03 Aplikace šifrování (1.5h)

tags: `řsss-řk`

> Aplikace šifrování. Aplikace symetrických blokových šifrovacích algoritmů - integrita a důvěrnost, režimy provozu. Aplikace dalších symetrických algoritmů. Aplikace hašovacích funkcí, MAC. Hybridní kryptosystémy. (PV079)

Bezpečnostní cíle

- Důvěrnost (*Confidentiality*): zprávu mohou číst jen autorizované subjekty :arrow_right: šifrování
- Integrita (*Integrity*): zpráva není nijak modifikována neutorizovaným subjektem :arrow_right: MAC, hash + podpis
- Autentizace (*Authentication*): ověření identity zařízení, procesu, původce zprávy :arrow_right: podpis, heslo/PIN, biometrika, karta, …

Symetrické blokové šifry

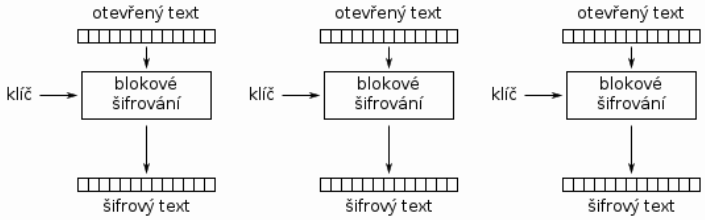
Hlavní myšlenky: * Stejný klíč pro šifrování a dešifrování sdílený mezi oběma účastněnými se stranami * Blokové a proudové šifry * Výhody: rychlejší než asymetrické šifrovací algoritmy, menší klíče * Nevýhody: problémem je distribuce klíče

Implementační problémy algoritmů pro symetrické šifrování: * Zarovnání (*padding*): * Pro zpracování blokovými šiframi musí být plaintext dělitelný požadovanou velikostí bloku * Zarovnání je připojeno k plaintextu, tak je umožněno zpracování po celých blocích, de facto slouží jako "výplň" * Zarovnání je po dešifrování odstraněno * Příklad: PKCS #7 přidává N bajtů hodnoty N (platí pro $N < 256$) pro zarovnání bloku * IV (inicializační vektor/hodnota): * Přidává do šifrování více náhodnosti * Cílem je nešifrovat dva stejné bloky na stejný výsledný šifrovaný text * Má stejnou velikost jako je velikost bloku * Přesné použití IV závisí na použitém operačním módu šifrování * Není tajemstvím, obvykle distribuován se šifrovaným textem * Stejné IV by nemělo být se stejným klíčem použito vícekrát (neumožňuje odvodit vztahy mezi šifrovanými zprávami)

Operační módy / provozní řežimy (DES & AES)

Electronic CodeBook (ECB), blokový mód

- Zpráva je rozdělena do 64bitových nezávislých bloků, které jsou samostatně zašifrovány.
- $C_i = E_K(P_i)$
- Výhody: žádná režie
- Nevýhody: repetice ve zprávě se může projevit v šifrovaném textu!
  - Problém hlavně u zpráv, které se mění jen velmi málo (formáty s vysokou redundancí – video, audio)
  - ECB-šifrovaný obrázek (např. Linux tučňák Tux) má viditelné vzory, zatímco s módy šifrování CTR/CBC vypadá jako náhodný šum
  - Důvod: šifrované bloky jsou na sobě nezávislé (stejné vstupy se mapují na stejné výstupy)
- Vhodné použití: posílání jen pár bloků dat, testování implementace šifry

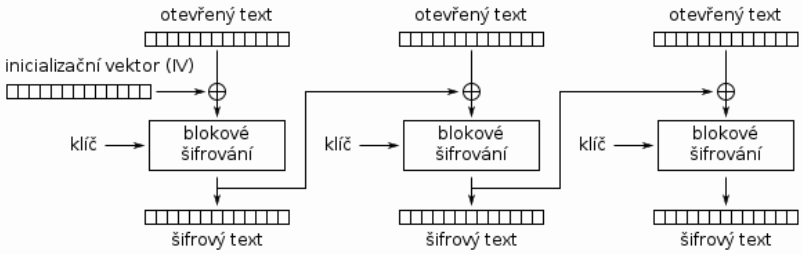


Šifrování v režimu kódové knihy (ECB)

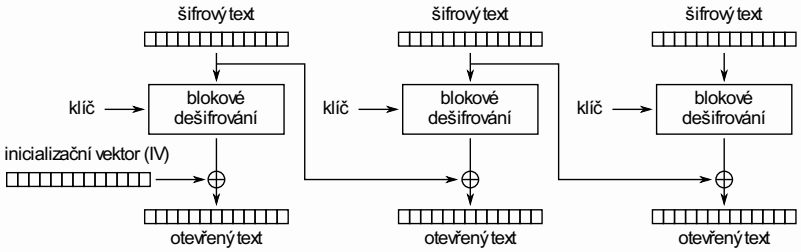
Cipher Block Chaining (CBC), blokový mód

- Zpráva je rozdělena do 64bitových bloků, které jsou XORovány s předchozím šifrovým blokem (začíná se s IV)
- $C_i = E_K(P_i \oplus C_{i-1})$, kde $C_{-1} = IV$
- Výhody: jednoduchá implementace a pochopení, široce používáno
- Nevýhody:
  - Každý šifrový blok závisí na **všech** předchozích blocích zprávy , tj. změna ve zprávě změní celý původní šifrový blok, tak i následující šifrový blok
  - IV musí být znám odesílateli i příjemci – neměl by být zaslán nešifrovaně, musí být buď pevná hodnota, nebo být odeslána šifrovaně v režimu ECB před zbytkem zprávy, pro zaslání IV se doporučuje stejný pricip jako byl použit pro výměnu klíče
  - Na konci zprávy může být poslední blok zarovnán, musí se s tím vypořádat (odstranění z dešifrovaného textu)
  - Z důvodu řetězení není možná paralelizace šifrování či předpočítání (nevhodné pro okamžité použití), dešifrování paralelizovat lze
- Vhodné použití: HTTPS, posílání zpráv
- Nevhodné použití: pevný formát dat bez kontroly integrity dat – útočník může přehodit určitý bit (výsledkem je nesmysl)

Pokud není poslední blok plaintextu stejně velký jako velikost bloku šifry, musí být zarovnán. PKCS #5 je doporučován. * PKCS #5 zarovnání bere hodnoty mezi 1 a 8 pro DES a 1 a 16 bajtů pro AES-128 * Hodnota bajtu = počet přidaných bajtů – pro vyplnění jednoho bajtu je přidáno jedenkrát 0x01, pro osm bajtů osmkrát 0x08… * Výsledná délka je násobek 8, nebo 16, nebo … bajtů * Po dešifrování se přečte poslední dešifrovaná hodnota (*N*) a odebere se daný počet (*N*) bajtů * Rozdíl mezi PKCS #5 a PKCS # 7 je v tom, že PKCS #5 je definován jen pro blokové šifry, které používají 64bitovou (8bajtovou) velikost bloku * V podstatě sekundární kontrola dešifrování - špatně pokud poslední hodnota není mezi 1 a 8 (16…) – kvůli velikosti bloku, nemůže být větší.



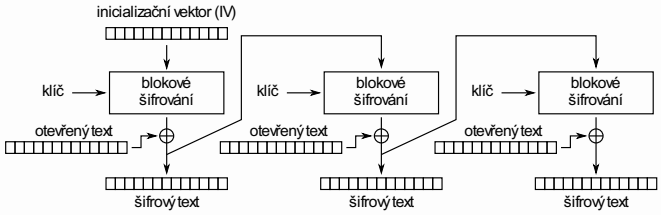
Šifrování v režimu řetězení šifrových bloků (CBC)



Dešifrování v režimu řetězení šifrových bloků (CBC)

Cipher FeedBack (CFB), proudový mód

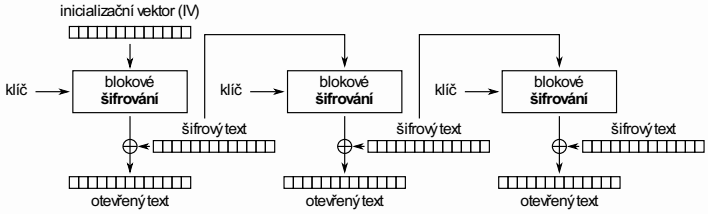
- Zpráva se zpracovává jako proud bitů a přidává se k výstupu (XOR) a výsledek se užívá v další fázi
- $C_i = E_K(C_{i-1}) \oplus P_i$, kde $C_{-1} = IV$
- Postup je velmi podobný jako u CBC, akorát se prohazuje pořadí operací – místo aby se nejprve XORoval otevřený blok s předcházejícím šifrovým blokem a pak výsledek šifroval blokovou šifrou, nejprve zašifruje předchozí šifrový blok a výsledkem XORuje otevřený blok
- Jakýkoliv počet bitů (1, 8, 64, …) může být vložen zpět (CFB-1, CFB-8, CFB-64, …)
- Nejefektivnější je vzít všechny bity v bloku (64 nebo 128)



Šifrování v režimu šifrové zpětné vazby (CFB)

šifrování v CFB módu

- Výhody: podobné CBC, nejběžnější proudový mód, není potřeba zarovnávat, dešifrování lze paralelizovat, dešifrovací funkce je jako šifrovací, jen je XORován šifrový text s výsledkem šifrování
- Nevýhody:
  - Použití jen pro bitově nebo bajtově orientovaný vstup – proudový mód
  - Šifrování nelze paralelizovat
  - Chyby se propagují několik bloků po výskytu chyby (dáno velikostí feedback registru a hodnoty posunu)
  - Je potřeba IV
- Vhodné použití: je podobné CBC, ale CBC je preferováno
- Nevhodné použití: jednoblokové zprávy – útočník může prohodit konkrétní bit a to ovlivní i následující blok, který bude celý poškozen

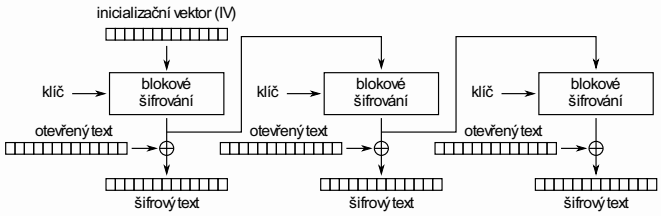


Dešifrování v režimu šifrové zpětné vazby (CFB)

šifrování v CFB módu

Output Feedback (OFB), proudový mód

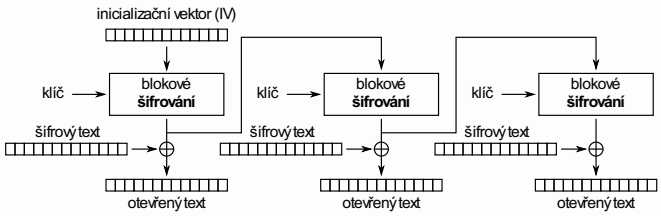
- Zpráva se zpracovává jako proud bitů, přidána ke zprávě, ale feedback je nezávislý na zprávě
- $C_i = O_i \oplus P_i$, kde $O_i = E_K(O_{i-1})$ a $O_{-1} = IV$



Šifrování v režimu zpětné vazby z výstupu (OFB)

šifrování v OFB módu

- Výhody:
  - Podobné CTR (předpočítání), ale méně používáno
  - Sice nelze obecně paralelizovat, ale lze předpočítat výsledek blokové šifry, jelikož plaintext je s tímto výsledkem XORován a takto je možné paralelizovat zbytek výpočtu
- Nevýhody:
  - Nesmí být nikdy znovu použita stejná dvojice klíč + IV (jelikož feedback vkládaný do následující šifry není vázán na plaintext)
  - IV je potřeba (znalost odesílatelem i příjemcem)
  - Původně uváděna s m-bitovým feedbackem, ale měl by se používat pouze celý feedback blok (OFB-64 nebo OFB-128)
  - Útočník může prohodit bit bez porušení celého bloku
- Vhodné použití: vysokorychlostní síťové šifrování (např. streamování – může se předpočítat bloková šifra); všude, kde by šifrování (drahá operace) mělo být provedeno před dostupností zprávy
- Nevhodné použití: data se specifickým formátem (téměř všude kromě multimédií)

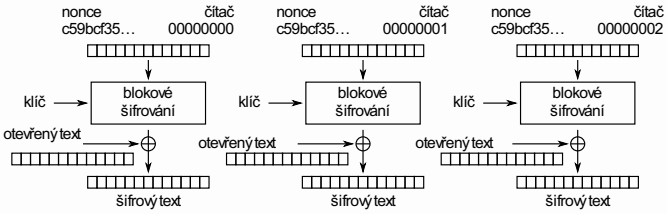


Dešifrování v režimu zpětné vazby z výstupu (OFB)

šifrování v OFB módu

Counter (CTR), proudový mód

- Podobné OFB, ale šifruje počítadlo (*nonce*) místo feedback hodnoty
- Musí mít vždy jiný klíč a počítadlo pro každý blok plaintextu (nikdy recyklovat)
- $C_i = P_i \oplus E_K(i)$ (iniciální hodnota počítadla musí být náhodná)



Šifrování v čítačovém režimu (CTR)

šifrování v CTR módu

- Výhody:
  - Rychlost, je rychlejší než CBC
  - Efektivita – paralelní (de)šifrování, předpočítání
  - Chyby jsou izolované, nepropagují se do dalších bloků
- Vhodné použití:
  - Pro šifrování ve vysokorychlostní síti
  - Všude kde OFB (výše)

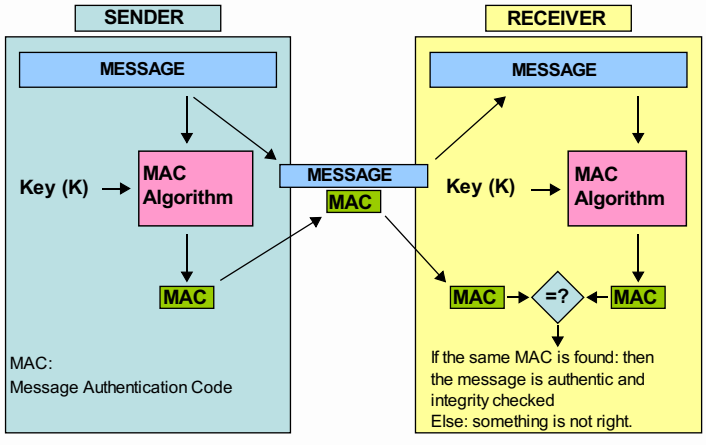
Shrnutí vybraných vlastností šifrovacích módů

	ECB	CBC	CFB	
Předpočítání	:x:	:x:	:x:	:heavy
Paralelní šifrování	:heavy_check_mark:	:x:	:x:	:x:
Paralelní dešifrování	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:	:x:
Náhodný přístup	:heavy_check_mark:	:heavy_check_mark:	:heavy_check_mark:	:x:
Porušených bloků plaintextu při změně 1 šifrového bloku	1	2	2	1

Message Authentication Code (MAC)

MAC je kryptografický kontrolní součet s pevnou délkou pro variabilně velké zprávy. Je to malý blok generovaný pomocí blokové šifry nebo hashovací funkce. Blok závisí na zprávě m a libovolném klíči k, takže MAC = $C_k(m)$. Je to podobné šifrování, avšak není zde podmínka reverzibility.

Výsledný blok je připojen ke zprávě jako pečeť a poslána. Příjemce provede stejný výpočet nad zprávou jako odesílatel a zkontroluje, jestli výsledek sedí s obdrženým MAC. Takto je zajištěna jistota, že zpráva je nepozměněná a přichází od daného odesílatele (autentizace pomocí předem dohodnutého klíče).



Message Authentication Code příklad

Podobně jako u hashovacích funkcí, MAC zhustí (zkompresuje) zprávu do bloku fixní délky. Rozdílnost od hashovací funkce je to, že MAC používá klíč a kromě integrity také poskytuje autentizaci. Nicméně MAC není digitální podpis! Při kombinaci se šifrováním je obecně považováno provést nejprve MAC, potom šifrovat. Ale kombinace mohou být různé.

Požadavky na MAC:

1. MAC je many-to-one funkce (více zpráv může mít stejný MAC). Nicméně, pokud známe zprávu a MAC, musí být nemožné najít jinou zprávu se stejným MAC.
2. MACs by měly být rovnoměrně rozloženy.
3. MAC by měl být stejně závislý na všech bitech zprávy.

Implementace MAC

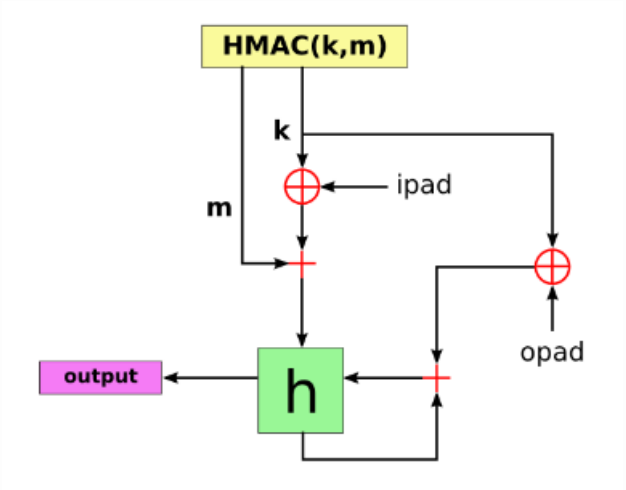
Pomocí symetrické šifry pro MAC (DAA - Data authentication algorithm)

Jakýkoliv mód blokové šifry s zřetězením může být aplikován a finální blok slouží jako MAC. Např. Data authentication Algorithm (DAA) je široce používaný MAC založený na DES-CBC. * Používá IV=0 a nulové zarovnání finálního bloku * Šifruje zprávu pomocí DES v CBC módu * Finální blok je zaslán jako MAC * Nyní je tento výsledek považován z hlediska bezpečnosti za slabý :arrow_right: jdi do AES-256

Pomocí hashovacích funkcí pro mac (HMAC)

MAC založené na hashovací funkci jsou žádoucí, jelikož hashovací funkce jsou obecně rychlejší a kód pro kryptografické hashovací funkce je široce dostupný.

HMAC je počítán jako: `HMAC = Hash[(K+ XOR opad) || Hash[K+ XOR ipad || M]]` `K+` je klíč, který je zarovnán nulami na požadovanou délku a `opad`, `ipad` jsou specifické zarovnávací konstanty definované standardem. Režie je jen o 3 výpočty hashe více, než kolik potřebuje samotná zpráva. Může být použitá jakákoliv hashovací funkce (např. SHA-3, SHA-256, Whirlpool)



HMAC příklad

Prokázaná bezpečnost HMAC souvisí s bezpečnostní hashovacího algoritmu, na kterém je založena. Útok na HMAC vyžaduje buď: * brute-force útok na použitý klíč * birthday útok (ale z důvodu použití klíče, je nutné získat velké množství zpráv)

Bezpečnost hashovacích funkcí a MAC

Stejně jako blokové šifry, hashovací funkce a MAC jsou zranitelné: 1. Útoky hrubou silou: * Relativně snadné cracknutí MD5 :arrow_right: použití alespoň 160bitového hashe * MAC se známou dvojicí zpráva+MAC – možný útok na klíč nebo na MAC :arrow_right: použití alespoň 160bitového hashe 2. Kryptoanalytické útoky zneužívající strukturu: * Stejně jako u blokových šifer, je snaha o to, aby se útočníci zmohli jen na brute-force útoky

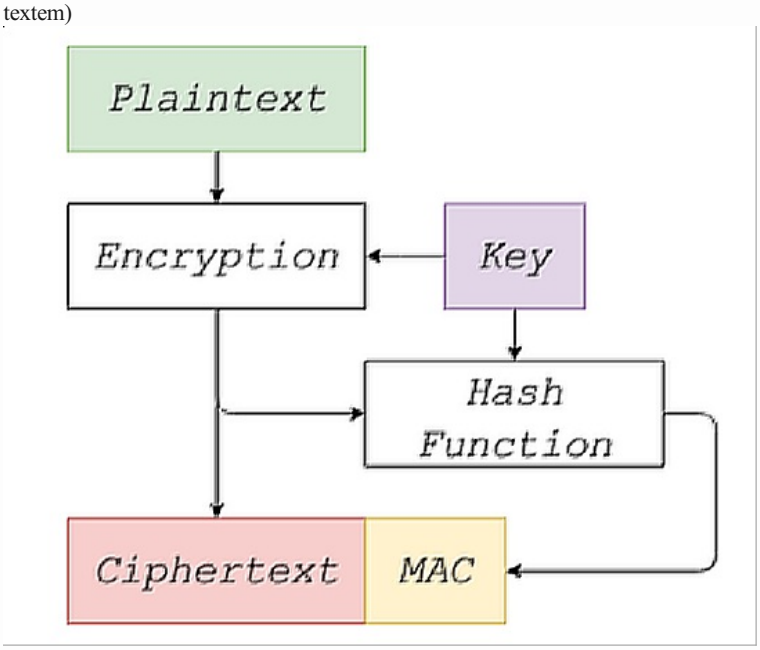
Autentizované šifrování

- Obyčejné šifrování poskytuje pouze důvěrnost, ale ne integritu nebo autentizaci

- Výsledkem autentizovaného šifrování je šifrovaný text a jeho MAC
  - Pokud je MAC neplatný, následované dešifrování neproběhne
  - MAC poskytuje integritu a autentizaci pomocí symetrické kryptografie
  - V porovnání s digitálními podpisy (nejsou to samé!): také poskytují nepopíratelnost, používají asymetrickou kryptografii a podepisují nešifrovaný obsah
  - Obecně, použití rozdílných klíčů pro MAC a šifrování
- Příklad: Galois/Counter Mode (GCM) je mód použití blokových šifer, který poskytuje autentizované šifrování

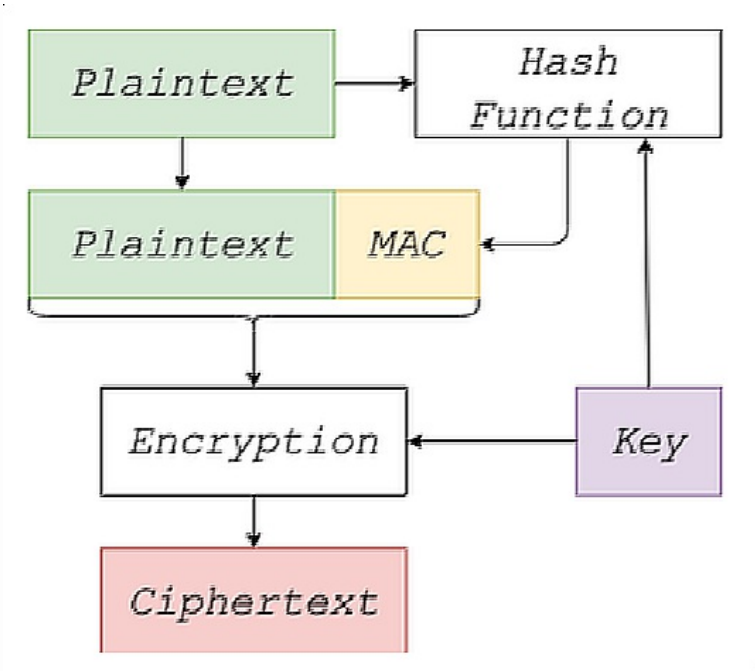
Encrypt-then-MAC (or Encrypt-then-Authenticate)

- Zašifrovat plaintext, pak spočítat MAC nad šifrovaným textem a připojit to k šifrovanému textu (zahrnuje IV a šifrovací metodu)
  - Integrita plaintextu a šifrovaného textu
  - MAC neodhaluje žádné info o plaintextu (protože MAC je spočítán nad šifrovaným textem)



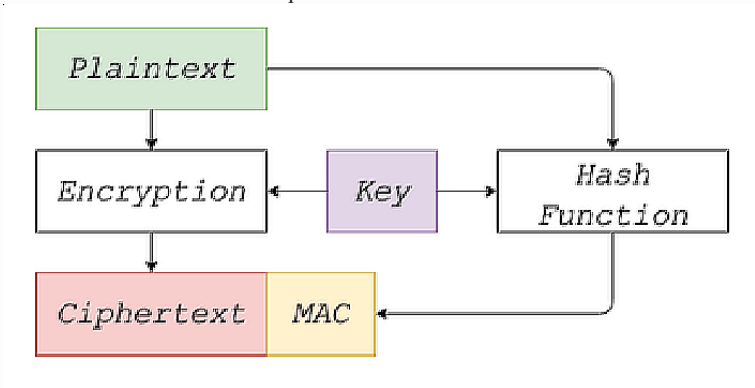
MAC-then-Encrypt (or Authenticate-then-Encrypt)

- Spočítat MAC nad plaintextem, připojit to k datům a pak celé zašifrovat
  - Integrita plaintextu, ale ne šifrovaného textu (může být měněn)
  - MAC neodhaluje žádné info o plaintextu (protože je šifrován)



Encrypt-and-MAC (or Encrypt-and-Authenticate)

- Spočítat MAC nad plaintextem, zašifrovat plaintext a pak připojit MAC na konec šifrovaného textu
  - Integrita plaintextu, ale ne šifrovaného textu (může být měněn)
  - MAC může odhalit informaci o plaintextu



Hybridní kryptosystémy

Hybridní kryptosystémy kombinují symetrickou a asymetrickou kryptografii. Asymetrická může být použita pro ustanovení spojení (autentizace stran, ustanovení symetrických klíčů). Jelikož je asymetrická kryptografie pomalejší, zbytek komunikace je šifrován s ustanovenými symetrickými klíči. Tento přístup je použit u TLS nebo PGP (příloha e-mailu zašifrovaná pomocí AES, symetrický klíč zašifrovaný veřejným PGP klíčem příjemce). Takto může například fungovat ustanovení klíču pomocí Diffie-Helman, kdy výměna prvotních zpráv protokolu je chráněna asymetrickou kryptografií (je nutná nonce pro čerstvost). * Výhody: - rychlost (použití symetrické kryptografie) - výhodné pro použití s široce distribuovanými uživateli - pomocí použití jednorázových klíču dosažení perfect forward secrecy - strany spolu nemusí sdílet společné tajemství (mají veřejné klíče)

04 Útoky na kryptografické systémy a

protokoly (1.5 hodinky)

tags: `řsss-řk, bezpečnost, PV204, PV079, PA197`

> Útoky na kryptografické systémy a protokoly. Využití hardwaru pro ochranu citlivých/kryptografických dat a jejich operací. Smartcards a jejich role jako bezpečnostního mechanismu.

Útoky

Útoky na hesla

- Offline Brute-force: reálně do 8 chars
- Dictionary
- Patterns: kombinace dictionary + bruteforce (Password[0-9]*)
- Rainbow tables: umožnují time-memory tradeoff, hashe jsou předpočítané a uložené v paměti
- Brute-force+Special HW: parallizace, GPU, ASIC (50 billion computations per second)

Jak se bránit? - Volit silná hesla - žádný hashovací systém nedokáže zachránit velmi slabé heslo - Zvýšit cenu bruteforce - **PBKDF2**: aplikace pseudonáhodné funkce HMAC na heslo se solí, přičemž tento proces je opakován několik krát (1000 - 10k). Výrazně zpomaluje bruteforce ale i zároveň uživatele. Dostatečné zpomalení útočníka se specializovaným HW může vést k nepoužitelnosti pro běžného uživatele s CPU. - **Scrypt**: Memory-hard funkce, namísto času bere pamět, pro jednu operaci vynucuje x bytů v paměti, limituje paralelizaci - **Argon2**: Memory-hard hash funkce, více flexibility než Scrypt, postaven na AES - **Bcrypt**: Implicitně přidává salt k hashi. [link] ### Replay útok Útočník odchytí validní zprávu a jejím přeposláním přesvědčí poctivou stranu o její validitě, přestože byla použita v jiném kontextu než pro který byla prvně vytvořena **Jak se bránit?** - přidání tagu s session ID a counterem nebo hash chain do zpráv - přídání timestamps do zpráv (nutná synchronizace), TTL (time to live) limit

Reflection útok

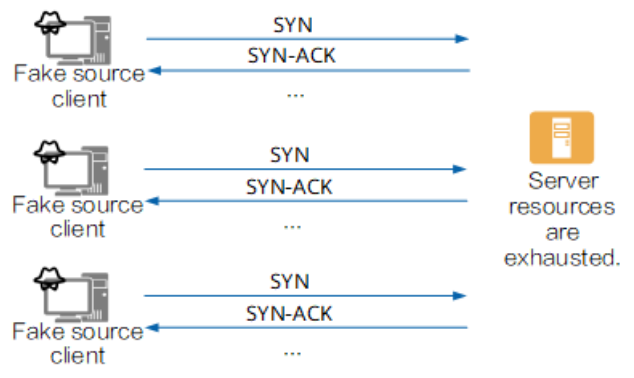
Útočník použije cíl útoku k tomu, aby authentizoval vlastní challenge. Například, jsou dvě zařízení, které se vzájemně autentizují pomocí sdíleného tajemství a HMACu. První zařízení požaduje autentizaci od druhého a pošle mu nonce, jehož HMAC má zaslat zpátky. Druhé zařízení (útočník) si otevře další komunikaci s prvním zařízením a přepošle mu stejný nonce. Validní HMAC pak zašle zpátky a je tak validně authentizované bez nutnosti znalosti sdíleného klíče. **Jak se bránit?** - nepoužívat symetrické výměny v protokolu

reflection attack

SYN flood attack

Záplava pakety SYN neboli SYN-flood je druh útoku označovaný jako Denial of Service. Útočník pošle posloupnost paketů s příznakem SYN cílovému počítači, ale již dále neodpovídá. Pokud se klient pokouší navázat spojení se serverem protokolem TCP, klient a server si za normálních okolností vymění tři TCP pakety:

1. Klient pošle na server paket s příznakem SYN (synchronizovat)
2. Server uzná (acknowledge) žádost o synchronizaci a pošle paket s příznaky SYN a ACK
3. Klient pošle paket s příznakem ACK



Side channels útoky Neinvazivní útoky postranními kanály jsou založeny na nedokonalosti fyzické implementace kryptografických algoritmů. Získávají informace o uložených datech ze zdrojů, které nejsou primárně určeny pro komunikaci.

Časová analýza

využívá faktu, že čas operace závisí na zpracovávaných datech často bývá důsledkem optimalizace nebo špatného návrhu algoritmu

Differenční power analýza

Neinvazivní útok, který využívá statickou analýzu energetické spotřeby různých operací na různých vstupech. Padding, Permutations, etc..

Oracle útoky

Útok, který využívá padding zprávy. Možný u CBC modu. Dobře popsané zde zdroj

Microarchitectural útoky – Meltdown, Spectre

Cache Timing útoky

Cache Timing útoky využívají časové rozdíly mezi přístupem k datům uloženým v mezipaměti a datům, která nejsou uložena v mezipaměti. Vzhledem k tomu, že přístup k datům uloženým v mezipaměti je rychlejší, může program zkontrolovat, zda jsou jeho data uložena v mezipaměti měřením doby, kterou k nim potřebuje.

V jedné formě útoku útočník zaplní mezipaměť svými vlastními daty. Když k datům přistupuje oběť, která používá stejnou mezipaměť, data oběti se přenesou do mezipaměti. Protože velikost mezipaměti je konečná, načtení dat oběti do mezipaměti vynutí některá data útočníka z mezipaměti. Útočník poté zkontroluje, které části jeho dat zůstávají v mezipaměti, a z těchto informací vyvodí, jaké části paměti oběti byly zpřístupněny.

Meltdown a Spectre

Meltdown a Spectre jsou dva útoky z 2017, které využívají kritické zranitelnosti moderních procesorů. Tyto zranitelnosti hardwaru umožňují programům krást data, která jsou aktuálně zpracovávána v počítači. Zatímco programy obvykle nemají povoleno číst data z jiných programů, škodlivý program může zneužít Meltdown a Spectre k získání tajemství uložených v paměti jiných spuštěných programů. * Meltdown - allows to read memory you have no privilege to access to. Exploits speculative execution - procesor řeší oprávnění instrukce až potom co je vyhodnocená jako oprávněná, mezivýsledky ale zůstávají v cachi než se přepíšou.

- Spectre - allows to read any byte in current process virtual memory (even isolated in sandboxes).

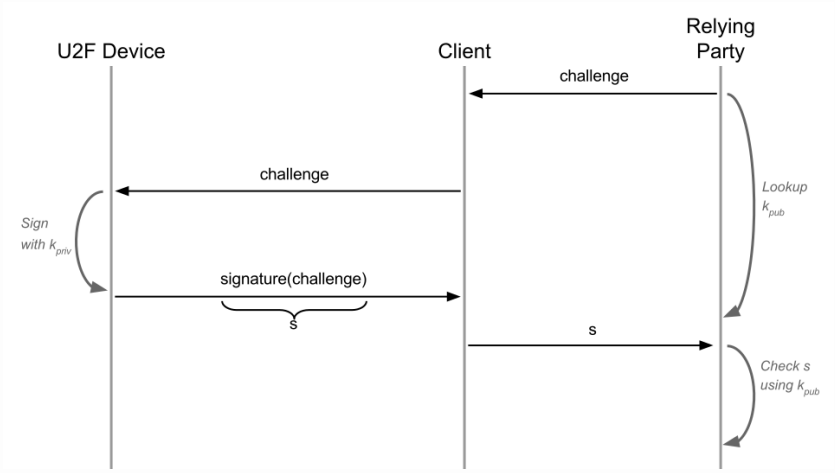
zdroj zdroj

Spectre dovoluje přečíst paměť pouze v rámci sdílené virtuální paměti (např. v rámci jednoho programu - jeden tab browseru může přečíst informace využívané druhým tabem), zatímco Meltdown využívá speculative execution, aby přečetl informace ze zabezpečené části paměti.

HW Ochrana citlivých dat

FIDO U2F tokens

authentizační tokeny, náhrada hesel tokenem s asymetrickým párem klíčů + challenge-response protokolem, např. Yubikey. Token ověřuje dvě věci během authentizace: 1. uživatel má fyzický přístup k zařízení, které bylo předem zaregistrované 2. uživatel je aktivní během authentizace (zmáčkne čudlík na tokenu)



TPM - Trusted platform module

Kryptografické SM, které jsou vevnitř/připojené k dalšímu zařízení. Cena v desítkách dolarů. Může být umístěn: 1. Samostatný čip na základní desce 2. Zakorporované do CPU 3. Software TPM - není HW

Použití: - Bezpečné vytváření a ukládání kryptografických klíčů - Potvrzení, že operační systém a firmware v zařízení jsou takové, jaké mají být, a nikdo s nimi nemanipuloval - šifrování/dešifrování disku

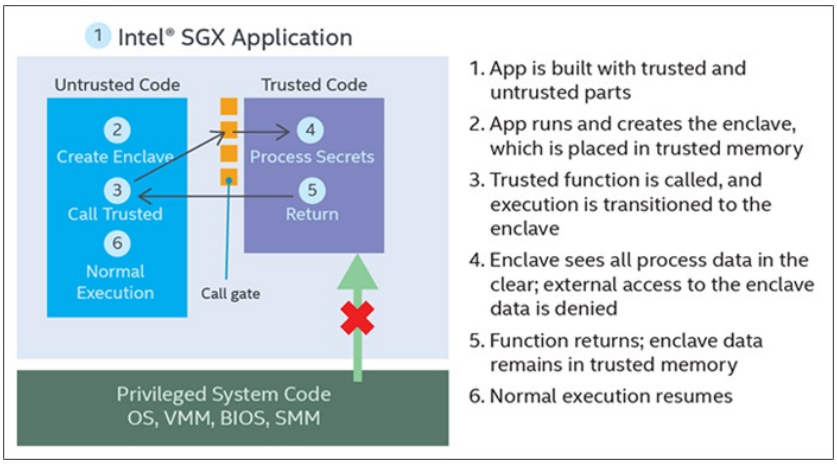
Hardware Security Modules (HSMs)

HSM je bezpečnostní zařízení, které lze přidat do systému pro správu, generování a bezpečné ukládání kryptografických klíčů. Vysoce výkonné HSM jsou externí zařízení připojená k síti pomocí TCP/IP. Jejich cena je ve stovkách a tisících dolarů (enterprise usage).

Použití: šifrování, dešifrování, generování klíčů, PRNG, podepisování / ověřování podpisu. zdroj - What are the differences between TPM and HSM? ### HW peněženky na kryptoměnu bezpečné uchování klíčů + podepisování transakcí, offline, Trezor nebo Ledger zdroj

Intel SGX

SGX (Software Guard Extensions) je rozšíření architektury Intel Instruction Set Architecture (ISA) umožnuje lépe zabezpečit software. Poskytuje především instrukce pro vytvoření izolovaného prostředí – Enclave. Důvěryhodný HW vytvoří tento bezpečný kontejner a uživatel vzdálené výpočetní služby nahraje požadovaný výpočet a data do zabezpečeného kontejneru. Důvěryhodný HW pak chrání důvěrnost a integritu dat, když jsou na nich prováděny výpočty.



Smartcards

SM je malý levný počítač navrhnutý tak aby byl co nejvíce bezpečný. SM je relativně **výkonné zařízení**, které je přitom dostatečně levné a přenosné. Primárně jsou používané v telekomunikacích SIM a finančním sektoru (bankovní karty, ePassports. Dnes spoustu secure HW používá SM – TPM, FIDO U2F tokeny, cryptocurrency HW peněženky - 8-32 bit CPU @ 5-50MHz - persistent memory 32-200+kB (EEPROM) - volatile fast RAM, usually <<10kB - truly random number generator - cryptographic coprocessor + alg (RSA, ECC, AES128, ECDSA)

SM benefity

- má **výpočetní schopnosti** – může šifrovat a podepisovat data, klíč nemusí nikdy opustit SM
- obsahuje standardizované crypto algoritmy
- **bezpečné uložiště** - management klíčů a citlivých dat
- má fyzickou ochranu
- narozdíl od např. smartphonů, je SM mnohem **jednoduší zařízení**, méně komplexity, méně bugů

Fyzická ochrana karty

- Tamper-evidence (fyzický útok bude vždy viditelný)
- Tamper-resistance (fyzický útok je náročný)
- Tamper-response (fyzický útok je detekovatelný a SM na něj zareaguje - vymazání klíčů)

Útoky na karty

invazivní útoky: fyzický útok na kartu s cílem dostat se k paměti s citlivými údaji *semi-invazivní útoky*: částečně fyzické poškození, ale čip funkční (např. expose communication bus, read data by microprobe) *side-channel útoky*: neplánovaný únik data SM přes různé kanály, které nejsou určené k přenosu dat např. power consumption analysis, timing attack *logické útoky*: nefyzické remote útoky, které většinou manipulují s komunikací terminálu a SM, Man-in-the middle attacks