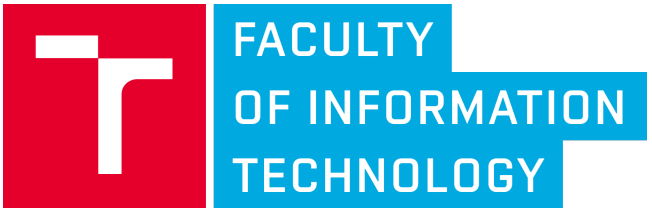


Bezpečná infrastruktura pro provoz senzorů

Technická dokumentace

Michal Kula

Vysoké učení technické v Brně
Fakulta informačních Technologií
Brno 2022

T A
Č R

Program Centra kompetence

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

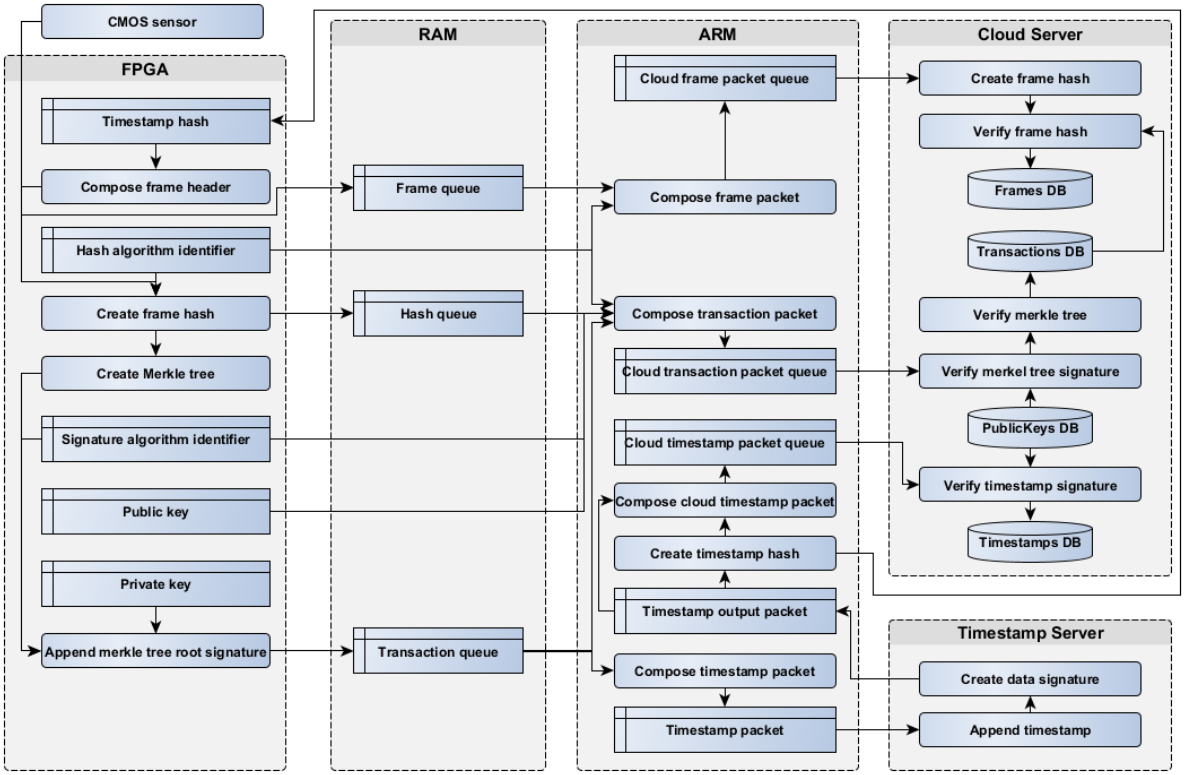
Obsah:

1. Úvod

Tato technická dokumentace popisuje systém bezpečné infrastruktury pro provoz senzorů vyvinutý v rámci projektu Secusen. Jako demonstrační a ověřovací aplikace bylo zvoleno využití kamerových senzorů. Primárním účelem systému je zajištění verifikovatelných časových razítek pro FPGA z výstupu V001, sběr snímků a metadat potřebných k autentizaci snímků pořízených tímto senzorem a tvorba systému pro ukládání a verifikaci snímků.

2. Struktura systému

Systém je rozdělen na FPGA zajišťující sběr dat z CMOS senzoru z výstupu V001, ARM systém zajišťující export dat z FPGA na síťový cloud, Cloud server umožňující sběr dat z několika senzorů s verifikací jejich autenticity a Časový server pro zajištění verifikovatelného času pro FPGA. Data-flow diagram systému je možné nalézt na obrázku 1.



Obrázek 1: Data flow diagram systému

2.1. FPGA

FPGA část systému je vyvinutá v rámci tohoto projektu jako funkční vzorek (V001). V rámci výstupu V002 byl pro účely simulace systémů s více kamerami vyvinut softwarový emulátor FPGA obsahující emulaci komponent původně implementovaných v rámci funkčního vzorku.

FPGA publikuje svoje nastavení a identitu ARM systému prostřednictvím registrů:

- 1B Hash algorithm identifier - identifikátoru hashovacího algoritmu používaného k zjištění případné manipulace s daty snímku
- 1B Signature algorithm identifier - identifikátoru algoritmu pro elektronický podpis dat k zajištění autenticity dat
- Public key - veřejný klíč pro zajištění identifikace senzoru a ověření autenticity dat, velikost klíče je dána použitým algoritmem

Sestavení hlavičky snímku

Ke každému snímku je připojena hlavička snímku o velikosti jednoho řádku snímku a obsahuje:

- 4B výšku snímku - výška snímku z CMOS senzoru
- 4B šířku snímku - šířka snímku z CMOS senzoru
- 4B id transakce - sekvenční číslo transakce sloužící pro mapování na Merkleovy stromy
- 4B id snímku v rámci transakce - sekvenční číslo snímku v rámci jedné transakce
- 1B id hashovacího algoritmu pro časové razítko - velikost časového razítka v bajtech
- 3B zarovnání
- hash časového razítka - hash časového razítka pro zjištění časového rozsahu, kdy byl snímek pořízen

Tato hlavička je připojena k snímku a přidána do fronty Frame queue.

Hashování snímku

Hashování snímku slouží k ověření jeho integrity. V rámci snímku jsou jednotlivé řádky včetně vygenerované hlavičky zahashovány prostřednictvím hashovacího algoritmu s identifikátorem hashovacího algoritmu. Na výsledné hashe je aplikováno zahashování stejným algoritmem. Výsledný hash je spolu s hlavičkou obsahující:

- 4B id transakce
- 4B id snímku v rámci transakce

přidán do fronty Hash queue. V současnosti je v rámci funkčního vzorku V001 podporován pouze hashovací algoritmus SHA-256.

Tvorba Merkleova stromu

Použití Merkleova stromu rozšiřuje zajištění integrity dat oproti předchozímu hashování na více snímků. Hashe jednotlivých snímků jsou nastaveny jako listy binárního stromu. Výstup každého uzlu stromu je hashem konkatenace výstupů jeho dětí. Díky použití Merkleova stromu je možné pro zajištění autenticity dat podepisovat pouze kombinace kořenu stromu a počtu jeho listů, čímž jsou sníženy nároky na rychlost algoritmu zajišťujícího elektronické podpisy. Další výhodou vyplývající z provázanosti snímků v rámci stromu je možnost regulovat četnost zajišťování časových razítek dle velikosti transakce (počtu listů v Merkleově stromu).

Uzly v Merkleově stromu se aktualizují s každým přidaným hashem snímku a není nutné si tyto hashe dále ukládat. Maximální množství uzlů, které je třeba mít uloženo v paměti při užití tohoto stromu, je $log_2 N$ kde N je celkový počet hashů.

Výpočet podpisu kořenu Merkleova stromu

Kořenový hash Merkleova stromu a celkový počet hashů transakce se v předem určených intervalech podepisuje prostřednictvím privátního klíče uloženého v FPGA algoritmem elektronického podpisu. Identifikátor tohoto algoritmu je z FPGA možné vyčíst prostřednictvím registru Signature algorithm identifier. Výstup obsahující:

- 4B id transakce
- 4B počet snímků (listů) v transakci
- kořenový hash transakce
- podpis transakce

je přidán do fronty TransactionData. V současné době je na fuknčním vzorku (výsledek V001) podporován pouze algoritmus elektronického podpisu EcDSA s eliptickou křivkou ed25519.

Výpočet kořenu Merkleova stromu a jeho podpis se provádí ve 2 intervalech, které jsou dané konfigurací FPGA:

- ověřovací interval - Výpočet podpisu v ověřovacím intervalu slouží pouze k okamžitému ověření autenticity nových snímků na straně Cloud serveru. Toho je docíleno tak, že neověřené snímky ARM drží v paměti do té doby, než je odeslán podepsaný kořenový hash, který tyto snímky zaštiťuje. Pro rozlišení ověřovacího intervalu je nastaven nejvyšší bit v počtu snímků na hodnotu 1.
- transakční interval - Kořenový hash podepsaný v transakčním intervalu slouží k dlouhodobé úschově a autentizaci celých transakcí na straně Cloud serveru. U takto podepsaného hashe je následně v režii ARMu zajištěn verifikovatelný čas z Časového serveru. Po přidání podepsaného hashe do fronty je uzavřena transakce a dochází k reinicializování stavu Merkleova stromu a identifikátoru snímku v transakci a je inkrementován identifikátor transakce.

2.2. ARM systém

Běh ARM systému je rozdělen do 3 částí, běžících v samostatných vláknech. Hlavní vlákno zajišťuje sběr a parsování dat z FPGA, kompletace transakcí a tvorbu síťových zpráv, které zařazuje do front pro odeslání. Zbylá 2 vlákna přebírají síťové zprávy vytvořené hlavním vláknem z front a zajišťují jejich odesílání a komunikaci s Cloud serverem a Timestamp serverem.

Autentifikace snímku

Autentifikovat snímek v navrženém systému využívajícím podepsané kořeny Merkleových stromů je možné pouze v případě, že dokážeme verifikovat, že hash patří do daného stromu. To lze provést rekonstrukcí stromu z hashů ostatních snímků z Merkleova stromu. Díky této vlastnosti je nutné zajistit spolu s přenosem kořenu Merkleova stromu na Cloud server i kompletní sadu hashů, z kterých byl vypočten. Tyto hashe ale při rozsáhlých transakcích již není možné uchovávat na malých FPGA pro zaslání spolu s ostatními daty transakce. Řešením by byl jejich přenos ve stejné frontě spolu se

snímky, nicméně počet snímků v paměti RAM je omezený a pokud by ARM nestihl odebrat jakýkoliv snímek před jeho přepsáním, nebylo by možné ověřit autenticitu jakéhokoliv jiného snímku z této transakce. Z těchto důvodů jsou mezi FPGA a ARMem 3 separátní fronty ohodnocené dle priority:

1. Transaction queue - fronta obsahující kořeny Merkleových stromů a jejich podpisy
2. Hash queue - fronta hashů jednotlivých snímků
3. Frame queue - fronta samotných snímků s jejich hlavičkami

Dále jsou v ARMu 3 fronty mezi hlavním vláknem a vláknem pro komunikaci s Cloud serverem:

1. Cloud timestamp packet queue - fronta síťových zpráv s časovými razítky
2. Cloud transaction packet queue - fronta síťových zpráv s kořenovým hashem, podpisem a všemi hashi v transakci
3. Cloud frame packet queue - fronta síťových zpráv se snímky

V obou případech jsou nejprve čtena data z front s vyšší prioritou.

Prevence zahlcení Cloud serveru

Pro prevenci zahlcení Cloud serveru ARM odesílá nejprve transakci a poté až snímky dané transakce. Z tohoto důvodu je ihned po obdržení snímku Cloud serverem možné spočítat jeho hash a porovnat ho s databází již ověřených validních transakcí. Zároveň ale není možné si v ARMu uchovávat v RAM paměti všechny snímky rozsáhlých dokončených transakcí. Z tohoto důvodu je v ARMu nastaven ověřovací interval, v kterém jsou pravidelně generovány a podepisovány kořeny aktuálně rozpočítaného Merkleova stromu. Ověřovací interval je nastaven jako kompromis mezi velikostí použité paměti potřebné na uložení nepodepsaných snímků a rychlosti tvorby podpisů na FPGA. Typický počet snímků na podpis je 4, 8 nebo 16 snímků.

2.3. Časový server

Komunikace s časovým serverem probíhá prostřednictvím POST metody nad HTTPS protokolem. Pro testovací účely byl vytvořen vnořený binární protokol pro zajištění podepsaných časových razítek. Tento protokol je zjednodušenou alternativou k TSP protokolu z RFC3161 používanému pro verifikaci existence hashe dat před datem podpisu získaným prostřednictvím časových autorit (např. PostSignum). Vzhledem k obdobné funkcionalitě navrženého protokolu může být vytvořený protokol v budoucnu tímto protokolem nahrazen.

2.4. Cloud server

Cloud server slouží k dlouhodobému uchovávání snímků, transakcí a časových razítek těchto transakcí. Komunikace s Cloud serverem probíhá prostřednictvím navrženého binárního protokolu zapouzdřeného do HTTPS protokolu. Z prvních 4 bajtů hlavičky zprávy v tomto zapouzdřeném binárním protokolu je možné určit velikost této zprávy, a tudíž je možné tento binární protokol použít i bez zapouzdření prostřednictvím socketů. ALternativně je možné posílat snímky z kamery na cloud server pomocí RTP protokolu nad UDP.

3. Zpracování dat

V navrženém systému se vyskytují 3 základní třídy dat. Časová razítka sloužící pro ověření času pořízení snímku, pořízené snímky spolu s jejich metadaty a transakce obsahující metadata pro ověření integrity, autenticity a ověření pořízeného času.

3.1. Zpracování časových razítek

Primárním účelem časových razítek v systému je ověření času pořízení snímku. Časová razítka jsou vydávána časovou autoritou, která poskytuje verifikovatelný čas podepsaný spolu se vstupním hashem dat.

Tvorba síťové zprávy pro Timestamp serverem

Síťové zprávy pro časový server jsou vytvořeny pouze z transakcí z fronty Transaction queue, které byly vytvořeny v transakčním intervalu. Z každé transakce je odstraněno id transakce a ze zbytku dat je vytvořen hash pomocí hashovacího algoritmu dle identifikátoru hash algoritmu z FPGA. Poté je vytvořena síťová zpráva pro časový server v následujícím tvaru:

- 1B identifikátor hashovacího algoritmu
- 3B zarovnání
- hash transakce

a přepisuje předchozí zprávu v paměti Timestamp packet. Následně je hlavní vlákno uvolněno pro další zpracování dat z FPGA front.

Odeslání zprávy

Z paměti Timestamp packet se periodicky pokouší číst novou hodnotu samostatné vlákno pro komunikace s Timestamp serverem. Po získání síťové zprávy se vlákno opakovaně pokouší odeslat připravenou zprávu na časový server.

Zpracování zprávy Timestamp serverem

Časový server k hashi transakce z požadavku připojí časové razítko a to následně podepíše prostřednictvím algoritmu daného identifikátorem algoritmů podpisu z jeho konfigurace. Následně odpovídá na požadavek pomocí odpovědi v binárním formátu application/octet-stream s následující strukturou:

- 1B id hashovacího algoritmu
- 1B id algoritmu podpisu
- 2B zarovnání
- hash transakce
- 15B časové razítko - UTC čas dle RFC 2459 Section 4.1.2.5.2
- 1B zarovnání
- podpis
- veřejný klíč - slouží k autentizaci a identifikaci časového serveru

Zpracování časového razítka

Přijatá odpověď od časového serveru přepisuje aktuální hodnotu z paměti Timestamp output packet, kde si ji přebírá hlavní vlákno systému ARM pro další zpracování. Po obdržení nové zprávy z paměti

Timestamp output packet je přijata odpověď zahashovaná prostřednictvím identifikátoru hashovacího algoritmu z FPGA a následně je tento hash zapsán do registru Timestamp hash v FPGA.

Tvorba síťové zprávy s časovým razítkem pro Cloud server

Pro tvorbu síťové zprávy pro Cloud server je, stejně jako u zpracování časového razítka, použit vypočtený hash z odpovědi z časového serveru. Vytvořenou zprávu ve formátu:

- 1B typ zprávy - u zprávy s časovým razítkem je nastavený na hodnotu 1
- 1B id hashovacího algoritmu - identifikátor hashovacího algoritmu z FPGA registru
- 2B velikost odpovědi od časového serveru
- odpověď od časového serveru
- hash odpovědi od časového serveru

je přidán do fronty Cloud timestamp packet queue pro další zpracování.

Odeslání datové zprávy s časovým razítkem na Cloud server

Z fronty Cloud timestamp packet queue se s nejvyšší prioritou periodicky pokouší číst data samostatné vlákno pro komunikace s Cloud serverem. Po obdržení síťové zprávy z fronty se vlákno opakovaně pokouší odeslat zprávu na Cloud server.

Zpracování síťové zprávy Cloud serverem

Cloud server po přijetí požadavku s časovým razítkem nejprve verifikuje autenticitu časové autority, co vydala časové razítko dle veřejných klíčů uložených v databázi, a poté úspěšně verifikovaná razítka zapisuje do databáze.

3.2. Zpracování snímků

Snímky jsou v systému pořízeny prostřednictvím CMOS senzoru a v rámci FPGA je k nim přidávána hlavička s metadaty obsahujícími časové razítko, velikosti a identifikaci v rámci transakčního systému. Takto rozšířené snímky jsou umístěny do fronty Frame queue.

Tvorba síťové zprávy

Z fronty Frame queue se odebírají pouze snímky z transakcí, které již jsou zařazeny do fronty Cloud transaction packet queue. Při načtení snímků z fronty Frame queue je nejprve přidána hlavička k snímku v následujícím formátu:

- 1B typ zprávy - u zprávy se snímkem je nastavený na hodnotu 2
- 1B id hashovacího algoritmu - identifikátor hashovacího algoritmu z FPGA registru
- 2B zarovnání
- obsah snímku z fronty
- hash snímku

Výsledná síťová zpráva je poté přidána do fronty Cloud frame packet queue pro zpracování vláknem pro komunikaci s Cloud serverem.

Odeslání zprávy

Z fronty Cloud frame packet queue se s nejnižší prioritou periodicky pokouší číst data samostatné vlákno pro komunikace s Cloud serverem. Po obdržení zprávy z fronty se tuto zprávu samostatné vlákno opakovaně pokouší odeslat na Cloud server.

Zpracování zprávy Cloud serverem

Ze zprávy se snímkem je volitelně dle konfigurace serveru buď vypočten hash dle id hashovacího algoritmu z hlavičky zprávy, nebo je hash přečten z konce snímku (bez verifikace integrity). Pokud je tento hash nalezen v databázi hashů z ověřených transakcí, je snímek považován za autentický a je zapsán do databáze snímků.

3.3. Zpracování transakcí

Transakce zasílané na Cloud server slouží k zajištění autenticity, integrity a verifikace času pořízení snímků. Pro kompletní ověření se transakce skládá z kořenového hashe Merkleova stromu a jeho podpisu umístěném ve frontě Transaction queue a hashů všech snímků této transakce umístěných ve frontě Hash queue.

Kompletace dat

Pro možnost ověření autenticity dat na straně Cloudu je nutné připojit všechny hashe z fronty Hash queue ke kořenovému stromu a jeho podpisu z fronty Transaction queue. Hlavní vlákno ARMu se proto periodicky pokouší číst data z těchto front a zapisovat je do interních struktur. Transakce se navíc interně dělí na podtransakce, kde všechny podtransakce kromě poslední odpovídají transakcím vytvořeným v ověřovacím intervalu a poslední z nich odpovídá transakci vytvořené v transakčním intervalu. Podtransakce se odesílají na server v sekvenčním pořadí, protože hashe každé podtransakce obsahují i hashe všech předchozích. Pokud je jakákoliv z podtransakcí jakékoliv transakce dokončena (všechny hashe jsou již načtené), je vytvořena síťová zpráva pro komunikaci s Cloud serverem v následujícím formátu:

- 1B typ zprávy - u zprávy s transakcí je nastavený na hodnotu 0
- 1B id hashovacího algoritmu - identifikátor hashovacího algoritmu z FPGA registru
- 1B id algoritmu podpisu - identifikátor podpisového algoritmu z FPGA registru
- 4B počet snímků (listů) v transakci
- kořenový hash transakce
- veřejný klíč z FPGA registru
- hashe jednotlivých snímků.

Tato zpráva je následně přidána do fronty Cloud transaction packet queue.

Odeslání zprávy

Z fronty Cloud transaction packet queue se střední prioritou se periodicky pokouší číst data samostatné vlákno pro komunikace s Cloud serverem. Po obdržení síťové zprávy z fronty se tuto zprávu samostatné vlákno opakovaně pokouší odeslat na Cloud server.

Zpracování zprávy Cloud serverem

U obdržené zprávy je zjištěna identita FPGA dle veřejného klíče ze zprávy dle databáze veřejných klíčů, poté je ověřena autenticita kořenu Merkleova stromu prostřednictvím veřejného klíče a na

závěr je zrekonstruován kořen Merkleova stromu pro zajištění validity hashů. Pokud data transakce projdou všemi ověřeními, jsou všechny hashe z listů Merkleova stromu a kořenový hash s podpisem zapsány do databáze transakcí.

4. Ověření funkcionality

Pro účely ověření funkcionality byl sestaven systém skládající se z dvou kamer, cloudové části a časového serveru (simulovaného, kompatibilního s PostSignum). Kamera využívala zapečený senzor V001 v FPGA části a službou starající se o odesílání snímku na cloud a výměnu časových razítek v ARM části. Obrazová data se z kamer přenášela nezabezpečeně jako v současném systému, podepsané hashe a časové značky a další kryptograficky citlivé informace byly přenášeny přes standardní HTTPS s vlastními certifikáty. Experimentálně bylo ověřeno očekávané chování systému. Byly provedeny testy s různou periodou podpisu vůči časové autoritě (5 sec, 1 min, 1 hod) a různé velikosti datových rámců pro potřeby nejen kamerových aplikací. Bylo také experimentálně otestováno, že jsme schopni ověřit autenticitu a integritu dat ze senzorů i při ztrátě jiných dat v jedné periodě časového serveru.

Pro ověření škálovatelnosti a chování s více senzory byl vytvořen simulátor FPGA časti v jazyce Python. Tento simulátor generuje prázdné snímky, přidává korektní hlavičky obrázků, počítá hash SHA256, vyhodnocuje Merkleův strom a podepisuje kořen tohoto stromu stejně jako FPGA. Díky tomu je možné simulovat chování systému s desítkami kamer bez nutnosti vlastnit fyzický HW. Je tak možné například ověřit limity služby pro ověřování a ukládání dat. Byl otestován systém dvaceti kamer produkují více než 1GB dat za sekundu v 1000 datových rámcích. V simulovaném testu jsme byly schopni ověřit validitu jednotlivých podpisů i na normálním kancelářském počítači. V reálném nasazení by jsme však byly limitování propustností využitých síťových rozhraní.

5. Závěr

Cílem tohoto software bylo vytvořit bezpečnou infrastrukturu umožňující provoz moderních senzorů se zajištěním autenticity, integrity a časového ukotvení pořízených dat. Infrastruktura byla implementována a následně ověřeny její parametry na úloze zpracování dat z kamerových senzorů v reálném a simulovaném prostředí. Z provedených experimentů můžeme konstatovat že byly splněny požadavky pro nasazení v dané aplikaci.

Vytvořená infrastruktura pro provoz senzorů slouží především k ověření možnosti nasazení a dosažení požadovaných parametrů. Převážná část zdrojových kódu byla pro jednoduchost napsána v jazyce Python, který je velmi vhodný pro experimentování ale méně už pro ostré nasazení v tak kritických aplikacích jako je monitorování dopravy. Proto bude tento výsledek projektu využit jako “golden standard” a bude následně re-implementována do stávajících systému Camea.

Navržená infrastruktura je rozumně škálovatelná na systémy obsahující desítky senzorů, což je více než dostatečné pro potřeby firmy Camea v oblasti monitorování dopravy, avšak například v průmyslových aplikacích toto může být limitující. Proto předpokládáme další možné rozšíření výsledku v oblasti zpracování dat ze stovek a více senzorů, kde bude nutné vypracovat nástroje pro inicializaci a správu těchto senzorů, případně vytvořit mechanismy pro přerozdělování zátěže při ukládání a ověřování dat na více uzlů v cloudu.