiar și de software-ul de sistem privilegiat, reducând astfel drastic TCB-ul aplicației. Printre cele mai populare se numără ARM Security Extensions (TrustZone [3]), Intel Software Guard Extensions (SGX [4]) și AMD Secure Encrypted Virtualization (SEV [5]). Printre posibilele cazuri de utilizare a execuției de încredere, se evidențiază: securizarea serviciilor de cloud computing (asigurarea confidențialității datelor clienților în cazul găzduirii de către terți), sistemele integrate (întărirea dispozitivelor mobile / IoT / industriale împotriva atacurilor cibernetice sau chiar a atacurilor fizice), gestionarea drepturilor digitale (securizarea licențelor și distribuției de software / media), securizarea aplicațiilor informatice critice personale / corporative (de exemplu, autentificarea utilizatorilor, tranzacții financiare de încredere, protejarea confidențialității bazelor de date ale clienților pentru respectarea GDPR).

De reținut că unele dintre tehnologiile de încredere disponibile au fost concepute doar pentru un subset al acestor aplicații. De exemplu, ARM TrustZone este utilizabilă numai în sistemele încorporate în care sunt utilizate astfel de procesoare. Tehnologia SGX de la Intel este disponibilă atât pe servere, cât și pe stații de lucru personale, dar în principal nu dispune de mijloacele necesare pentru a asigura o cale de intrare/ieșire de încredere cu periferice, lăsând pe dinafară multe aplicații care necesită o interacțiune sigură cu utilizatorul. În schimb, SEV de la AMD a fost conceput special pentru serviciile cloud, asigurând memoria mașinilor virtuale împotriva unui hipervizor care nu prezintă încredere.

1.1 Obiectivele Tezei

Tehnologiile de execuție de încredere promit o reîmprospătare a paradigmei de protejare a aplicațiilor sensibile împotriva actorilor privilegiați, cum ar fi un sistem de operare sau un hipervizor care nu prezintă încredere. Producătorii de procesoare (de exemplu, ARM, Intel și AMD) au început deja să încorporeze caracteristici de izolare asistată de hardware în ofertele lor comerciale, marcând o etapă importantă în peisajul securității cibernetice.

Cu toate acestea, există încă mai multe provocări în materie de cercetare pentru ca mediile de execuție de încredere (Trusted Execution Environments - TEE) să poată crește rata de adoptare a acestora. Astfel, definim următoarele obiective pentru teza noastră:

1. Studierea platformelor de execuție de încredere disponibile și evaluarea arhitecturilor, a modelul de securitate, aplicațiile practice și limitările acestora;
2. Soluții pentru asigurarea confidențialității aplicațiilor sensibile în cloud împotriva furnizorilor care nu sunt de încredere;
3. Descrierea cerințelor de dezvoltare și proiectare a aplicațiilor TEE, metode de facilitare a acestui proces;
4. Testarea performanței microcontroller-elor moderne disponibile în comerț pentru dezvoltarea de dispozitive integrate de securitate pentru utilizare în arhitecturi cibernetice de încredere;
5. Examinarea și dezvoltarea de metode de asigurare a accesului I/O securizat (de ex, tastatură, afișaj, imprimante) din mediile de execuție de încredere;
6. Explorarea utilizării tehnologiilor de încredere pentru consolidarea securității sistemelor de control industrial.

1.2 Contribuțiile Tezei

Teza noastră acoperă toate componentele unei infrastructuri de încredere, utilizând tehnologii de execuție de încredere (de exemplu, ARM TrustZone, Intel SGX) pentru a proteja datele sensibile ale aplicațiilor împotriva atacatorilor cu privilegii ridicate (de exemplu, sisteme de operare compromise), subliniind limitările actuale ale acestora (aplicabilitate, portabilitate, I/O de încredere) și contribuind cu soluții la aceste probleme.

Rezumând, principalele noastre contribuții sunt următoarele:

- O trecere în revistă a diferitelor tehnologii de încredere istorice și de ultimă oră, inclusiv cele mai recente soluții disponibile în comerț de la ARM, Intel și AMD.
- Analiza tehnologiilor de încredere în cloud disponibile în prezent și soluția noastră pentru asigurarea confidențialității și integrității aplicațiilor de editare a documentelor în colaborare (de exemplu, Google Docs), utilizând criptarea transparentă pe partea clientului prin intermediul unei extensii de browser.

- Privire de ansamblu asupra provocărilor de proiectare pentru dezvoltarea codului TEE și o prezentare generală a diferitelor SDK-uri și instrumente disponibile pentru platformele populare (ARM TrustZone și Intel SGX). Dezvoltăm *HiddenApp*, o tehnică de proxy pentru apeluri de sistem care facilitează migrarea aplicațiilor bazate pe Linux către o enclavă securizată TrustZone.
- Evaluăm capacitatea de utilizare a dispozitivelor încorporate cu microcontrolere cu resurse limitate într-o infrastructură de încredere, evaluând viteza de procesare și consumul de energie cu biblioteci criptografice moderne.
- O analiză amplă a problemei Trusted I/O Path și o comparație a soluțiilor de ultimă oră disponibile: perifericele acceptate, TEE-urile vizate și componentele TCB.
- Proiectarea unui dispozitiv portabil, încorporat (*TIO*) utilizat pentru a stabili un canal de comunicare securizat între aplicațiile de încredere și dispozitivele periferice bazate pe USB. Demonstrăm capacitățile sale pentru protejarea introducerii de date de la tastatură (de exemplu, parole) și pentru imprimarea în siguranță a documentelor dintr-o enclavă Intel SGX.
- O arhitectură pentru îmbunătățirea securității sistemelor de control industrial critice cu execuție de încredere și un dispozitiv firewall încorporat personalizat utilizat pentru a autentifica comenzile trimise prin rețele industriale tradiționale (utilizând protocolul Modbus).

1.3 Structura Tezei

Teza este structurată astfel:

În Capitolul 2, introducem conceptele de încredere în calculatoare și descriem principalele tehnologii de execuție de încredere: Intel TxT, ARM TrustZone, Intel SGX, AMD SEV, precum și alte lucrări de cercetare conexe.

Capitolul 3 discută cazul cloud-ului de neîncredere: lipsa garanțiilor de confidențialitate atunci când se găzduiesc aplicații pe serverele unor terțe părți. Pe măsură ce analizăm diverse abordări pentru asigurarea fiabilității serviciilor cloud, susținem că tehnologiile de execuție de încredere reprezintă o soluție viabilă pentru asigurarea confidențialității aplicațiilor care procesează date sensibile pe servere de neîncredere, cu suport experimental / viitor din partea tuturor furnizorilor majori de cloud (Amazon, Microsoft, Google, IBM). Ne aducem contribuția noastră, *SecCollab*, o metodă de securizare a documentelor utilizatorilor în aplicațiile web colaborative bazate pe cloud prin utilizarea criptării pe partea clientului.

Capitolul 4 prezintă noile provocări de dezvoltare de software introduse de mediile de execuție de încredere: necesitatea unor modificări arhitecturale, securitatea interacțiunii cu un sistem de operare neîncredere, integrarea serviciilor de încredere (atestare, sigilare, implementări criptografice) și portabilitatea pe alte platforme. Prezentăm cadre de ultimă ge-

nerație pentru a facilita dezvoltarea de aplicații de încredere sau chiar pentru a rula binare nemodificate în interiorul unor enclave securizate cu ajutorul bibliotecilor de abstractizare a sistemului de operare. În cele din urmă, descriem *HiddenApp*, abordarea noastră pentru rularea cu ușurință a aplicațiilor Linux în interiorul contextului securizat al ARM TrustZone, prin pasarea automată a apelurilor de sistem către sistemul de operare bogat pentru execuție.

În Capitolul 5, explorăm utilizarea dispozitivelor încorporate ca platforme de încredere pentru aplicații specifice (IoT, industriale și chiar securitatea PC-urilor). Luăm mai multe microcontrolere ieftine, disponibile în comerț, și evaluăm performanța, memoria și consumul de energie ale diferitelor biblioteci criptografice care implementează algoritmi (atât simetrici, cât și asimetrici) și protocoale (TLS) moderne. Ulterior, ne vom folosi de această lucrare pentru a proiecta mai multe prototipuri hardware care vor contribui la securizarea perifericelor de calculator și a protocoalelor de comunicare industrială cu ajutorul tehnologiilor de încredere.

Prin Capitolul 6, introducem problema Trusted I/O Path (calea de intrare/ieșire de încredere): din cauza faptului că sistemul de operare neîncrezător are, de obicei, acces privilegiat la majoritatea echipamentelor hardware, este necesară în prezent securizarea comunicării dintre TEE-uri și perifericele sistemului, în special atunci când se introduc sau se scot date sensibile. Sistematizăm lucrările de ultimă oră disponibile în 6.2, unde analizăm și comparăm mai multe soluții de I/O de încredere care vizează mai multe clase de dispozitive hardware și platforme de încredere. În 6.3, proiectăm *TIO* - un dispozitiv portabil pentru asigurarea unui canal de intrare/ieșire de încredere între TEE-urile Intel SGX și dispozitivele USB. Folosim TIO pentru a proteja introducerea tastaturii și pentru a securiza imprimarea din enclave.

În Capitolul 7, descriem o arhitectură pentru securizarea sistemelor de control industrial folosind execuția de încredere, propunând un dispozitiv încorporat personalizat cu costuri reduse care să acționeze ca gateway / firewall. Îl folosim pentru a îmbunătăți protocolul Modbus tradițional cu caracteristici de autentificare criptografică pentru a împiedica atacatorii să trimită comenzi malițioase care pot avea consecințe periculoase.

Încheiem în Capitolul 8 cu rezumatul tezei, concluziile, lucrările viitoare și lista contribuțiilor originale.

Capitolul 2

Tehnologii de Calcul de Încredere

Acest capitol descrie cele mai populare tehnologii informatice de încredere și evoluția acestora, începând cu modulul de încredere pentru platforme (TPM) și tehnologia de execuție de încredere (TxT) de la Intel, până la caracteristicile moderne de execuție de încredere încorporate în procesoarele moderne (ARM Trustzone, Intel SGX, AMD Secure Encrypted Virtualization). La final, vom prezenta câteva dintre provocările de cercetare în curs de desfășurare pe care le explorăm pe parcursul acestei teze.

2.1 Trusted Computing Group și modulul TPM

Termenul de "Trusted Computing" ("calcul de încredere") a fost promovat de Trusted Computing Platform Alliance în 1999, redenumit ulterior Trusted Computing Group, cu scopul de a face ca calculatoarele să se comporte într-un mod coerent și sigur, de exemplu, capacitatea de a verifica dacă nu au fost efectuate modificări neintenționate / dacă nu au fost implementate programe malware într-un sistem înainte de a debloca anumite secrete (de exemplu, chei de criptare, date confidențiale). TCG publică specificații pentru o arhitectură de calcul securizată, cea mai populară soluție a acestora fiind modulul de platformă de încredere (Trusted Platform Module - TPM).

TPM este un criptoprocesor încorporabil, adesea integrat în calculatoarele moderne (de exemplu, PC-uri, laptopuri, servere) pentru a le îmbunătăți securitatea. Acesta oferă un motor criptografic, o serie de registre de configurare a platformei (PCR) și o memorie nevolatilă (NVRAM) rezistentă la manipulare. Aceste caracteristici au fost concepute pentru a fi combinate între ele, rezultând o serie de funcții de securitate de nivel înalt: măsurarea integrității codului și a datelor (înainte de execuție), stocarea sigilată (blocarea / criptarea datelor într-o anumită stare a platformei) și atestarea la distanță (dovedirea integrității platformei în fața unei terțe părți înainte de furnizarea de secrete).

2.2 Mediile de execuție de încredere

.

Un mediu de execuție de încredere (Trusted Execution Environment - TEE) poate fi descris ca fiind o regiune de memorie izolată și un context de execuție a procesorului în care programele pot rula cu protecții de confidențialitate și integritate asigurate de platformă față de mediul normal (bogat), în special față de componentele cu privilegii superioare

(nucleul sistemului de operare, hipervizorul, firmware-ul sau chiar manipularea fizică), adesea considerate ca fiind de neîncredere.

Există mai multe abordări pentru implementarea unui TEE: numai software (de exemplu, Virtual Ghost [6], SofTEE [7]) și asistată de hardware (de exemplu, virtualizare securizată, modificări arhitecturale ale CPU care introduc contexte de execuție izolate). Ne concentrăm pe acestea din urmă și rezumăm tehnologiile de încredere disponibile în prezent în comerț.

2.2.1 Intel Trusted Execution Technology (TxT)

Intel Trusted Execution Technology (TxT [8]), una dintre primele tehnologii de execuție a tehnologiilor de securitate disponibile pe piață tehnologii de încredere, permite unui hipervizor de încredere să se lanseze la o fază târzie a procesului de boot (după ce sistemul de operare normal a pornit, excluzând întregul său lanț de pornire de la TCB al platformei). Intel TxT utilizează, de asemenea, un modul auxiliar Trusted Platform Module pentru a asigura măsurarea integrității, sigilarea datelor și atestarea de la distanță caracteristici.

Se spune că această caracteristică de lansare târzie alcătuiește o rădăcină dinamică de încredere pentru măsurători (DRTM) datorită faptului că integritatea sa nu este afectată de nici o cod executat anterior, spre deosebire de cea statică din cadrul unei platforme inițiale la rece. boot. Odată ce mașina virtuală a fost pornită și măsurată, resursele sale vor fi izolate de funcțiile de protecție a virtualizării hardware a procesorului. Sistemul de operare normal (neîncredere) poate continua să ruleze cot la cot cu acesta, în afara bazei de calcul de încredere (TCB) a sistemului.

2.2.2 ARM TrustZone

ARM TrustZone [3] este o extensie arhitecturală pentru microprocesoarele ARM care asigură o separare forțată de hardware între două domenii: Lumea Securizată, unde poate fi implementat un mediu de execuție de încredere, și Lumea Normală, unde se află stiva de software bogat (adică sistemul de operare și aplicațiile utilizatorului care nu sunt de încredere). Acest lucru a fost realizat prin adăugarea unui nou context de securitate, complementar nivelurilor tradiționale de privilegii (kernelspace și userspace). Arhitectura TrustZone impune, de asemenea, izolarea memoriei cu ajutorul unei unități de gestionare a memoriei (MMU) îmbunătățite, precum și a unui controler de întreruperi modificat pentru a asigura gestionarea prioritară a întreruperilor în lumea securizată. În plus, un sistem pe cip (SoC) ARM poate include periferice care cunosc TrustZone (de exemplu, memorie RAM statică/dinamică suplimentară sau memorie flash), care pot utiliza steguleţele de execuție pentru a lua decizii de autorizare la nivel hardware pentru tranzacțiile lor pe magistrală.

Comutarea între cele două lumi se poate face numai prin utilizarea unui nivel de excepție special, extra-privilegiat, numit Monitor Mode, în care un gestionar instalat în firmware este responsabil de validarea și salvarea/restabilirea în siguranță a contextelor CPU în mod corespunzător. Acest proces are loc în mod automat, atunci când se primește o întrerupere

hardware care trebuie gestionată de un alt domeniu de securitate, sau manual, atunci când software-ul utilizează noua instrucțiune Secure Monitor Call (SMC) pentru a accesa serviciile de încredere (similar cu Supervisor Call utilizat pentru a trece de la modul utilizator la modul kernel).

2.2.3 Intel Software Guard Extensions (SGX)

Cu **Software Guard Extensions** [4], Intel a dezvoltat un nou mediu de execuție de încredere pentru familia de procesoare x86 de uz general. Folosind SGX, aplicațiile din spațiul utilizatorului pot rula în contexte de execuție speciale numite *Enclave*, cu protecții la nivel hardware (inclusiv criptarea RAM) împotriva unui sistem de operare privilegiat care să le citească memoria sau să le modifice memoria fluxul de execuție, rezultând astfel un TCB minim format doar din CPU hardware, microcodul său / firmware-ul încorporat și enclava software.

Din punctul de vedere al utilizatorului, o aplicație SGX instalează și lansează aceeași ca orice alt program al sistemului de operare. Pentru un dezvoltator, aplicația trebuie să fie împărțită în două componente majore: codul enclavei (care va fi executat în interiorul TEE), și programul de neîncredere (utilizat inițial pentru a încărca enclava și să furnizeze servicii de neîncredere ale sistemului de operare, de exemplu, rețea, sistem de fișiere, accesul la periferice). Aplicațiile nesigure și enclavele lor pot comuta între ele folosind Enclave apeluri și apeluri externe. Atunci când enclava este încărcată, codul său, împreună cu datele și metadatele inițiale (de exemplu, numărul de versiune, cheia publică a dezvoltatorului) va fi automat hașurat de către hardware-ul în interiorul mai multor registre speciale de măsurare, care sunt exclusiv descrie în mod unic enclava care rulează în interiorul mediului de încredere. Aceste măsurători pot fi utilizate pentru atestarea locală / la distanță și pentru sigilarea datelor scopuri de sigilare a datelor.

2.2.4 AMD Secure Encrypted Virtualization

Începând cu noile lor procesoare pentru servere EPYC (bazate pe Zen), **Secure Encrypted Virtualization** (SEV [5]) este cea mai recentă incursiune a AMD în domeniul tehnologiilor de securitate hardware. AMD SEV funcționează prin criptarea transparentă a memoriei RAM a mașinii virtuale invitate pentru a o face inaccesibilă hipervizoarelor și chiar atacatorilor fizici, transformând efectiv instanțele virtualizate în enclave sigure.

Acesta utilizează un controler de criptare securizată a memoriei (SME) on-die pentru a intercepta toate tranzacțiile DRAM și pentru a le cripta / decripta fără probleme. Acest lucru se poate face, de asemenea, într-o manieră mai granulară prin utilizarea unui nou bit *enCrypted* în interiorul structurii paginii pentru activarea / dezactivarea acestui proces. Generarea și gestionarea cheilor are loc în interiorul procesorului securizat suplimentar al procesorului (AMD-SP), un nucleu de microcontroler bazat pe ARM utilizat pentru funcționalitatea platformei de încredere. Acest procesor criptografic oferă, de asemenea, un TPM bazat pe firmware și un API utilizabil de către hipervizoare pentru a genera chei de criptare pentru mașinile lor virtuale, deși nu le pot vedea efectiv.

În timpul lansării, imaginea inițială a unei VM este încărcată de către hipervizor sub formă de pagini necriptate. Procesorul securizat ia, de asemenea, o măsurătoare criptografică a conținutului și metadatelor sale, similar cu PCR-urile unui TPM. Mașinile virtuale locale și părțile terțe de la distanță pot solicita apoi un raport de atestare pentru a autentifica și furniza secretele inițiale către mașina virtuală de încredere.

2.3 Provocări în domeniul Tehnologiilor de Încredere

Subliniem existența unui compromis software/hardware în majoritatea tehnologiilor de execuție de încredere: abordările hard, deși oferă o performanță și o reziliență sporite printr-o mai bună izolare a memoriei, suferă de o flexibilitate redusă, în special în cazul vulnerabilităților de securitate: dacă sunt (sau când sunt) descoperite, erorile la nivel de arhitectură tind să fie greu de corectat, adesea prin utilizarea de patch-uri de microcod și/sau ale compilatorului (de exemplu, pentru a preveni canalele secundare) pot duce la pierderi de performanță. Proiectarea unui TEE cu o combinație adecvată de primitive hardware validate din punct de vedere al securității și o bună flexibilitate firmware rămâne o provocare în continuare.

Enclavele securizate au un domeniu de aplicabilitate larg: calcul mobil, cloud, stații de lucru și chiar calcul industrial. Anumite tehnologii bazate pe hardware sunt utilizabile numai în domeniile în care sunt disponibile astfel de unități centrale de procesare (de exemplu, AMD SEV pentru servere, ARM TrustZone pentru aplicații integrate). În teza noastră, abordăm pe scurt utilizarea acestor tehnologii în problema cloud-ului lipsit de încredere în Capitolul 3 (*Servicii cloud demne de încredere*) și securitatea industrială în Capitolul 7. (*Arhitectura industrială fiabilă*).

Observăm că există un cost suplimentar de dezvoltare atunci când se scriu aplicații TEE: deoarece sistemul de operare nu este de încredere, toate serviciile furnizate în mod normal de acesta (de exemplu, sistemul de fișiere, rețelele, I/O periferice, comunicarea între procese) trebuie fie securizate separat, de obicei prin utilizarea criptografiei (de exemplu, validarea rezultatelor API, utilizarea protocoalelor de transport criptate), fie, în unele cazuri, trebuie reimplementate unele dintre ele (de exemplu, gestionarea memoriei, ceasuri de încredere). Aceasta a fost ținta unor cercetări extinse, rezultând mai multe abordări, cum ar fi bibliotecile de abstractizare a sistemului de operare sau cadrele de validare a apelurilor de sistem pentru toate platformele TEE comerciale; le vom descrie în Capitolul 4.

În cele din urmă, una dintre cele mai importante probleme rămase nerezolvate pentru tehnologiile de încredere este dificultatea de a asigura căi de intrare/ieșire de încredere cu dispozitivele periferice ale unui sistem. Acest lucru este deosebit de important pentru susținerea aplicațiilor interactive (de exemplu, pentru protejarea intrărilor de la tastatură, a ieșirilor de pe ecran). Unele tehnologii TEE (e.g., ARM TrustZone) permit acest lucru, însă rămâne o provocare pentru platformele de uz general (de exemplu, x86). Aceasta este, de asemenea, o cerință pentru sistemele industriale, unde interfețele ciberfizice sunt omniprezente. Discutăm aceste aspecte în Capitolul 6.

Capitolul 3

Servicii Cloud de Încredere

În ultimul deceniu, omniprezența accesului la internet și evoluția tehnologiilor web au dus la apariția serviciilor și aplicațiilor bazate pe cloud. Cu toate acestea, datele utilizatorului stocate acolo sunt ușor de accesat de către furnizorii de cloud, cu implicații importante în materie de securitate și confidențialitate. Acest capitol abordează problema norului lipsit de încredere și se orientează către o arhitectură de încredere pentru aplicațiile online.

3.1 Starea actuală a soluțiilor cloud de încredere

3.1.1 Criptarea pe partea clientului

Cea mai simplă soluție pentru a asigura protecția datelor într-un mediu cloud ar fi să se facă o **criptare pe partea clientului**, în care serverul nu va cunoaște conținutul real al datelor unui utilizator. Anumite aplicații permit chiar ca acest lucru să fie realizat într-un mod transparent, de exemplu: un serviciu terț de stocare/sincronizare a fișierelor în cloud, cum ar fi Dropbox, poate fi utilizat cu un sistem de fișiere virtual care oferă o vizualizare criptată serverului cloud.

Din nefericire, această metodă funcționează numai atunci când serverul nu are nevoie să efectueze niciun calcul asupra datelor pentru ca funcțiile aplicației să funcționeze. Criptarea Homomorfă Completă (FHE) este o nouă schemă promițătoare pentru rezolvarea acestei probleme, dar, din păcate, nu este întotdeauna potrivită pentru toate scenariile, având în vedere caracteristicile sale de flexibilitate și performanță mai puțin ideale [9].

3.1.2 Platforme Cloud de Încredere

În afară de criptarea pe partea clientului, **Tehnologiile informatice de încredere** au fost mult timp dorite de industrie pentru a oferi o soluție accesibilă pentru asigurarea confidențialității și integrității clienților. Mediile de execuție de încredere, cum ar fi Intel SGX și AMD SEV, pot oferi o izolare sporită față de utilizatorii cu acces privilegiat (de exemplu, administratorii de sistem ai serverelor) prin excluderea întregii stive de virtualizare (hipervizor) și, opțional, a sistemului de operare din TCB-ul lor. Furnizorii populari de cloud (Google, Microsoft, IBM, Red Hat) au început să le adopte, anunțând viitoarele caracteristici confidențiale de cloud pentru VM-urile lor [10].

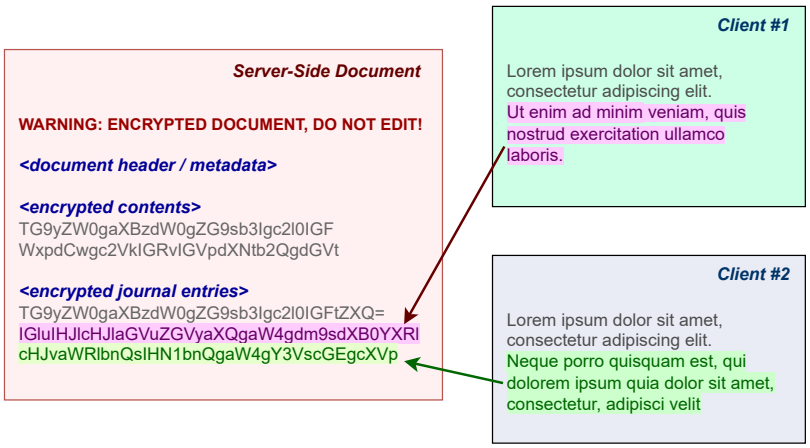


Figura 3.1: Structura documentului colaborativ criptat.

3.2 SecCollab - Îmbunătățirea confidențialității pentru editoarele colaborative de documente cloud

Prezentăm o metodă de asigurare a confidențialității și integrității pentru aceste aplicații, păstrând în același timp caracteristica lor de bază, aceea că mai mulți autori pot edita în colaborare același document în timp real, de exemplu, Google Docs.

Astfel, vor exista două versiuni ale documentului: pe server și pe client, ilustrate în figura 3.1. Noi folosim documentul de pe partea serverului doar ca mecanism de stocare și sincronizare criptat, bazat pe protocolul său colaborativ. Luăm fiecare transformare operațională efectuată de utilizator asupra documentului său decriptat și o adăugăm în interiorul unui bloc special de jurnal criptat. Operațiunile obișnuite efectuate pe care serverul le va vedea aici sunt doar adăugiri. Atunci când jurnalul devine prea mare, un client va achiziționa un mutex (blocare) asupra documentului și va executa o operațiune specială de snapshotting (compactare).

Am implementat un addon de probă de concept pentru serviciul Google Docs sub forma unei extensii de browser, care acționează ca agent middleware între pagina web și server, interceptând și modificând traficul realizat prin AJAX (Asynchronous JavaScript and XML) astfel încât serverul să primească doar date criptate.

În implementarea noastră, fiecare intrare în jurnal este un șir JSON (care conține metadate de transformare) care cântărește aproximativ 40 octeți. În plus, am utilizat un algoritm de criptare AES256-GCM folosește un vector de inițializare pe 96 de biți plus o etichetă de autentificare pe 64 de biți, cu o greutate totală de $16bytes$. Codificarea Base64 are, de asemenea, o suprataxă dată de formula $ceil(n/3) * 4$ plus o umplutură (1-4 octeți), astfel încât dimensiunea rezultată a unui bloc de jurnal criptat este de $ceil((41+16)/3)*4+pad = 76 + pad = 80$ octeți. Procesul nostru de fotografiere instantanee limitează dimensiunea maximă a jurnalului la $33.6KB$.

Capitolul 4

Dezvoltarea Aplicațiilor de Încredere

În acest capitol, descriem provocările de dezvoltare a aplicațiilor introduse de execuția de încredere: cerințele pentru apelurile API de încredere față de cele de neîncredere, abstracțiile pentru software-ul TEE transplatformat și straturile de compatibilitate de ultimă generație pentru rularea programelor normale, nemodificate, în cadrul TEE-urilor.

4.1 Modelul de dezvoltare

În mod normal, aplicațiile existente care doresc să utilizeze tehnologiile de execuție de încredere vor necesita o reproiectare, și anume, codul lor va trebui să fie împărțit în două: o componentă de încredere, critică pentru securitate (pentru a fi executată în interiorul mediului securizat) și o parte nesigură (pentru execuție nesigură în interiorul unui sistem de operare bogat). În plus, fiecare platformă TEE utilizează o arhitectură și o API diferite pentru dezvoltarea de aplicații de încredere. Astfel, este necesar un efort de dezvoltare mai mare din partea furnizorilor care doresc să susțină mai multe TEE, ceea ce se traduce prin creșterea costurilor. Pentru a atenua această problemă, se pot utiliza abstracțiuni reutilizabile (de exemplu, sub formă de biblioteci), deși acestea pot crește dimensiunea TCB.

O altă problemă este legată de cerințele suplimentare de proiectare a software-ului de încredere. O aplicație tipică utilizează interfața de apel de sistem a nucleului sistemului de operare pentru accesarea sistemului de fișiere, comunicarea cu alte procese sau prin rețea, citirea datelor de intrare de la utilizatori și afișarea rezultatelor pe ecran etc. O aplicație securizată ar putea avea nevoie de unele dintre aceste caracteristici și va trebui fie să fie implementată de mediul de încredere, fie, pentru unele dintre ele, să fie transmisă sistemului de operare pentru execuție.

Există soluții, de obicei prin adăugarea unui strat de compatibilitate cu sistemul de operare în cadrul TEE pentru a delega apelurile de sistem fie către serviciile platformei de încredere, fie chiar către sistemul de operare nesigur (și securizarea interacțiunii prin alte mijloace), așa cum prezentăm în secțiunea următoare.

4.2 Cadre de Dezvoltare în TEE

Cadrele pentru dezvoltarea unui mediu de execuție de încredere sunt un subiect de cercetare fierbinte. Această secțiune trece în revistă astfel de soluții grupate în funcție de platforma vizată. De asemenea, prezentăm câteva straturi de abstractizare cross-TEE care ajută dezvoltatorii să vizeze mai multe platforme în același timp.

Primul, ARM TrustZone, necesită dezvoltarea unui firmware de încredere special (bootloader, monitor handler special și microkernel de încredere). Cadre precum PrivateZone [11] permit dezvoltatorilor să divizeze cu ușurință programul și să ruleze părți din aplicațiile lor în interiorul lumii securizate, în timp ce alte lucrări se concentrează pe aducerea unui suport specializat pentru limbajele de programare pentru TEE-uri (de exemplu, .NET Framework, Rust).

TrustShadow [12] descrie o metodă de a rula binare Linux nemodificate în interiorul lumii securizate folosind o schemă transparentă de redirecționare a apelurilor de sistem: aplicațiile din interiorul TEE au cererile lor transmise prin proxy către sistemul de operare bogat, executate, apoi rezultatele sunt copiate înapoi. Lucrarea noastră, *HiddenApp*, realizează, de asemenea, proxyarea apelurilor de sistem pentru rularea programelor Linux în interiorul mediului de încredere. Principala diferență constă în faptul că noi folosim un proces din spațiul utilizatorului care nu este de încredere (wrapper) ca gateway pentru apeluri, într-un mod similar cu abordarea OCall a Intel SGX.

Pentru enclavele din spațiul utilizatorului Intel SGX, dezvoltatorii trebuie să scrie în mod special o bibliotecă cu codul care să ruleze în contextul TEE. Au fost create mai multe SDK-uri pentru a facilita acest lucru (OpenSGX, Intel SGX SDK). Rularea aplicațiilor nemodificate în interiorul enclavelor Intel SGX este posibilă prin utilizarea unor sisteme de operare de bibliotecă, cum ar fi Haven [13] pentru aplicații Windows, SCONE [14] pentru containere Linux precum Docker și Graphene-SGX [15], care rulează binare POSIX nemodificate. Acestea permit portarea cu ușurință a aplicațiilor în enclave, în detrimentul unei complexități crescute a TCB.

4.3 HiddenApp - Securizarea Aplicațiilor Linux folosind ARM TrustZone

HiddenApp este soluția noastră pentru a porta cu ușurință programele existente într-un mediu de încredere ARM TrustZone [3]. Reamintim că lumea securizată TrustZone are, de asemenea, niveluri de privilegii separate pentru modurile utilizator / kernel. Astfel, am dezvoltat un microkernel care oferă servicii securizate, funcționând în același timp și ca un strat de abstractizare a sistemului de operare, care interceptează apelurile de sistem din TEE, le transmite către sistemul de operare normal (Rich) pentru execuție și apoi returnează rezultatele acestora către aplicația de încredere. Rețineți că codul sursă al aplicației nu necesită nicio modificare, oferind o modalitate ușoară de protejare a acesteia în fața unui sistem de operare de neîncredere și de minimizare a TCB.

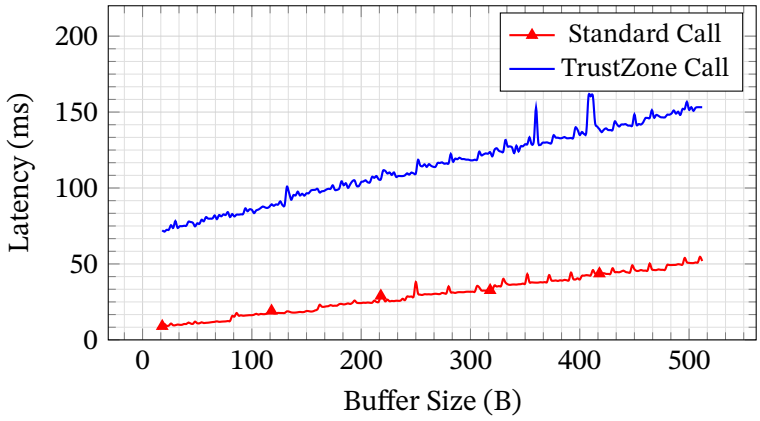


Figura 4.1: Duratele syscall-urilor de write pentru buffere de dimensiuni variate [16].

Soluția noastră utilizează mai multe componente. Când sistemul pornește, un încărcător securizat va configura corespunzător memoria, controlerul de întreruperi și alte periferice conștiente de TrustZone (de exemplu, consola serială, intrarea tactilă sau afișajul) și va instala un gestionar de mod de monitorizare (pentru Secure Monitor Calls – `SMC`, schimbarea contextului între lumea securizată / normală). Rețineți că, cu ARM TrustZone, un segment din memoria fizică a SoC-ului va fi rezervat pentru lumea securizată și va fi inaccesibil pentru sistemul de operare normal. Firmware-ul de încredere poate bloca mai multe întreruperi care să ajungă la sistemul de operare normal (de exemplu, pentru a intermedia accesul pentru dispozitivele de încredere). După aceea, atât **Microkernelul de încredere**, cât și **Sistemul de operare bogat** sunt lansate în contextele lor de execuție respective. Sistemul de operare normal (un nucleu GNU/Linux) primește controlul și așteaptă ca aplicațiile de încredere să fie rulate de către utilizator.

Atât *Microkernelul de încredere*, cât și *Modulul Kernel de comunicare din zona de încredere* vor trebui să transmită/dezmembreze datele de apelare a sistemului utilizând o structură de pachete personalizată. Astfel, acestea au nevoie de un cod de abstractizare pentru toate rutinele de serviciu Rich OS disponibile: ordinea parametrilor, tipurile de date, direcțiile/lungimile bufferului și comportamentul lor general (adică API-ul de spațiu utilizator al kernelului). Fiecare apel de sistem trebuie să aibă, de asemenea, verificări adecvate pentru a preveni accesul neautorizat (de exemplu, atacurile Iago) sau vulnerabilitățile de depășire a bufferului. În acest nivel pot fi implementate și caracteristici de încredere personalizate, de exemplu, criptarea transparentă a sistemului de fișiere.

Am implementat și am testat soluția noastră pe o placă de dezvoltare NXP i.MX53 alimentată de Linux v2.6.35, folosind un U-Boot modificat ca bootloader pentru a realiza inițializarea platformei TrustZone și pentru a porni sistemele de operare. Microkernelul nostru a fost proiectat pentru a fi cât mai mic posibil. Acest lucru a dus la un TCB de $\approx$ 2000 linii de cod (C + asamblare), cu excepția bibliotecii de criptografie (LibTomCrypt) utilizată pentru verificarea semnăturii digitale bazate pe RSA, cu o amprentă mare (30 *KB* ca fișier binar). L-am validat prin testarea unor programe simple care necesită interacțiune cu sistemul de operare, cum ar fi citirea/scrierea sistemului de fișiere, un client telnet și un client ssh.

Latența apelurilor de sistem este prezentată în Tabelul 4.1. O problemă evidentă este reprezentată de un supratax mare de ≈ 49 *ms* între apelurile de sistem normale și cele proxy, în principal din cauza numeroaselor schimbări de context în modul Supervisor și Monitor efectuate în acest proces. Putem observa că timpul este amortizat atunci când execuția apelului de sistem durează mai mult, cum este cazul rutinei `write` Deși diferența de performanță este destul de mare, punem accentul pe îmbunătățirile de securitate rezultate din protejarea datelor sensibile ale aplicațiilor împotriva atacatorilor puternici și considerăm costurile ca fiind acceptabile. Optimizări de viteză sunt încă posibile, deși le lăsăm pentru lucrări ulterioare.

Tabela 4.1: Latențe medii ale apelurilor [16].

Nume	Standard (ms)	TrustZone (ms)	Overhead (%)
getpid	0.58	49.33	8325 %
socket	12.6	61.5	384 %
write (256B)	29.60	114.27	386 %
write (512B)	51.92	153.19	295 %

Capitolul 5

Utilizarea dispozitivelor incorporate în arhitecturi de încredere

În mod tradițional, dispozitivele încorporate nu dispun de o securitate modernă din cauza capacităților lor limitate (putere de procesare, memorie și dimensiunea codului firmware). Criptografia este adesea necesară pentru a asigura securitatea informațiilor în astfel de sisteme integrate, dar costul de implementare este adesea considerat ridicat. Din cauza puterii de calcul și a memoriei exigente pe care le necesită acești algoritmi și a resurselor limitate disponibile din partea hardware-ului, dezvoltatorii au folosit anterior practici de securitate inadecvate sau chiar le-au omis complet. În prezent, pe măsură ce tehnologia se îmbunătățește continuu, dispunem de procesoare mai rapide, mai mici și mai eficiente din punct de vedere energetic, până în punctul în care costurile de adăugare a securității în aplicațiile integrate au devenit suficient de scăzute pentru a fi fezabile, justificate chiar și pentru produsele de nivel inferior.

În acest capitol, testăm mai multe platforme moderne de microcontrolere, evaluând diverși algoritmi și biblioteci criptografice pe mai multe dispozitive, pentru a ajuta la alegerea celui mai potrivit produs pentru un proiect eficient din punct de vedere al costurilor, dar sigur. Lucrarea noastră este utilizată mai târziu în această teză pentru proiectarea dispozitivelor încorporate utilizate pentru a spori infrastructurile de încredere cu caracteristici lipsă (cum ar fi Trusted I/O și firewall-ul sistemelor industriale de încredere).

5.1 Evaluarea primitivelor criptografice pe platforme de microcontrollere

Am evaluat performanța mai multor biblioteci open-source comercializate pentru utilizare încorporată, selectând algoritmi criptografici populari (criptare simetrică și asimetrică, hashing și autentificare) pe mai multe procesoare ARM Cortex-M cu costuri reduse. Am implementat un firmware modular în C care ne-a permis să le testăm pe toate, să raportăm și să comparăm rezultatele.

Comparația globală a performanțelor algoritmilor simetrici este prezentată în Figura 5.1.

Ultimul algoritm care a fost testat, cifrul asimetric RSA (Tabelul 5.1) a avut cel mai mare impact asupra sistemelor integrate. Nicio implementare nu a putut rula pe EFM32ZG din cauza dimensiunii programului care depășea memoria flash a acestuia (40KB față de 32KB),

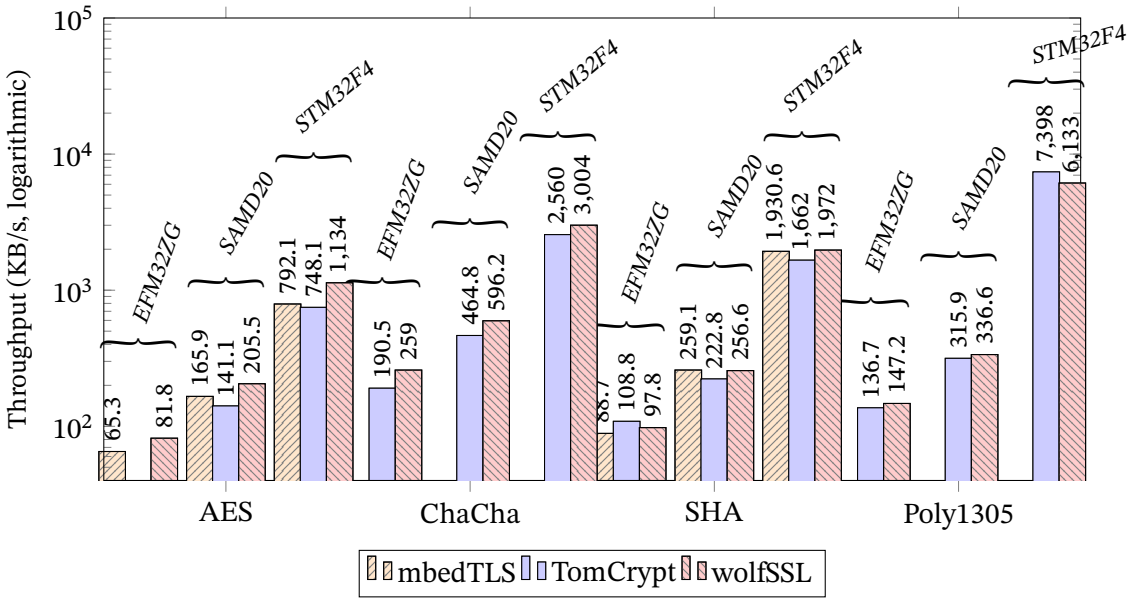


Figura 5.1: Throughtput comparison of AES, ChaCha, SHA and Poly1305 on all platforms

Platform	Library	Prog.Size (Bytes)	Memory (Bytes)	Decrypt (ms)	Encrypt (ms)
SAMD20	mbedTLS	46160	5412	830.6	44.1
SAMD20	TomCrypt	42696	8536	3335.4	340.3
SAMD20	wolfSSL	44720	3272	3699.5	364.3
STM32F4	mbedTLS	44628	6978	88.6	4.2
STM32F4	TomCrypt	41256	10112	274.0	22.8
STM32F4	wolfSSL	42360	4856	348.4	28.8

Tabela 5.1: Rezultatele algoritmului asimetric RSA (1024 bit).

indiferent de optimizările pe care le-am încercat. La celelalte două microcontrolere, o singură operațiune de 128 de octeți a necesitat mult mai multe cicluri pentru a fi finalizată, așa cum era de așteptat din partea sistemului de criptografie RSA, care este foarte greu de utilizat.

În cele din urmă, cerințele de putere sunt prezentate în Figura 5.2. Curentul consum a fost în esență același pentru toate implementările de pe o platformă, indiferent de algoritmul executat, astfel încât tabelul include doar consumul combinat de energie electrică. rezultatele combinate.

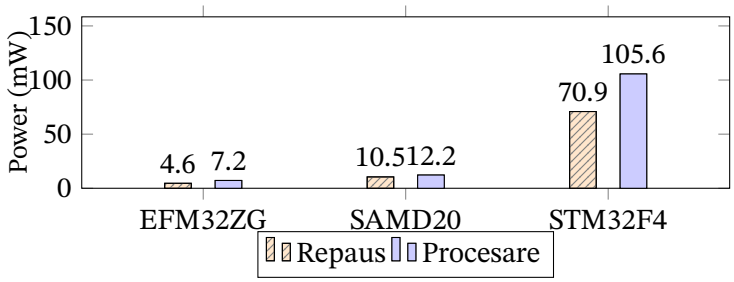


Figura 5.2: Consum (repaus versus execuție).

5.2 Considerații ale consumului de energie a protocolului TLS în dispozitive IoT

.

Dispozitivele comerciale de tip Internet of Things (IoT) ar trebui protejeze confidențialitatea utilizatorilor și să împiedice controlul de către atacatori de la distanță, însă, de obicei, nu reușesc să facă acest lucru din cauza măsurilor de securitate insuficiente sau a implementărilor necorespunzătoare. Multe protocoale de aplicații specifice IoT care sunt optimizate pentru o complexitate și o utilizare redusă a resurselor, cum ar fi CoAP sau MQTT. Aceste protocoale se bazează pe alte straturi pentru a asigura securitatea, cum ar fi un canal de comunicare securizat asigurat de Transport Layer Security (TLS), care este rareori utilizat din considerente legate de consumul de energie.

Lucrarea noastră măsoară impactul energetic al utilizării protocolului Transport Layer Security (TLS) în dispozitive IoT cu conectivitate WiFi (am testat un dispozitiv de dezvoltare Espressif ESP32 board) și demonstrăm că, în condiții reale este, de fapt, neglijabil.

Am folosit un multimetru digital interfațat cu un computer pentru a măsura consumul de curent al dispozitivului testat, împreună cu un osciloscop conectat în paralel pentru a capta vârfuri mai precise (datorate) și având un canal secundar conectat la un GPIO, semnalizând diferitele stări ale protocolului (ilustrat în Figura 5.3).

Utilizând metodologia de mai sus, performanța dispozitivului (timpul de procesare și utilizarea curentului) este evaluată folosind mai multe scenarii. În primul rând, valorile de bază ale curentului vor fi măsurate pentru diferite stări de bază și moduri de funcționare: inactivitate, o buclă NOP și algoritmii vizați într-un mediu sintetic (adică fără activitate de rețea, cu radioul oprit). Aceste valori de bază sunt, apoi, potrivite cu formele de undă complexe obținute la rularea unei sesiuni TLS reale.

Contribuțiile la consumul de energie au fost reprezentate grafic în figura 5.4. Observăm că radioul (în modul de recepție) necesită doar aproximativ jumătate din energia totală, restul (energie statică) fiind utilizat de microcontroler doar fiind activ.

În ceea ce privește cheltuielile TLS, operațiunea de strângere de mână este semnificativ mai costisitoare decât schimbul de date al aplicației, care se realizează cu ajutorul unui

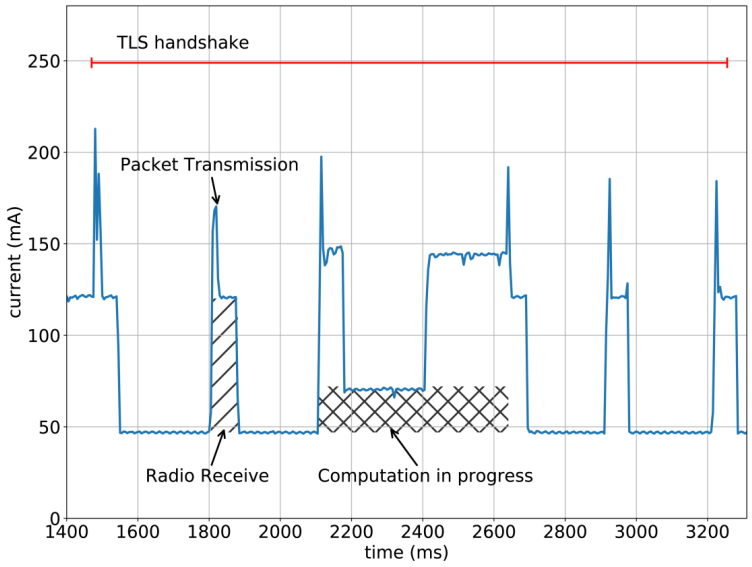
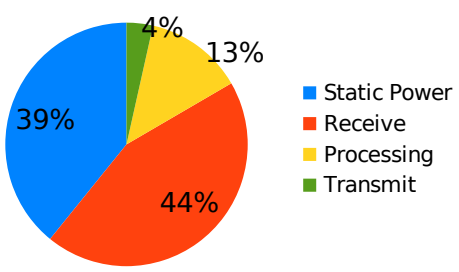
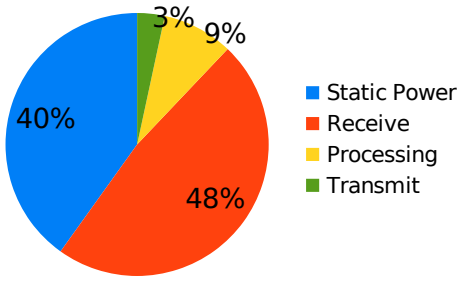


Figura 5.3: Stabilire conexiune TLS - captură DMM [17]

cifru simetric. În cazul aplicațiilor care schimbă volume mari de date, caracterul unic al handshake-ului face ca aportul său global să fie neglijabil. În cazul aplicațiilor care transmit date ocazional, poate fi avantajos să se deconecteze de la rețeaua fără fir și să plaseze dispozitivul într-o stare de veghe cu consum redus de energie, dar conexiunea trebuie restabilită, ceea ce forțează un cost semnificativ al handshake-ului.



(a) Negociere Sesiune TLS



(b) Recepție TLS

Figura 5.4: Contribuțiile de consum ale TLS [17]

Capitolul 6

Soluții I/O de încredere

6.1 Problem Overview

În mod normal, sistemul de operare este responsabil pentru interfațarea cu lumea externă (de exemplu, rețea, stocare, periferice). Programele care rulează în interiorul unui mediu de execuție de încredere pot utiliza în mod liber serviciile sistemului de operare, dar nu sunt de încredere. Cu toate acestea, unele cazuri de utilizare pot necesita un acces securizat la anumite componente hardware. De exemplu, tastatura este adesea utilizată ca metodă de autentificare (de exemplu, prin intermediul unor fraze de trecere). Din păcate, unele platforme populare (de exemplu, PC-urile) nu dispun de mijloace integrate de stabilire a unor căi de comunicare de încredere cu perifericele sale, despre care vom discuta în acest capitol.

Multe aplicații critice din punct de vedere al securității necesită interacțiunea cu perifericele (de exemplu, tastatură, afișaj, ecran tactil) sau cu unele alte dispozitive (de exemplu, echipamente industriale conectate prin adaptoare seriale). Aceste aplicații ar beneficia foarte mult dacă ar fi izolate în interiorul unui mediu de execuție de încredere, dar modul obișnuit în care interacționează cu hardware-ul este utilizarea serviciilor sistemului de operare care nu sunt de încredere (prin intermediul driverelor de dispozitiv). Pentru a se proteja împotriva acestui lucru, fie trebuie să se refuze accesul la perifericele hardware specifice din partea sistemului de operare, fie trebuie să se stabilească un canal de comunicare de încredere cu TEE-ul aplicației, astfel încât un kernel Man-in-the-Middle rău intenționat să nu poată interveni. Acest lucru este definit ca fiind problema *Trusted I/O Path* [18] și există mai multe abordări pentru rezolvarea acesteia, în funcție de clasa dispozitivului periferic și de caracteristicile disponibile ale platformei.

6.2 Sistematizarea soluțiilor I/O de încredere pentru TEE-uri

În această secțiune, vor fi prezentate mai multe lucrări din domeniul Trusted I/O Path, evidențiind conceptele de noutate, diferențele notabile și îmbunătățirile acestora. Rețineți că, pentru fiecare dintre articole, a fost atribuit un pseudonim dintr-un singur cuvânt, care va fi utilizat în restul lucrării pentru o identificare ușoară.

După cum este prezentat în tabelul 6.1, soluțiile disponibile pentru căi de încredere sunt diverse, cu diferite clase de aplicații și cerințe de platformă. Am utilizat o clasificare multilaterală în funcție de: tipul de periferic (Dispozitiv de introducere a datelor umane / Afișaj

Name	Isolation	Interface	HID	Display	Others	Target TEE	+Hypervisor	+Chip/firmware	+Ext. Dev	TCB LoC
			Peripherals				TCB Additions			
ZTIC [19]	Ext	USB	●	◐	○	*Remote*	✗	✗	✓	$\approx$ 110KB [1]
Bumpy [20]	Ext	USB	●	○	○	Flicker	✗	✗	✓	$\approx$ 8.5k
Bumpy Display	Ext	Network	○	◐	○		✗	✗	✓	$\approx$ 10k
UTP [21]	Virt	Driver	●	◐	○	Flicker	✓	✗	✗	$\approx$ 2.3k
VTP x86 [18]	Virt	MMIO	●	◐	●	TrustVisor	✓	✗	✗	$\approx$ 15k
Intel PAVP [22]	Chip	GPU	○	●	○	DRTM, SGX	✗	✓	✗	n/a
TrustUI [23]	Virt	Driver	◐	◐	○	TrustZone	✓	✗	✗	$\approx$ 10k
Wimpy [24]	Virt	MMIO	●	○	●	DRTM	✓	✗	✗	$\approx$ 3.5k
GSK [25]	Virt	GPU MMIO	○	●	○	TrustVisor	✓	✗	✗	$\approx$ 35k
SGXIO [26]	Virt	Driver	●	○	○	SGX	✓	✗	✗	n/a
BASTION-SGX [27]	Chip	Bluetooth HCI	●	○	●	SGX	✗	✓	✗	n/a
ProximiTEE [28]	Ext	USB	●	○	○	SGX	✗	✗	✓	$\approx$ 5k
SecDisplay [29]	Virt	USB	●	◐	○	TrustZone	✓	✗	✗	$\approx$ 2k
TIO [30]	Ext	USB	●	○	●	SGX	✗	✗	✓	$\approx$ 27k [2]
Aurora [31]	SMM Virt	MMIO	●	○	●	SGX	✗	✓	✗	$\approx$ 3.3k + 696KB [3]

Tabela 6.1: Comparison of Trusted Path Solutions.

[1] only binary size given [2] counts firmware, enclave framework & full asymmetric crypto lib [3] hypervisor (LoC) + enclave library (compiled binary)

/ altele); mecanismele de izolare I/O utilizate (bazate pe virtualizare - *Virt*, control al accesului bazat pe chipset - *Chip*, dispozitiv extern - *Ext*); interfață / protocol: USB / Bluetooth / Rețea; pentru abordările bazate pe virtualizare, metoda de implementare a acestora: MMIO (memory mapped I/O) / virtualizarea driverului de dispozitiv.

Rețineți că semicercul denotă o implementare parțială a caracteristicii; pentru HID, aceasta înseamnă că numai un subset de dispozitive de intrare este utilizabil; pentru afișare, aceasta înseamnă ieșire numai text. Sunt indicate componentele TCB suplimentare (inclusiv dispozitivul / modulul cip adăugat, în afară de platforma CPU).

O primă observație este că implementarea unui afișaj grafic de încredere este o problemă dificilă: multe soluții au funcționat doar pentru o ieșire parțială, în mod text, fie folosind un LCD extern, fie o ieșire securizată a consolei (folosind comutarea modului text de tip BIOS).

Menționăm că, pentru baza de calcul de încredere - o măsură de comparație de dorit, va-

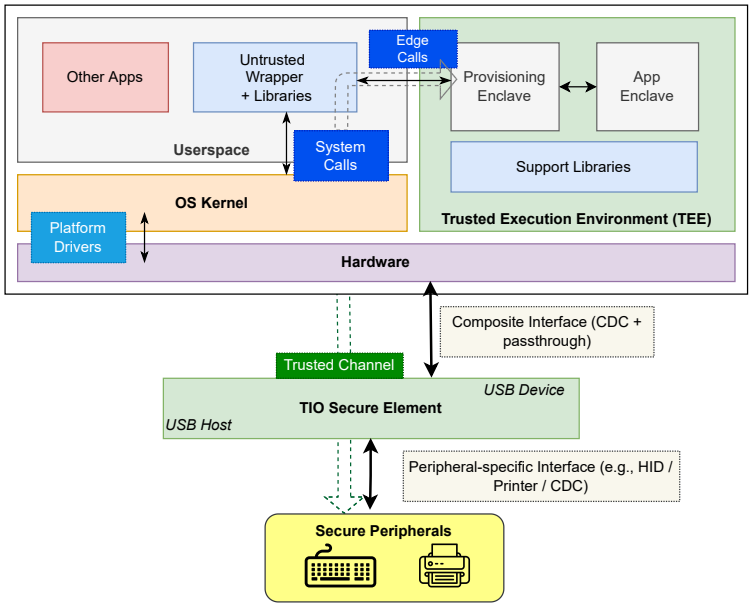


Figura 6.1: Arhitectura de sistem TIO.

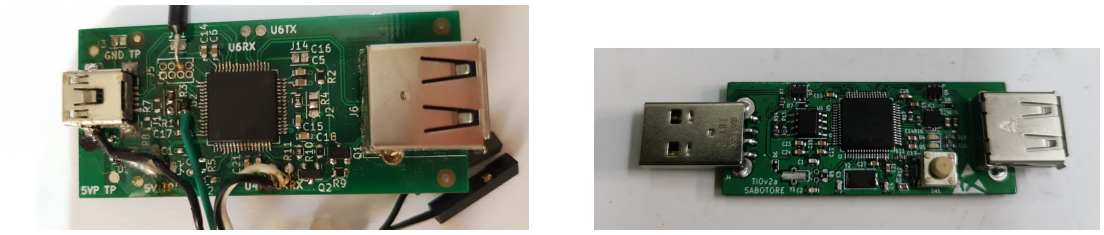
lorile dimensiunilor au fost preluate ca atare din lucrări și nu sunt fiabile în acest scop din cauza diferențelor dintre metodologiile de măsurare (de exemplu, linii de cod vs. dimensiuni binare, diferite dispozitive acceptate / seturi de caracteristici, platformele țintă care suportă costuri generale ale cadrului, biblioteci criptografice neoptimizate utilizate).

6.3 TIO - I/O securizat pentru Intel SGX

TIO constă într-un dongle hardware personalizat cu două porturi USB: o interfață gazdă pentru conectarea perifericelor generice și o interfață separată pentru dispozitive USB pentru interacțiunea cu PC-ul cu SGX. Astfel, elementul securizat TIO acționează ca o poartă de intrare/ieșire transparentă, stabilind un canal securizat între enclavă și un periferic, altfel inconștient, pentru schimbul de pachete USB. Arhitectura generală este ilustrată în Figura 6.1.

La stabilirea canalului securizat între o enclavă și modulul TIO, folosim un protocol de autentificare adecvat care protejează utilizatorul împotriva atacurilor de tip man-in-the-middle prin autentificarea reciprocă a părților implicate. Pentru aceasta, introducem o cerință inițială: dispozitivul de încredere trebuie să fie asociat în prealabil cu mașina utilizatorului (prin pornirea unui sistem de operare Linux readonly stocat în memoria flash a dispozitivului).

Prototipul TIO înglobează un microcontroler ieftin STM32F405 ARM Cortex-M4 care rulează la 168 MHz, cu 192KB SRAM și 1MB flash. Acesta include, de asemenea, 2 interfețe USB OTG, una utilizată pentru conectarea cu PC-ul, iar cealaltă configurată ca gazdă USB pentru perifericele de încredere. Dispozitivul asemănător unui dongle a fost împachetat ca



(a) v1 (cu fire atașate pentru depanare) (b) v2 (mai curat, conector Type A)

Figura 6.2: Prototipuri hardware TIO.

o placă de circuite imprimate cu factor mic, cu conectori și componente auxiliare, așa cum este prezentat în figura 6.2.

Tabela 6.2: TCB size measurements for the microcontroller firmware and the enclave, split components.

(a) Microcontroller Firmware

Module	LoC	ObjCode (KB)
mbedTLS	15018	44.1
STM32 SDK	8277	18.1
Firmware code	3215	11.2

(b) Device Provisioning Enclave

Module	LoC	ObjCode (KB)
IPP Crypto	22955	254
SGX tRTS	9742	111
DPE code	687	11.1

Rezultatele privind dimensiunea TCB sunt prezentate în tabelul 6.2a pentru *firmware-ul microcontrolerului* și în tabelul 6.2b pentru *enclavă*. Observăm că enclava are o amprentă totală mai mare în comparație cu codul încorporat, datorită cadrului Intel SGX de mari dimensiuni utilizat.

Tabela 6.3: Performance measurements.

Handshake Time	1165 ms
HID Report Latency Overhead (RTT)	1.3 ms

În tabelul 6.3, am măsurat timpul necesar pentru ca enclava să efectueze handshake-ul inițial Diffie-Hellman cu microcontrolerul și timpul de latență al rapoartelor de intrare de la un periferic HID (o tastatură USB).

Implementarea noastră înțelege în prezent protocoalele USB HID și Printer class. Soluția noastră poate fi extinsă pentru a acoperi și alte clase de periferice prin implementarea clasei de dispozitive corespunzătoare în interiorul enclavei (pentru anumite aplicații, dezvoltatorul trebuie să îmbunătățească, de asemenea, firmware-ul microcontrolerului pentru gestionarea/filtrarea stării I/O specifice clasei sau pentru a adăuga indicarea activității).

Capitolul 7

Infrastructură Industrială de Încredere

Orice utilizare a computerelor conectate ridică problema securității. Acest lucru este valabil și pentru sistemele industriale utilizate pentru automatizarea eficientă a proceselor dintr-o fabrică / uzină: deoarece astfel de sisteme sunt capabile să controleze utilaje fizice, cum ar fi motoarele electrice sau supapele, potențialul de daune în cazul unui atac cibernetic ar putea avea consecințe devastatoare. În acest capitol, explorăm utilizarea tehnologiilor de execuție de încredere în efortul de a îmbunătăți securitatea aplicațiilor sensibile prin separarea acestora de un sistem de operare bogat potențial vulnerabil, astfel încât, în cazul în care sunt exploatate, raza de acțiune a atacatorului va fi limitată.

Lucrarea noastră propune o arhitectură care utilizează TEE-uri și un dispozitiv firewall pentru a izola rețeaua de control industrial sensibilă din punct de vedere al securității de sistemele de uz general (de exemplu, PC-urile operatorilor, serverele la distanță) care pot fi expuse atacurilor cibernetice din surse externe (de exemplu, internet, stick-uri USB), oferind în același timp mijloacele necesare pentru emiterea în siguranță a comenzilor normale și excepționale (oprire de urgență) către controlerele de automatizare. În acest scop, îmbunătățim cel mai popular protocol de comunicare electronică industrială, Modbus, cu caracteristici de autentificare criptografică și proiectăm un dispozitiv firewall accesibil pentru a facilita tranziția echipamentelor vechi la o rețea securizată. Deoarece cerințele în timp real sunt obligatorii, demonstrăm că până și microcontrolerele cu putere redusă, disponibile în comerț, pot gestiona calculele criptografice necesare pentru un protocol securizat cu o performanță acceptabilă.

Arhitectura noastră, ilustrată în figura 7.1, introduce un canal de comunicare autentificat între echipamentul de control și aplicațiile de încredere care rulează pe dispozitive care nu sunt de încredere (în rețeaua corporativă/de management care este, probabil, conectată la internet). Am utilizat un firewall bazat pe microcontroler, cu costuri reduse și consum redus de energie, ca model pentru integrarea unei rețele de dispozitive vechi printr-o rețea serială, deși fiecare dintre controlere poate implementa protocolul Modbus autentificat separat, datorită cerințelor hardware reduse ale soluției noastre. Din acest motiv, noi abstractizăm în continuare dispozitivele industriale individuale și considerăm gateway-ul nostru ca fiind singurul sistem de control în cadrul protocolului nostru.

Modificăm Modbus într-un mod compatibil, adăugând un protocol de autentificare criptografică prin utilizarea unui cod de funcție personalizat și încapsulând tot restul ca date,

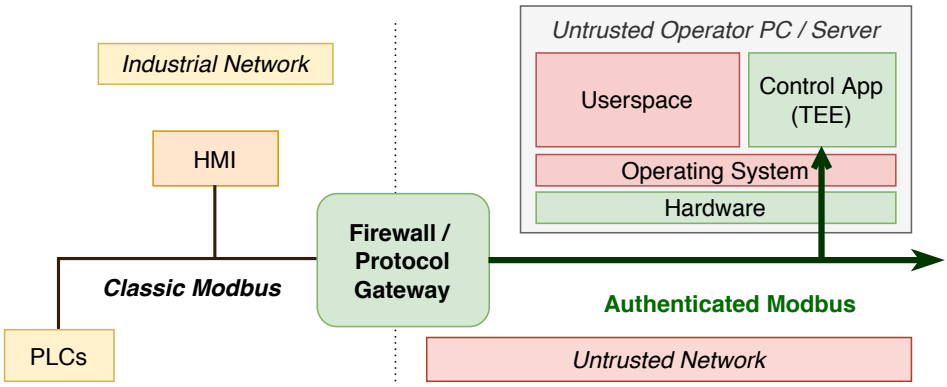


Figura 7.1: Arhitectură ICS de încredere.

astfel încât orice dispozitiv de traducere a protocolului aflat în cale (cum ar fi Modbus TCP la Modbus RTU) va continua să funcționeze conform așteptărilor.

Am implementat un prototip de firewall folosind un kit de dezvoltare a microcontrolerului Olimex STM32F4. MCU are o frecvență maximă de 168MHz, 192KB de memorie RAM și 1MB de stocare flash. Două dintre UART-urile (interfețe seriale) ale plăcii au fost conectate prin cablu la un Raspberry Pi și la un laptop care simulează dispozitivele de control industrial / master folosind Python.

Am folosit un analizor logic (Figura 7.2) pentru a determina performanța / timpul de latență introdus de protocolul nostru modificat (deoarece Modbus RTU are timpi stricți). Timpul total de execuție MCU al operațiilor Diffie Hellman autentificate este de $\approx 1100\ ms$, deși este împărțit între 2 cereri.

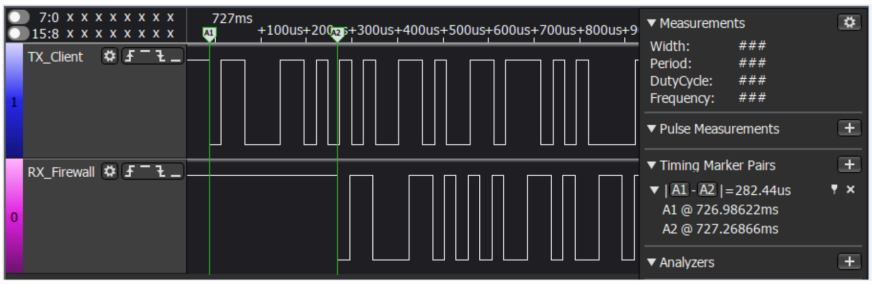


Figura 7.2: Captură analizor logic.

În condiții normale de funcționare, latența de calcul MAC simetric introdusă este de $280uS/bloc$ ($1\ block = 16\ bytes$). Pentru calcularea perioadei maxime de tăcere a Modbus: $t_{silence} = \frac{3,5*10}{baudrate}$, care, pentru un baudrate de 115200 se obține: 303 uS. Există, de asemenea, un overhead fix de dimensiune PDU de 33 $bytes$, care ar crește durata de transmisie (în special la viteze de transmisie mici), dar în rest nu va afecta niciuna dintre cerințele protocolului.

Capitolul 8

Concluzii

Teza noastră s-a axat pe mediile de execuție de încredere (Trusted Execution Environments - TEE), o tehnologie care oferă zone hardware izolate care pot fi utilizate pentru a proteja aplicațiile sensibile împotriva programelor malware mai privilegiate și prin reducerea complexității bazei de calcul de încredere (Trusted Computing Base - TCB), setul de componente critice de securitate dintr-un sistem.

Am explorat mai multe aspecte de cercetare specifice: fiabilitatea cloud, arhitectura software de încredere și provocările inginerești, fezabilitatea dispozitivelor încorporate în infrastructuri de încredere, asigurarea unei căi de intrare/ieșire de încredere pentru perifericele hardware și întărirea securității dispozitivelor industriale cu ajutorul TEE-urilor.

8.1 Rezumat al tezei

În Capitolul 2, am descris evoluția conceptelor informatice de încredere care au condus la apariția unor medii de execuție de încredere practice. Am prezentat diversele tehnologii de izolare disponibile în hardware de bază (atât pentru dispozitive încorporate / mobile: ARM TrustZone, cât și pentru cipurile de calcul general: Trusted Execution Technology de la Intel, succesorul său, Software Guard Extensions și Secure Encrypted Virtualization de la AMD), împreună cu publicațiile științifice aferente.

În Capitolul 3, începem prin a trece în revistă problema încrederii în cloud și abordările pentru protejarea datelor utilizatorilor pe serverele neîncrezătoare. Susținem că noile soluții sunt permise de noile tehnologii de încredere și de disponibilitatea lor actuală / viitoare în ofertele comerciale ale furnizorilor de top (Amazon AWS, Microsoft Azure, Google Cloud), o mișcare care poate crește gradul de încredere al serviciilor în cloud. Dezvoltăm *SecCollab*, abordarea noastră pentru asigurarea confidențialității în sistemele de editare a documentelor bazate pe cloud. Aceasta se prezintă sub forma unei extensii de browser și valorifică protocolul existent de sincronizare diferențială utilizat de aplicațiile colaborative, implementând un jurnal criptat deasupra acestuia pentru a păstra atât confidențialitatea documentelor utilizatorilor, cât și funcțiile de editare în timp real ale acestora.

În continuare, în Capitolul 4, am prezentat modul în care conceptul de execuție de încredere necesită schimbări în paradigmele de dezvoltare software pentru aplicațiile de încredere. Deoarece sistemul de operare este considerat acum ca fiind de neîncredere, programele care se bazează pe serviciile sale (de exemplu, sistemul de fișiere, rețelele, accesul I/O hardware) trebuie să-și protejeze secretele prin mijloace suplimentare (criptografie sau

alte servicii de încredere ale platformei). În secțiunea 4.2, am discutat mai multe lucrări de cercetare care vizează minimizarea efortului de dezvoltare: rularea programelor în interiorul TEE-urilor cu modificări minime sau deloc sau cadre pentru crearea de aplicații de încredere între platforme. Am descris, de asemenea, abordarea noastră, HiddenApp (4.3), pentru a permite executarea sigură a aplicațiilor Linux existente în interiorul unui TEE ARM TrustZone. Pentru aceasta, am dezvoltat un microkernel care poate rula programe nealterate în interiorul lumii securizate prin interceptarea apelurilor de sistem ale acestora și transmiterea lor către sistemul de operare bogat pentru procesare.

Capitolul 5 a arătat că microcontrolerele moderne pot rula cu succes cod criptografic cu performanțe rezonabile, depinzând în principal de cerințele de putere ale aplicației. Au fost evaluate mai multe biblioteci criptografice open-source pentru a fi rulate pe dispozitive de mici dimensiuni, unde spațiul de stocare și memoria sunt limitate, și s-a demonstrat că consumul de energie datorat procesării criptografice este neglijabil. Ulterior, am utilizat acest lucru ca element de bază pentru proiectarea de dispozitive încorporate pentru a spori securitatea aplicațiilor de încredere.

De la Capitolul 6, trecem la problema căii de intrare/ieșire de încredere: asigurarea unei comunicări sigure între mediile de execuție de încredere și perifericele hardware în fața unui sistem de operare de neîncredere. În secțiunea 6.2, am prezentat o sistematizare a soluțiilor de ultimă generație existente privind căile de încredere pentru diferite platforme TEE (Intel TxT, ARM TrustZone, Intel SGX) și dispozitive (clasificate pe tipuri: intrare HID/ieșire pe ecran; în funcție de interfață: I/O cu memorie, GPU, USB, Bluetooth etc.). În 6.3, am dezvoltat *TIO*, o abordare bazată pe hardware pentru stabilirea unei căi de intrare/ieșire securizate între dispozitivele USB generice și TEE-urile bazate pe Intel SGX. Dispozitivul nostru încorporat este o modalitate mică și practică de a spori fiabilitatea aplicațiilor care necesită o interacțiune sigură cu utilizatorul, pe care o demonstrăm prin protejarea intrărilor de la tastatură și prin imprimarea în siguranță a documentelor PDF.

În cele din urmă, în Capitolul 7, am descris o arhitectură pentru securizarea sistemelor de control industrial. Soluția noastră se bazează pe un dispozitiv firewall cu costuri reduse, situat la granița dintre rețeaua de control sensibilă și rețelele de management/corporative care nu prezintă încredere sau chiar în spatele fiecărei unități ciber-fizice. O aplicație de încredere (care rezidă în interiorul unei enclave pe PC-ul operatorului) stabilește un canal de încredere de intrare/ieșire (I/O) cu firewall-ul încorporat și semnează fiecare solicitare cu o cheie de autentificare partajată, pe care dispozitivul o poate folosi apoi pentru a filtra orice comenzi malițioase, împiedicând astfel sabotajul chiar și din partea atacatorilor privilegiați. De asemenea, am modernizat un protocol industrial popular (Modbus) cu câmpuri de integritate criptografică pentru a asigura autenticitatea pachetelor trimise în cadrul rețelelor tradiționale.

8.2 Contribuții

În teza noastră, am adus câteva contribuții originale pentru a rezolva unele dintre problemele actuale care ar putea îngreuna adoptarea tehnologiilor de execuție de încredere: dezvoltarea de aplicații de încredere și calea de încredere I/O. De asemenea, propunem soluții pentru a îmbunătăți securitatea în toate domeniile de aplicații populare: cloud, calcul personal, încorporat și industrial.

1. Am prezentat un context cuprinzător privind istoricul tehnologiilor de încredere, ale TEE-urilor comerciale disponibile până în prezent, precum și o trecere în revistă a numeroase lucrări de ultimă oră legate de acest domeniu.

2. Am proiectat SecCollab [32], o metodă de asigurare a confidențialității aplicațiilor de editare colaborativă a documentelor online, bazate pe web, prin utilizarea unei extensii de browser pentru a adăuga criptarea pe partea clientului la protocoalele de sincronizare diferențială.

3. Am prezentat provocările legate de dezvoltarea aplicațiilor care vizează mediile izolate și am implementat HiddenApp [16], o soluție pentru rularea aplicațiilor Linux nemodificate în interiorul lumii securizate a ARM TrustZone.

4. Am testat performanța criptografică a mai multor dispozitive încorporate și biblioteci software [33, 17] cu scopul de a le utiliza ca dispozitive de încredere.

5. Am abordat problema securizării interacțiunii TEE-urilor cu perifericele de intrare/ieșire din sistemele de operare de neîncredere, propunând TIO [30], o soluție hardware cu costuri reduse pentru enclavele Intel SGX pentru a comunica în siguranță cu alte dispozitive USB (de exemplu, tastatură, imprimante). Am sistematizat, de asemenea, celelalte soluții Trusted I/O Path, comparând utilizabilitatea și dimensiunile TCB ale acestora.

6. Am propus o arhitectură pentru îmbunătățirea securității sistemelor ciber-fizice (industriale) [34] cu ajutorul mediilor de execuție de încredere și al dispozitivelor integrate personalizate, cu costuri reduse, îmbunătățind protocolul Modbus, în mod tradițional nesigur, pentru a face autentificarea criptografică a comenzilor de control sensibile sau a datelor senzorilor.

8.3 Dezvoltări ulterioare

În cele din urmă, prezentăm direcțiile viitoare pe care am dori să le urmărim sau să le vedem realizate în domeniul tehnologiilor de încredere. În primul rând, susținem că securitatea cibernetică a calculatoarelor de uz general ar fi foarte benefică dacă furnizorii ar colabora cu o platformă mai deschisă pentru execuția de încredere, astfel încât dezvoltatorii să se poată baza pe un set standardizat de caracteristici (de exemplu, API-uri de apelare la limită, atestare la distanță, I/O de încredere pentru unele periferice de bază), ceea ce ar spori adoptarea soluțiilor de execuție de încredere.

De asemenea, intenționăm să dezvoltăm în continuare dispozitivul nostru USB TIO încorporat pentru a îmbunătăți performanța, pentru a oferi suport pentru alte clase de periferice (de exemplu, hub USB) și pentru a-l integra în aplicațiile de birou care necesită o protecție sporită a secretelor (administratori de parole, stocare de certificate, ssh etc.). Am dori, de asemenea, să explorăm posibilitatea de a construi un mediu de execuție fiabil portabil ca dispozitiv de securitate portabil (utilizând o unitate centrală de procesare a aplicațiilor încorporată, cum ar fi cea a Raspberry Pi).

8.4 Lista publicațiilor

1. Dumitru C. Trancă, **Florin Stancu**, Răzvan Rughiniș and Daniel Rosner, *"SiloSense: ZigBee-based wireless measurement system architecture for agriculture parameter monitoring"*, 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), Barcelona, pp. 0330-0335, 2017 (IEEE), DOI: 10.1109/CoDIT.2017.8102613, WOS: 000450826500057.

2. **Florin Stancu**, Mihai Chiroiu and Răzvan Rughiniș, *"SecCollab - Improving Confidentiality for Existing Cloud-Based Collaborative Editors"*, 2017 21st International Conference on Control Systems and Computer Science (CSCS), Bucharest, pp. 324-331, 2017 (IEEE), DOI: 10.1109/CSCS.2017.51, WOS: 000449004400044.

3. Veronica Velciu, **Florin Stancu** and Mihai Chiroiu, *"HiddenApp - Securing Linux Applications Using ARM TrustZone"*, Innovative Security Solutions for Information Technology and Communications (SECITC), 2018 Lecture Notes in Computer Science, vol 11359, 2018 (Springer, Cham), DOI: 10.1007/978-3-030-12942-2_5.

4. **Florin Stancu**, Dumitru C. Trancă, Mihai Chiroiu and Răzvan Rughiniş, *"Evaluation of cryptographic primitives on modern microcontroller platforms"*, 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet), Cluj-Napoca, pp. 1-6, 2018 (IEEE), DOI: 10.1109/ROEDUNET.2018.8514127, WOS: 000517570500005.

5. Daniel Rosner, Cristiana Trifu, Dumitru C. Trancă, Iuliu Vasilescu and **Florin Stancu**, *"Magnetic Field Sensor for UAV Power Line Acquisition and Tracking"*, 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet), Cluj-Napoca, pp. 1-5, 2018 (IEEE), DOI: 10.1109/ROEDUNET.2018.8514123, WOS: 000517570500002.

6. Răzvan Tataroiu, **Florin Stancu** and Dumitru C. Trancă, *"Energy Considerations Regarding Transport Layer Security in Wireless IoT Devices"*, 2019 22nd International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, pp. 337-341, 2019 (IEEE), DOI: 10.1109/CSCS.2019.00060, WOS: 000491270300053.

7. **Florin Stancu**, Dumitru C. Trancă and Mihai Chiroiu, *"TIO - Secure Input/Output for Intel SGX Enclaves"*, 2019 International Workshop on Secure Internet of Things (SIOT), 2019 (IEEE), DOI: 10.1109/SIOT48044.2019.9637105.

8. **Florin Stancu**, Răzvan Rughiniș, Dumitru C. Trancă and Ioana Popescu, *"Trusted Industrial Modbus Firewall for Critical Infrastructure Systems"*, 2020 RoEduNet (19th RoEduNet Conference: Networking in Education and Research), 2020 (IEEE), DOI: 10.1109/RoEduNet51892.2020.9324884, WOS: 000654265900033.

9. **Florin Stancu**, Alexandru Mircea, Răzvan Rughiniș, Mihai Chiroiu, *"Systematization of Trusted I/O solutions for Isolated Eecution Environments"*, accepted for publication at U.P.B. Scientific Bulletin, Series C, Bucharest, Romania, 2022 (Journal).

Bibliografie

[1] M. Larabel, Phoronix, "The Linux Kernel Enters 2020 At 27.8 Million Lines In Git," https://www.phoronix.com/scan.php?page=news_item&px=Linux-Git-Stats-EOY2019, January 2020.

[2] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The untapped potential of trusted execution environments on mobile devices," *IEEE Security & Privacy*, vol. 12, no. 4, pp. 29–37, 2014.

[3] ARM Holdings, "ARM TrustZone Security Extensions," https://developer.arm.com/technologies/trustzone.

[4] Intel, "Intel SGX Software Guard Extensions," https://software.intel.com/en-us/sgx.

[5] D. Kaplan, J. Powell, and T. Woller, "AMD Memory Encryption," *White paper*, 2016.

[6] J. Criswell, N. Dautenhahn, and V. Adve, "Virtual ghost: Protecting applications from hostile operating systems," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 81–96, 2014.

[7] U. Lee and C. Park, "SofTEE: Software-based trusted execution environment for user applications," *IEEE Access*, vol. 8, pp. 121 874–121 888, 2020.

[8] W. Futral and J. Greene, "Fundamental principles of intel® txt," in *Intel® Trusted Execution Technology for Server Platforms*. Springer, 2013, pp. 15–36.

[9] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, pp. 1–10, 2007.

[10] F. Y. Rashid, "The rise of confidential computing: Big tech companies are adopting a new security model to protect data while it's in use-[news]," *IEEE Spectrum*, vol. 57, no. 6, pp. 8–9, 2020.

[11] J. Jang, C. Choi, J. Lee, N. Kwak, S. Lee, Y. Choi, and B. B. Kang, "Privatezone: Providing a private execution environment using arm trustzone," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 797–810, 2018.

[12] L. Guan, P. Liu, X. Xing, X. Ge, S. Zhang, M. Yu, and T. Jaeger, "TrustShadow: Secure execution of unmodified applications with ARM trustzone," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2017, pp. 488–501.

[13] A. Baumann, M. Peinado, and G. Hunt, "Shielding applications from an untrusted cloud with Haven," *ACM Transactions on Computer Systems (TOCS)*, vol. 33, no. 3, pp. 1–26, 2015.

[14] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'keeffe, M. L. Stillwell *et al.*, "{SCONE}: Secure linux containers with intel {SGX}," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 689–703.

[15] C.-C. Tsai, D. E. Porter, and M. Vij, "{Graphene-SGX}: A practical library {OS} for unmodified applications on {SGX}," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 645–658.

[16] V. Velciu, F. Stancu, and M. Chiroiu, "Hiddenapp – Securing linux applications using ARM TrustZone," in *International Conference on Security for Information Technology and Communications*. Springer, Cham, 2018, pp. 41–52.

[17] R. Tataroiu, F. A. Stancu, and D.-C. Tranca, "Energy considerations regarding Transport Layer Security in wireless IOT devices," in *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2019, pp. 337–341.

[18] Z. Zhou, V. D. Gligor, J. Newsome, and J. M. McCune, "Building verifiable trusted path on commodity x86 computers," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 616–630.

[19] T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch, "The Zurich Trusted Information Channel–an efficient defence against man-in-the-middle and malicious software attacks," in *International Conference on Trusted Computing*. Springer, 2008, pp. 75–91.

[20] J. M. McCune, "Safe passage for passwords and other sensitive data," in *Proceedings of the Network and Distributed System Security Symposium, 2009*, 2009.

[21] A. Filyanov, J. M. McCuney, A.-R. Sadeghiz, and M. Winandy, "Uni-directional trusted path: Transaction confirmation on just one device," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2011, pp. 1–12.

[22] X. Ruan, *Platform Embedded Security Technology Revealed*. Springer Nature, 2014.

[23] W. Li, M. Ma, J. Han, Y. Xia, B. Zang, C.-K. Chu, and T. Li, "Building trusted path on untrusted device drivers for mobile devices," in *Proceedings of 5th Asia-Pacific Workshop on Systems*, 2014, pp. 1–7.

[24] Z. Zhou, M. Yu, and V. D. Gligor, "Dancing with giants: Wimpy kernels for on-demand isolated i/o," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 308–323.

[25] M. Yu, V. D. Gligor, and Z. Zhou, "Trusted display on untrusted commodity platforms," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 989–1003.

[26] S. Weiser and M. Werner, "SGXIO: Generic trusted I/O path for Intel SGX," in *Proceedings of the seventh ACM on conference on data and application security and privacy*, 2017, pp. 261–268.

[27] T. Peters, R. Lal, S. Varadarajan, P. Pappachan, and D. Kotz, "BASTION-SGX: Bluetooth and architectural support for trusted I/O on SGX," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, 2018, pp. 1–9.

[28] A. Dhar, I. Puddu, K. Kostiainen, and S. Capkun, "ProximiTEE: Hardened SGX attestation by proximity verification," in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 2020, pp. 5–16.

[29] J. Cui, Y. Zhang, Z. Cai, A. Liu, and Y. Li, "Securing display path for security-sensitive applications on mobile devices," *Computers, Materials and Continua*, vol. 55, no. 1, p. 17, 2018.

[30] D. C. T. F. A. Stancu and M. Chiroiu, "TIO – Secure Input/Output for Intel SGX Enclaves," in *International Workshop on Secure Internet of Things (SIOT)*, 2019.

[31] H. Liang, M. Li, Y. Chen, L. Jiang, Z. Xie, and T. Yang, "Establishing trusted I/O paths for SGX client systems with Aurora," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1589–1600, 2019.

[32] F. A. Stancu, M. Chiroiu, and R. Rughinis, "SecCollab – Improving Confidentiality for Existing Cloud-Based Collaborative Editors," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2017, pp. 324–331.

[33] F. A. Stancu, C. D. Trancă, M. D. Chiroiu, and R. Rughiniş, "Evaluation of cryptographic primitives on modern microcontroller platforms," in *2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2018, pp. 1–6.

[34] F. A. Stancu, R. V. Rughinis, C. D. Tranca, and I. L. Popescu, "Trusted Industrial Modbus Firewall for Critical Infrastructure Systems," in *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2020, pp. 1–5.