Hashovací techniky

Seminární práce KI/AKA

MILAN GITTLER

Obsah

Hashovaní....1
Hashovací funkce: Klíčový prvek moderní kryptografie a informatiky....1
Strategie řešení kolizí v hashovacích tabulkách....2
Salting a další způsoby zabezpečení hashovaných dat....3
HMAC: Ověření integrity a autentičnosti zprávy....3
Perceptuální hashování....4
Praktické aplikace hashování....5
Závěr....6
Seznam použitých zdrojů....7

Hashovaní

Hashování je proces používaný v informatice k převodu vstupních dat variabilní délky na výstup pevné délky, čímž vznikne výsledná hash hodnota (česky otisk). Tato technika se široce využívá pro efektivní ukládání a vyhledávání dat ve strukturách zvaných hashovací tabulky. Klíčovou vlastností hashovací funkce je, že i malá změna vstupních dat vede k výrazně odlišné hash hodnotě, což je známé jako lavinový efekt. To je zásadní pro zabezpečení dat, protože zajišťuje, že odvození původních dat z hash hodnoty je extrémně obtížné. Kromě toho se hashování používá v kryptografii pro ověření integrity dat, v databázích pro rychlé vyhledávání a v distribuovaných systémech pro konzistentní hashování a rozložení zátěže.

Hashovací funkce: Klíčový prvek moderní kryptografie a informatiky

Hashovací funkce představují základní stavební bloky v mnoha oblastech informatiky, od zabezpečení dat až po efektivní ukládání a vyhledávání informací. Tyto funkce jsou navrženy tak, aby z libovolně dlouhého vstupu vytvořily výstup pevné délky, zvaný hash hodnota. Tento odstavec poskytne přehled nejvýznamnějších hashovacích funkcí, vysvětlí jejich klíčové charakteristiky a poukáže na jejich praktické aplikace.

Vytvoření hash hodnoty začíná přijetím libovolného množství vstupních dat. Tato data jsou pak zpracována hashovací funkcí, která provádí řadu komplexních matematických a logických operací, aby vyprodukovala výstupní hodnotu pevně dané délky. Tato délka je specifická pro každou hashovací funkci; například SHA-256 vždy produkuje hash hodnotu délky 256 bitů, bez ohledu na velikost nebo délku původních vstupních dat. Proces vytvoření hashovací hodnoty je jednosměrný: zatímco generování hash hodnoty z daného vstupu je rychlé a přímé, pokus o zpětné získání původních dat z hash hodnoty je prakticky nemožný kvůli vysokému stupni komprese a bezpečnostním vlastnostem hashovací funkce

Hashovací funkce musí splňovat několik kritérií, aby byla považována za bezpečnou:

1. **Determinismus:** Stejná vstupní data musí vždy generovat stejnou hash hodnotu.
2. **Rychlost výpočtu:** Hashování musí být provedeno rychle, i pro velké objemy dat.
3. **Lavinový efekt:** I malá změna v jednom bitu vstupních dat musí vést k radikálně odlišné hash hodnotě.
4. **Ochrana proti kolizím:** Mělo by být velmi obtížné najít dva různé vstupy, které produkují stejnou hash hodnotu.

Příklady Hashovacích Funkcí

- **MD5:** Jedná se o starší hashovací funkci, která produkuje 128bitový hash. Přestože byla široce používána, dnes se kvůli bezpečnostním slabostem a schopnosti najít kolize již nedoporučuje pro bezpečnostní aplikace.

- **SHA-1:** Další starší funkce generující 160bitový hash. Podobně jako MD5, i SHA-1 je považováno za zranitelné kvůli možnosti najít kolize, což vedlo k jejímu postupnému nahrazení bezpečnějšími alternativami.
- **SHA-256 a SHA-3:** Tyto funkce patří mezi současné standardy pro kryptografické hashování, nabízejí vyšší stupeň bezpečnosti a jsou odolné vůči známým útokům. SHA-256 je součástí SHA-2 rodiny hashovacích algoritmů, zatímco SHA-3 je jeho následovníkem, který přináší další zlepšení v oblasti bezpečnosti a efektivity.

Hashovací funkce se používají v široké škále aplikací, od zabezpečení digitálních podpisů a ověřování integrity dat až po implementaci bezpečného ukládání hesel a efektivního vyhledávání v databázích. V kryptoměnách, jako je Bitcoin, hashovací funkce umožňují "těžbu" nových mincí a zajišťují integritu blockchainu. Hashovací funkce jsou nezbytnou součástí moderní informatiky a kryptografie. Výběr správné hashovací funkce a její aplikace může mít zásadní vliv na bezpečnost a efektivitu systému. Ačkoli se bezpečnostní standardy neustále vyvíjejí, základní principy a aplikace hashování zůstávají klíčové pro ochranu a správu digitálních dat.

Strategie řešení kolizí v hashovacích tabulkách

Při používání hashovacích funkcí v aplikacích, jako jsou hashovací tabulky, může docházet k situaci, kdy dva různé vstupy generují stejnou hash hodnotu – jevu známého jako kolize. Protože každá hash hodnota by měla ideálně odpovídat jedinečnému vstupu, kolize představují problém, který je nutné řešit, aby hashovací tabulky fungovaly správně a efektivně. Existují různé strategie pro řešení kolizí, z nichž každá má své specifické výhody a použití. Zde jsou dvě nejčastěji používané metody:

1. Řetězení (Separate Chaining)

- Tato metoda spočívá v použití externí datové struktury, jako je spojový seznam nebo strom, pro ukládání všech prvků, které hashují na stejnou hodnotu. Každý slot v hashovací tabulce tedy může obsahovat ukazatel na takovou strukturu, která obsahuje všechny prvky se stejným hashem. Tímto způsobem lze snadno přidávat nové prvky i v případě kolize, avšak s rostoucím počtem kolizí může dojít k poklesu výkonu při vyhledávání.

2. Otevřená adresace (Open Addressing)

- Otevřená adresace předchází kolizím bez použití dodatečných datových struktur. Pokud dojde ke kolizi, algoritmus hledá další volnou pozici v hashovací tabulce podle předem definované sekvence. Mezi běžné metody otevřené adresace se řadí lineární prohledávání, kvadratické prohledávání a dvojité hashování. Tyto metody se liší strategií, jak najít náhradní pozici po kolizi. I když otevřená adresace minimalizuje paměťové nároky, při vysoké hustotě zaplnění tabulky může dojít k poklesu výkonu.

3. Sloučené hashování

- Kromě čistého řetězení a otevřené adresace existuje možnost kombinovat principy obou těchto přístupů, čímž se snažíme využít jejich výhod a minimalizovat nedostatky. Jednou z takových kombinovaných metod je technika zvaná dvojité hashování s řetězením.

Dvojité hashování s řetězením využívá dvojité hashování (které je formou otevřené adresace) pro první úroveň rozptýlení a poté aplikuje řetězení jako sekundární strategii pro řešení kolizí v rámci jednotlivých pozic. Tento přístup umožňuje efektivně distribuovat prvky do hashovací tabulky, čímž se snižuje pravděpodobnost kolizí, ale v případě, že kolize nastane, dojde na využití spojových seznamů nebo jiných struktur uvnitř dané pozice.

Výběr metody pro řešení kolizí závisí na konkrétních požadavcích aplikace, jako je očekávaný počet prvků, přípustná hustota zaplnění hashovací tabulky a požadavky na rychlost vyhledávání. Zatímco řetězení umožňuje tabulce efektivně zpracovávat velké množství kolizí za cenu potřeby dodatečné paměti pro spojové seznamy nebo stromy, otevřená adresace nabízí jednodušší a paměťově efektivnější řešení, které však může trpět poklesem výkonu při vysokém zatížení. Použití kombinovaných metod může vést k lepšímu využití paměti a zlepšení výkonu vyhledávání, zejména v aplikacích, kde je nutné zpracovat velké množství dat s vysokou rychlostí.

Salting a další způsoby zabezpečení hashovaných dat

V moderním světě kryptografie a zabezpečení dat hrají hashovací funkce klíčovou roli nejen při ukládání a ověřování informací, ale také v ochraně proti různým typům útoků. Jednou z hlavních strategií pro zvýšení bezpečnosti hashovaných dat, zejména hesel, je použití techniky známé jako **salting**.

- Salting zahrnuje přidání náhodně generovaného řetězce, známého jako sůl, k uživatelskému heslu před jeho hashováním. Tento proces zaručuje, že stejná hesla od různých uživatelů vygenerují různé hash hodnoty, čímž se značně ztíží útoky hrubou silou nebo útoky pomocí předpočítaných tabulek, jako jsou rainbow tables.
- **Implementace:** Pro každé heslo je generována unikátní sůl, která je uložena společně s hash hodnotou hesla. Při ověřování hesla se stejná sůl přidá k zadanému heslu, a výsledná hash hodnota je porovnána s uloženou hash hodnotou. Díky tomu, že každé heslo má svojí vlastní sůl, je téměř nemožné použít předpočítané tabulky pro dekódování hash hodnot.

Kromě saltingu existují i další techniky a metody zvyšující bezpečnost hashovaných dat:

1. **Peppering:** Podobně jako salting, se peppering používá univerzálně pro všechna hesla. Pepper (pepř) je uložen **odděleně** od databáze hesel, čímž poskytuje další úroveň zabezpečení. Je však třeba řídit jeho uložení a přístup, aby nebyl kompromitován.
2. **Key Stretching:** Techniky jako PBKDF2, bcrypt, nebo scrypt implementují proces zvaný key stretching, který zahrnuje opakované aplikování hashovací funkce na heslo společně se solí po mnoho iterací.

HMAC: Ověření integrity a autentičnosti zprávy

HMAC (Hash-based Message Authentication Code) je specifický typ kódování zpráv využívající hashovací funkce pro ověření integrity a autentičnosti zprávy. Jedná se o významný nástroj v oblasti počítačového zabezpečení, který kombinuje tajný klíč s původní zprávou, přičemž z tohoto spojení se poté generuje hash hodnota. Tato metoda poskytuje bezpečný způsob, jak

zajistit, že zpráva nebyla během přenosu nijak modifikována, a zároveň verifikovat identitu odesílatele.

HMAC využívá kombinaci hashovacích funkcí a kryptografických klíčů k vytvoření ověřitelného digitálního podpisu pro data nebo zprávy. Proces vytvoření HMAC zahrnuje následující kroky:

1. **Spojení klíče se zprávou:** Nejprve se tajný klíč kombinuje se zprávou, která má být odeslána. Tato kombinace zvyšuje bezpečnost, protože hash hodnota nyní závisí nejen na obsahu zprávy, ale také na tajném klíči, který je známý pouze oprávněným stranám.
2. **Aplikace hashovací funkce:** Na kombinaci klíče a zprávy se použije hashovací funkce. Tento krok transformuje data do hash hodnoty, což je výstup pevné délky, který slouží jako digitální podpis zprávy.
3. **Přenos zprávy a HMAC:** Výsledná hash hodnota (HMAC) je připojena k původní zprávě. Tento balíček je poté odeslán příjemci.
4. **Ověření na straně příjemce:** Při přijetí zprávy příjemce opakuje proces HMAC s použitím stejného tajného klíče. Pokud výsledná hash hodnota generovaná příjemcem odpovídá hash hodnotě přiložené k zprávě, lze důvěřovat integritě a autentičnosti zprávy.

Výhody použití HMAC zahrnují vysokou míru bezpečnosti a odolnosti vůči různým útokům, jako je například útok prostřednictvím opakování nebo modifikace zprávy. HMAC najde uplatnění v mnoha aplikacích, včetně zabezpečených komunikací, ověřování API klíčů a digitálního podpisování dokumentů. Jeho efektivita a bezpečnost dělají z HMAC klíčový nástroj pro zajištění důvěrnosti a integrity dat v digitálním světě.

Perceptuální hashování

Perceptuální hashování, známé také jako obsahově citlivé hashování nebo hashování založené na podobnosti, je technika, která umožňuje vytváření hash hodnot reprezentujících multimediální obsah, jako jsou obrázky, audio nahrávky nebo videa, na základě jeho perceptuálních vlastností. Na rozdíl od tradičních hashovacích funkcí, které produkují zcela rozdílné výstupy i při minimálních změnách vstupu, perceptuální hashovací funkce generují podobné nebo identické hash hodnoty pro obsah, který je vnímán jako podobný lidskými smysly. Tato vlastnost činí perceptuální hashování extrémně užitečným pro účely, jako je ověřování pravosti, detekce duplicitního obsahu a vyhledávání založené na podobnosti.

Aplikace perceptuálního hashování:

1. **Detekce a filtrování duplicitního obsahu:** Perceptuální hash může identifikovat obrázky, videa nebo zvukové záznamy, které jsou různé ve smyslu datového obsahu, ale podobné z pohledu uživatele, což pomáhá eliminovat duplikáty nebo nechtěné variace v databázích.
2. **Autorská práva a ochrana obsahu:** Perceptuální hashování umožňuje platformám a službám rozpoznat chráněný obsah nahraný uživateli tím, že srovnává hash hodnoty nahrávek s databází hash hodnot chráněného materiálu.
3. **Vyhledávání založené na podobnosti:** Využití v oblastech, jako je obrázkové vyhledávání a management digitálních aktiv, kde uživatelé mohou hledat obsah na základě vizuální nebo auditivní podobnosti s referenčním vzorem

Praktické aplikace hashování

Hashování najde své uplatnění v široké škále oblastí v informatice a digitální komunikaci, od zabezpečení dat a autentizace uživatelů až po optimalizaci vyhledávání a efektivní správu databází. Tento odstavec poskytuje přehled některých z nejzajímavějších a nejpoučnějších příkladů využití hashování v praxi:

Zabezpečení a ochrana soukromí

- **Ukládání hesel:** Jednou z nejběžnějších aplikací hashování je bezpečné ukládání uživatelských hesel. Místo uchovávání hesel v čitelné formě se v databázích ukládají jejich hash hodnoty, což ztěžuje zneužití hesel v případě úniku dat.
- **Digitální podpisy:** Hashování se využívá i v procesu vytváření digitálních podpisů, což umožňuje ověřit původ a integritu elektronických dokumentů a transakcí, což je klíčové pro elektronické obchodování a právní komunikaci.

Optimalizace a výkon

- **Vyhledávání v databázích:** Hashovací tabulky umožňují rychlé vyhledávání a přístup k datům, což je neocenitelné v aplikacích vyžadujících vysoký výkon, jako jsou vyhledávače nebo systémy pro správu obsahu.
- **Deduplikace dat:** Perceptuální hashování a podobné techniky pomáhají identifikovat a eliminovat duplicitní nebo velmi podobný obsah v digitálních archívech, což šetří úložný prostor a zjednodušuje správu dat.

Inovace a výzkum

- **Blockchain a kryptoměny:** Blockchain technologie využívá hashování pro zajištění integrity a neproměnnosti záznamů. Kryptoměny jako Bitcoin se spoléhají na hashování nejen pro potvrzování transakcí, ale i pro těžbu nových měn.
- **Biometrické systémy:** V biometrických autentizačních systémech se hashování používá k ochraně biometrických dat, jako jsou otisky prstů nebo skenování sítnice, čímž se zvyšuje bezpečnost a ochrana soukromí uživatelů.

Tyto příklady ukazují, jak je hashování nezbytným nástrojem pro řešení široké škály technických a bezpečnostních výzev v digitálním věku. Jeho schopnost rychle a bezpečně zpracovávat informace nachází uplatnění ve stále rostoucím spektru aplikací, od základních funkcí zabezpečení až po pokročilé analytické a správní systémy.

Závěr

V této seminární práci jsme se zabývali širokým spektrem témat spojených s hashováním, od základních principů a technik přes strategie řešení kolizí až po pokročilé metody zabezpečení. V průběhu naší analýzy jsme objevili, jak je hashování nezbytné pro zajištění integrity dat, bezpečné ukládání hesel, efektivní vyhledávání v databázích a mnoho dalších aplikací v informatice a kryptografii.

Jednou z klíčových lekcí z této seminární práce je, že i přes svou zdánlivou jednoduchost, hashování představuje komplexní oblast s hlubokými důsledky pro bezpečnost a výkon systémů. Jak technologie pokračuje ve svém rychlém vývoji, tak i potřeba robustních hashovacích technik roste, aby dokázaly čelit stále sofistikovanějším útokům. Rovněž jsme zdůraznili význam správného výběru hashovací funkce, která musí splňovat kritéria jako rychlost, odolnost proti kolizím a schopnost odolávat pokusům o inverzi. Moderní hashovací funkce jako SHA-256 a SHA-3 jsou příkladem, jak může být hashování účinně použito k zabezpečení dat, ačkoli je stále důležité zůstat ostražitý vůči potenciálním bezpečnostním slabostem. Perceptuální hashování nám ukázalo, jak mohou být principy hashování rozšířeny do nových oblastí, jako je detekce duplicitního obsahu a autorská práva, přinášejíc nové možnosti pro správu a ochranu digitálního obsahu.

V závěru je třeba zdůraznit, že hashování je životně důležitou technologií, která se dotýká mnoha aspektů digitálního světa, od zabezpečení a ochrany soukromí až po optimalizaci výkonu a inovace. Jak se budeme ubírat dále do éry digitalizace, význam a aplikace hashování budou nejspíš dále růst. Proto je pro každého informatika zásadní porozumět principům a praxi hashování, aby mohl efektivně navigovat a využívat tuto klíčovou technologii ve prospěch bezpečnějšího a efektivnějšího digitálního prostředí.

Seznam použitých zdrojů

Hashování: https://web.mit.edu/16.070/www/lecture/hashing.pdf

Hashování: https://www.cs.upc.edu/~mjserna/docencia/grauA/T16/Hashing.pdf

Hashování: http://mpesguntur.com/home/PDF/NOTES/CSE/ads/Unit_2_%20Hashing.pdf

Hashovací funkce: https://cs.indstate.edu/~fsagar/doc/paper.pdf

HMAC: https://csrc.nist.gov/files/pubs/fips/198/final/docs/fips-198a.pdf

Perceptualní hashování: https://www.ofcom.org.uk/__data/assets/pdf_file/0036/247977/Perceptual-hashing-technology.pdf