Bezpečnost a integrita dat

Principy a technologie, šifrování

PK-PT1 přednáška 7

Financováno Evropskou unií

NextGenerationEU

Národní plán obnovy

FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

České vysoké učení technické v Praze
Fakulta informačních technologií

Úvod

- “I” v metodice STRIDE znamená únik informací (“Information Disclosure”) a obvykle se omezuje za pomoci šifrování.
  - nemyslete si, že když data šifrujete, tak jste v naprostém bezpečí!
  - bezpečnost závisí mj. na:
    - volbě šifry kvalitě generátoru
    - náhodných čísel způsobu
    - generování klíče/ů způsobu
    - skladování klíče/ů a mnoha dalších
    - faktorech
- “T” v metodice STRIDE znamená neoprávněnou manipulaci s daty (“Data Tampering”) a obvykle ji detekujeme použitím
  -
  -

digitálních otisků
digitálních podpisů

Volba šifry

- Šifer existuje mnoho.
- Kvalitních šifer existuje málo.
- Proprietární šifry (např. Crypto1) byly typicky prolomeny právě díky tomu, že jejich navrhovatelé nespolupracovali s komunitou a dopustili se při návrhu těchto šifer závažných chyb, které nebyly včas odhaleny a opraveny.

→→ Nepokoušejte se navrhnout své vlastní šifrovací algoritmy.Používejte šifry, které jsou prověřené komunitou.

- Typ šifry a délku šifrovacího klíče volíme podle povahy dat a prostředí.
- Fakt, že zvolíme k použití šifru s dlouhým klíčem, automaticky neznamená, že naše data budou v bezpečí!

Při volbě šifry je třeba pamatovat i na to, jak ji správně použít (např. operační módy blokových šifer, zarovnání dat, …)

Symetrické šifry I

Proudové šifry

- ~~RC4~~, ChaCha20, Vernamova šifra, …
- Jsou velmi rychlé.
- Nemusíme se zabývat zarovnáním dat — šifrujeme po bitech/bytech.
- Pokud je stejný klíč použit vícekrát a zná-li útočník otevřený i šifrový text ke stejné zprávě, máme hrozbu typu Information Disclosure pro další zprávy šifrované tímtéž klíčem.

- Pokud je stejný klíč použit vícekrát a zná-li útočník 2 šifrové texty, jejich XOR (= XOR otevřených textů) může podrobit frekvenční analýze a dešifrovat část zpráv.

Blokové šifry

- ~~DES~~, ~~3DES~~, AES, …
- Jsou o dost pomalejší než proudové šifry

Symetrické šifry II

Mají operační módy (~~ECB~~, CBC, CFB, OFB, CTR, MAC, …)

I u blokových šifer se snažte zabránit použití stejného klíče vícekrát!

Pokud musíte používat stejný klíč/heslo, používejte navíc solení.

(Kryptografická) Sůl

Sůl představuje náhodná data, která bývají dalším vstupem do hashovací funkce, do které vstupuje klíč/heslo. Sůl, na rozdíl od klíče/hesla, může být přenášena po médiu bez nutnosti ji šifrovat. Díky jejímu použití budou i stejně vypadající data různě šifrována a pro útočníka bude situace mnohem obtížnější.

Asymetrické šifry I

Šifrování párem veřejný-soukromý klíč.

-
- Velmi pomalé, protože vyžadují operace nad velkými čísly.
  - RSA, ElGamal, McEliece, algoritmy založené na mřížkách, …
- Je nutné zajistit dobrou ochranu soukromého klíče.
- Používají se při digitálním podpisu.
  - Mac OS X — od verze 10.5 je možné podepisovat kterýkoliv Mach-O executable včetně balíčků `.pkg` a `.mpkg`.
  - Windows — používá technologii Authenticode. Podepsat lze jakýkoliv PE soubor, Active-X control, ovladač, MSI instalační skript, … Windows automaticky při prvním spuštění zobrazují informaci o podpisu. Je-li podpis přítomen a platný, a je-li podepsán certifikátem vystaveným důvěryhodnou CA, je úroveň varování nižší.

Srovnání symetrických a asymetrických šifrovacích algoritmů I

Počet bitů	**Sym. alg.**	**FFC** (D-H, DSA)	**IFC** (RSA)	**ECC** (ECDSA)
80	2TDEA (2DES)	$L = 1024, N = 160$	$k = 1024$	$f = 160 - 233$
112	3TDEA (3DES)	$L = 2048, N = 224$	$k = 2048$	$= 224 - 255$
				$f = 256 \quad 383$
128	AES-128	$L = 3072, N = 256$	$k = 3072$	
192	AES-192	$L = 7680, N = 384$	k	$f \quad -$
				$f = 384 - 511$
				$f \geq 512$
256	AES-256	$L = 15360, N = 512$	k	

Tabulka: Srovnání počtu bitů zabezpečení [4]. Sloupec **Počet bitů** zohledňuje známé útoky na šifry, a proto je hodnota nižší než délka klíče. Sloupec **FFC** označuje kryptografii nad konečným tělesem, kde L je bitová délka veřejného klíče a N bitová délka soukromého klíče. Sloupec **IFC** označuje algoritmy založené na celočíselné faktorizaci a k značí velikost modulu v bitech. Sloupec **ECC** značí kryptografii eliptických křivek a f označuje délku veřejného klíče v bitech.

Srovnání symetrických a asymetrických šifrovacích algoritmů II

Roky	**Sym. alg. a MAC**	**FFC** (D-H, DSA)	**IFC** (RSA)	**ECC** (ECDSA)

... – 2010	2TDEA (2DES), 3TDEA (3DES), AES-128, AES-192, AES-256	min. $L = 1024$, $N = 160$	min. $k = 1024$	min. $f = 160$
... – 2030	3TDEA, AES-128, AES-192, AES-256	min. $L = 2048$, $N = 224$	min. $k = 2048$	min. $f = 224$
2030 – ...	AES-128, AES-192, AES-256	min. $L = 3072$, $N = 256$	min. $k = 3072$	min. $f = 256$

Tabulka: Doporučené délky klíčů [4].

Generátory náhodných čísel I

- Kvalita generátoru náhodných čísel je základním požadavkem pro to, aby mohl být použit pro kryptografické účely.
- Kvalitní generátor (pseudo)náhodných čísel musí splňovat:

Kvalitní generátor (pseudo)náhodných čísel

1. generovaná čísla musí být nepredikovatelná
   -
   -

next bit test — ze znalosti posloupnosti vygenerovaných bitů nelze odvodit hodnotu dalšího bitu

state compromise — ze znalosti vnitřního stavu generátoru nelze odvodit hodnoty dosud vygenerovaných bitů

2. generovaná čísla mají mít rovnoměrné rozdělení
3. generátor musí mít dostatečně dlouhou periodu

Generátory náhodných čísel II

Funkce `rand()`, která je součástí standardu ISO/IEC 9899:1990 (ISO C90), není bezpečná pro kryptografické účely.

```
int rand() {
  next = next * 1103515245 + 12345;
  return (unsigned int)(next/65536) %
  32768;
}
```

- Jde o generátor čísel založený na lineární kongruenci.
- Funkce je “thread hostile” — volání více vlákny způsobí časově závislou chybu.

Následující hodnota je snadno predikovatelná → next bit test!!

Předcházející hodnoty lze snadno dopočítat → state compromise!!

→ Pro kryptografické účely nepoužívat.

Generování pseudonáhodných čísel s CryptoAPI I

Generování se provádí funkcí `CryptGenRandom`, která bere v úvahu následující [1]:

ID procesu, ID vlákna, počet tiků procesoru od bootu, lokální čas, různé výkonnostní čítače procesoru, MD4 otisk uživatelova prostředí, cca 200

-
-
  - dalších nízkoúrovňových informací z operačního systému, a případnou
  - další entropii zadanou vývojářem.
- Z toho všeho udělá SHA-1 otisk — semínko pro generátor čísel.
- Vlastní generování probíhá podle standardu FIPS 186-2 příloha 3.1 [3].
- Před generováním je nutné získat `HANDLE` na poskytovatele kryptografických služeb `HCRYPTPROV` funkcí
- `CryptAcquireContext`.
-

V prostředí .Net použijeme objekt `RNGCryptoServiceProvider` z namespace `System.Security.Cryptography`.

Také lze použít CAPICOM od verze 2: objekt `CAPICOM.Utilities.1`, metoda `GetRandom`.

Generování pseudonáhodných čísel s CryptoAPI II

Kryptograficky bezpečné generování pseudonáhodných čísel

```
#include <windows.h>
#include <wincrypt.h>

HCRYPTPROV hProv = NULL;
DWORD error = ERROR_SUCCESS;
BOOL success = FALSE;

/* Získání HCRYPTPROV, stačí zavolat 1x na začátku programu. */
success = CryptAcquireContext( &hProv, NULL, NULL, PROV_RSA_FULL, CRYPT_VERIFYCONTEXT );
if( !success )
  error = GetLastError();

...

/* Generování čísla. Jeden HCRYPTPROV lze použít pro generování opakovaně a i z různých vláken současně. */
success = CryptGenRandom( hProv, pocet_bytu_ke_generovani_do_bufferu, (BYTE*)buffer );
if( !success )
  error = GetLastError();

...

/* Uvolnění HCRYPTPROV. */
if( hProv != NULL )
  CryptReleaseContext( hProv, 0 );
```

Generování pseudonáhodných čísel s OpenSSL

- Nabízí 2 funkce na generování (pseudo)náhodných dat, a to:
- `RAND_bytes`, která generuje kryptograficky silná pseudonáhodná data do bufferu.
- `RAND_pseudo_bytes`, slabší a ne nutně nepredikovatelné.

Kryptograficky bezpečné generování pseudonáhodných čísel

```
#include <openssl/rand.h>

int iErr = 0;
time_t t = time(NULL);
unsigned char buffer[256];

/* Doplnění semínka generátoru */
Err = RAND_seed( &t, sizeof(t) );

/* Generování dat */
iErr = RAND_bytes( buffer, sizeof(buffer) );

/* Uvolnění vnitřních dat generátoru */
RAND_cleanup();
```

Generování pseudonáhodných čísel s OpenSSL

- Nabízí 2 funkce na generování (pseudo)náhodných dat, a to:
- `RAND_bytes`, která generuje kryptograficky silná pseudonáhodná data do bufferu.
- `RAND_pseudo_bytes`, slabší a ne nutně nepredikovatelné.

Kryptograficky bezpečné generování pseudonáhodných čísel?

```
#include <openssl/rand.h>

int iErr = 0;
time_t t = time(NULL); /* Nízká entropie, odhadnutelné */
unsigned char buffer[256];

/* Doplnění semínka generátoru */
Err = RAND_seed( &t, sizeof(t) );

/* Generování dat */
iErr = RAND_bytes( buffer, sizeof(buffer) );

/* Uvolnění vnitřních dat generátoru */
RAND_cleanup();
```

Generování pseudonáhodných čísel s OpenSSL

Generování pseudonáhodných čísel s OpenSSL

- Nabízí 2 funkce na generování (pseudo)náhodných dat, a to:
- `RAND_bytes`, která generuje kryptograficky silná pseudonáhodná data do bufferu.
- `RAND_pseudo_bytes`, slabší a ne nutně nepredikovatelné.

Kryptograficky bezpečné generování pseudonáhodných čísel

```
#include <openssl/rand.h>

int iErr = 0;
time_t t = time(NULL);
unsigned char buffer[256];

/* Doplnění semínka generátoru */
Err = RAND_seed( &t, sizeof(t) ); /* V OpenSSL jen doplněk existující entropie */

/* Generování dat */
iErr = RAND_bytes( buffer, sizeof(buffer) );

/* Uvolnění vnitřních dat generátoru */
RAND_cleanup();
```

Hesla a klíče I

- Kryptografické algoritmy pracují s klíči, ne s hesly.
- Uživatelé si pamatují hesla, ne klíče → často potřebujeme transformovat bity hesla na bity klíče.
- Entropie hesla by měla mít nejméně tolik bitů, kolik má klíč.

Bitová délka hesla pro náhodné heslo

$L = n \cdot \log_2 m$, kde m je počet znaků abecedy hesla a n je počet znaků hesla.

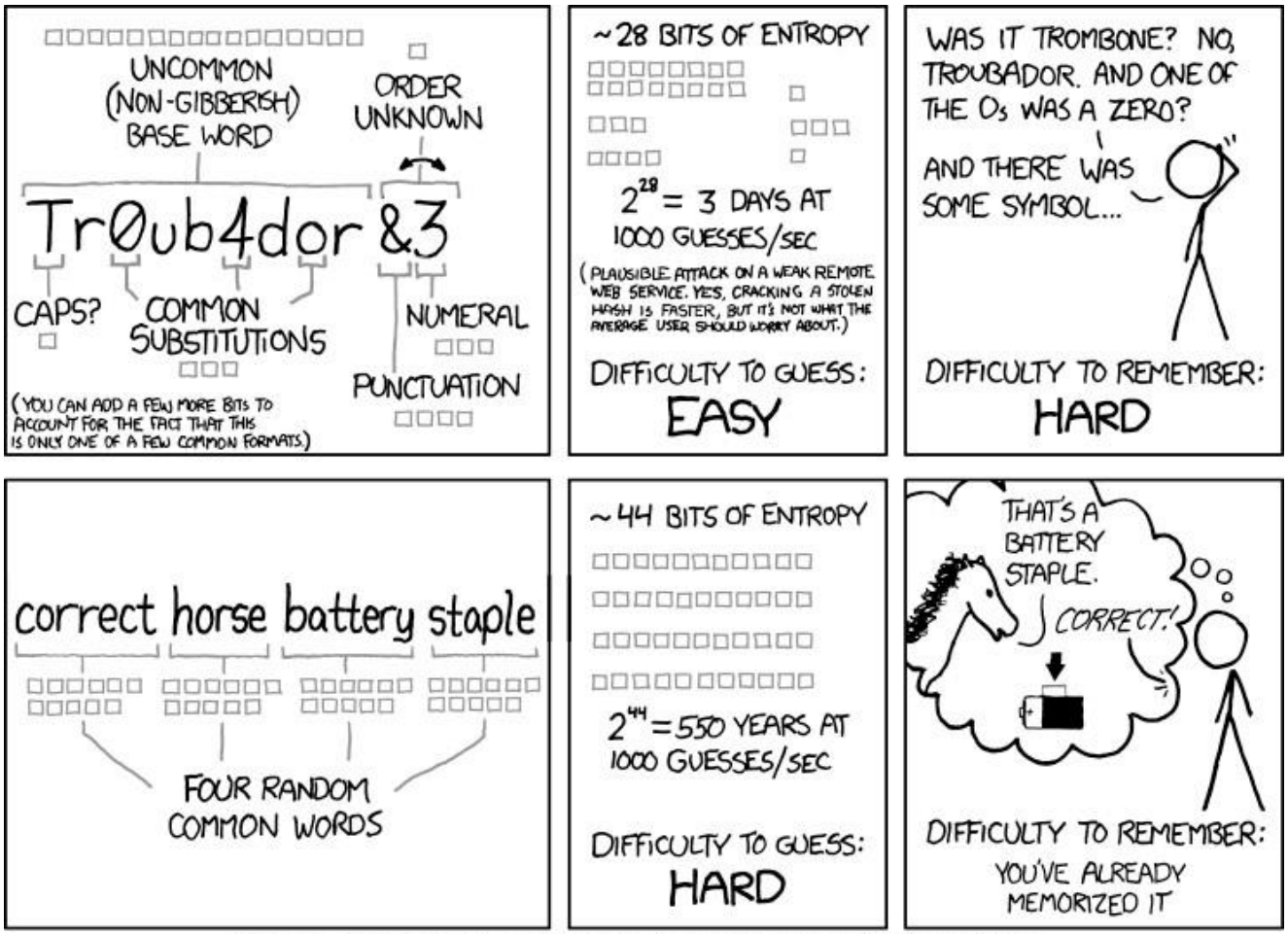
- Ani dlouhé heslo nemusí být bezpečné. Vždy si ověřte jeho odhadovanou sílu (např. http://www.passwordmeter.com/).

Abeceda	m	n pro 56b klíč	n pro 128b klíč

Čísla	10	17	40
Písmena (A-Z nebo a-z)	26	12	28
Písmena (A-Z a a-z)	52	10	23
Alfanumerické znaky	62	10	22
Alfanumerické znaky a speciální znaky	93	9	20

Tabulka: Množiny znaků a délky hesel pro 56b a 128b klíče.

Hesla a klíče II



Obrázek: https://xkcd.com/936/.

Passwords and keys III

Podle “Password trends 2019”[7]:

- Více než 80 % hesel je 7-13 znaků dlouhých. 20 % všech hesel mají pouze 8 znaků.
- Skoro 30 % všech hesel mají tvar písmena+číslo, kde číslo má nejčastěji 2-4 cifry. Dalších 26 % hesel je tvořeno pouze písmeny. Pro zajímavost, hesla ve tvaru číslo+písmena tvoří jen 5.3 % všech hesel.
- Nejběžnější typ hesla je tvořen pouze 8 písmeny (6.7 %).

Nejčastější hesla v roce 2019[8]:

1 123456

2 123456789

3 qwerty

4 password

5 111111
6 12345678
7 abc123
8 1234567
9 password1
10 12345

Hesla a klíče IV

Doporučení pro požadavky na hesla [11]

- Hesla zvolená uživatelem musí být minimálně 8 znaků dlouhá.

Hesla zvolená uživatelem by měla umožnit délku až 64 znaků.

Zakazuje se ořezávání hesel na maximální délku.

Zakazuje se ukládání a zobrazování nápověd k heslům.

Zakazují se dotazy na specifické znalosti uživatele (“Jaký je váš oblíbený film?”)

Zakazuje se používání pravidel na složení hesla (skupiny znaků atd.) Neměly by se používat mechanismy pro pravidelnou změnu hesla.

Vyžaduje se kontrola, že se hesla nevyskytují v seznamech běžně používaných hesel.

Při ověřování hesel musí být použity zpomalovací mechanismy.

Hesla musí být uložena tak, aby byla odolná offline útokům.

...

Generování klíčů z hesel

I

Pro konverzi hesla na klíč je nutné používat ověřenou funkci pro odvozování klíčů (key-derivation function):

- Pro libovolně dlouhý vstup vygeneruje výstup o dané délce.
- Do výstupu jsou zahrnuty všechny bity hesla.
- Musí zahrnovat sůl (salt) jako obranu proti rozpoznání opakovaných hesel.
- Musí být pomalá — obrana proti zkoušení hesel hrubou silou. Často realizováno technikou opakovaného hashování.
- Musí být náročná paměť — obrana proti použití specializovaného hardware.
- Nemusí nutně být bezkolizní.

Generování klíčů z hesel

II

Známé key-derivation funkce:

- MD5, SHA1, SHA2, …— nepoužívat, moc rychlé
- PBKDF2 — pozor na rychlost a nároky na paměť
- bcrypt scrypt
- Argon2 [10] — vítěz Password Hashing Competition (2013-15) [9].

III

Pokud je potřeba změnit způsob uložení hesla [12]

- **Vyresetovat hesla**: Smazat hesla všem, vygenerovat nová. Rychlé, spolehlivé, umožňuje vygenerování bezpečných hesel. Velmi obtěžující pro uživatele, problematická distribuce nových hesel (e-mail?).

Generování klíčů z hesel

- **Vyresetovat zranitelná hesla**: Jako výše, ale jen pro hesla, která se vám podaří cracknout.
  Stejné nevýhody jako výše, plus potřebu provádět útok (možné legální potíže), plus nejistota o tom, jaký typ útoku útočník zvolí.
- **Přeuložit po přihlášení**: Poté, co se uživatel přihlásí, využijeme zadané heslo k uložení nového hashe.
  Uživatel ani nezpozoruje, že se něco děje. Problém s mrtvými účty.
- **Přehashovat všechna hesla**: Hashe hesel jsou přehashovány novým algoritmem. Při ověřování hesel se použijí oba hashe, při změně hesla se uloží už jen nový hash.
  Přátelské vůči uživateli, ale složitější na implementaci. Vyžaduje ověřovat hesla více algoritmy.

Správa klíčů

I

(Kryptografický) Klíč [4]

Kryptografickým klíčem rozumíme parametr kryptografického algoritmu, který určuje jeho chod tak, že entita se znalostí klíče je schopna reprodukovat nebo zvrátit chod algoritmu, zatímco entita bez znalosti klíče ne.

Správa klíčů

Správou klíčů rozumíme nakládání s kryptografickými klíči a souvisejícími parametry (inicializační vektory, hesla) během celého životního cyklu klíčů včetně jejich tvorby, skladování, výměny, vstupu, výstupu a likvidace.

II

Klíče dělíme podle jejich klasifikace jako asymetrické (veřejné, soukromé) nebo symetrické, a podle jejich použití. Podle předpokládané doby života klíče ještě určujeme, jsou-li trvalé (static) anebo dočasné (ephemeral).

Trvalé klíče

Trvalé klíče se používají pro autentizaci, integritu, nepopiratelnost a mohou být použity pro generování dočasných klíčů. Trvalé klíče používáme také k ochraně dat a souborů. Vzhledem k dlouhodobé povaze těchto klíčů je nutné s nimi nakládat co nejbezpečněji, protože budou cílem útoků.

Dočasné klíče

Dočasné klíče jsou používány různými síťovými protokoly, např. IPSec, SSL/TLS, RPC, DCOM, pro ochranu informací s omezenou dobou platnosti. Jejich generování a použití je obvykle před uživatelem skryto.

Správa klíčů

III

Příklad [5]

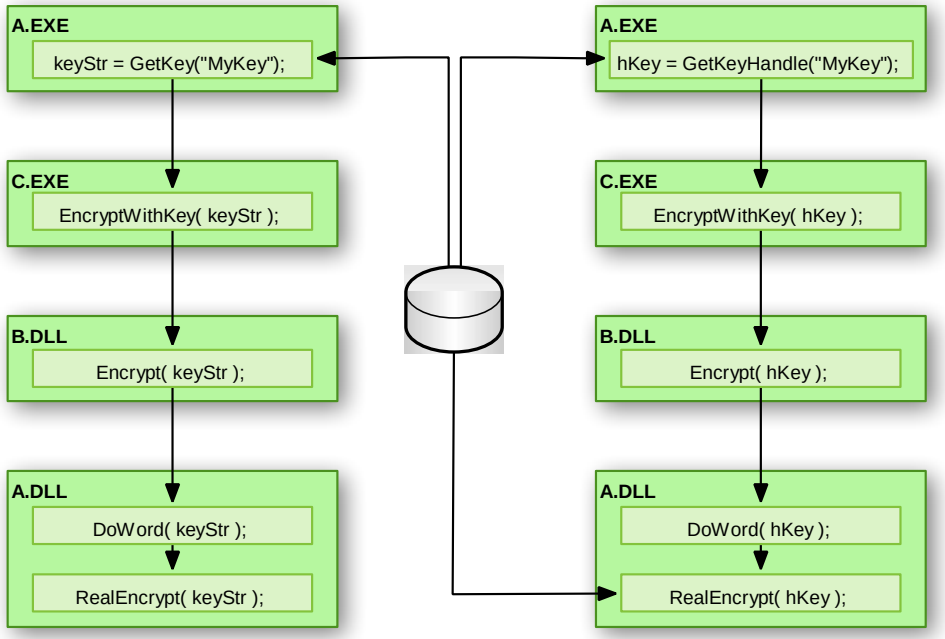
Díky nedostatečně zabezpečenému klíči byla prolomena ochrana DVD. Přehrávač XingDVD Player obsahoval napevno nastavený klíč, který se používal pro softwarové dešifrování DVD. Klíč byl za pomoci reverse engineeringu z přehrávače získán a použit k výrobě programu DeCSS. → Klíče nikdy neukládejte ve zdrojových souborech programu.

- Zásadní otázka: Kam tedy klíče uložit??

Správa klíčů

IV

Klíč by měl v čitelné podobě existovat co nejkratší dobu.



Obrázek: Pohyb klíče [1]. Která z metod na obrázku je bezpečnější?

Správa klíčů

→ Klíč se snažte používat co nejkratší dobu a jen lokálně. Po jeho použití ho explicitně smažte (např. `SecureZeroMemory`).

V

Správa klíčů

Generování klíče/klíčů

- Klíče generujeme pomocí funkce `CryptGenKey` pro asymetrické i symetrické šifry.
- Funkce vrátí `HANDLE` na klíč `HKEY`, dále pracujeme jen s ním.
- Klíč je uložen v poskytovateli kryptografických služeb (CSP).
- Klíč je při generování možno označit za exportovatelný (příznak `CRYPT_EXPORTABLE`).
- Jedině exportovatelný klíč lze z CSP získat v surové podobě funkcí `CryptExportKey`.

Správa klíčů

VI

Výměna klíčů

Výměna klíčů/Zřízení klíče (Key Exchange/Key Establishment) [4]

Proces, během kterého se bezpečně zřídí klíč mezi kryptografickými moduly za použití manuálního přenosu, automatických metod (protokoly pro přenos klíče a/nebo dohodu na klíči), anebo jejich kombinace.

- Neimplementujte algoritmy výměny klíče, použijte protokol, který ji zajistí sám:
  - IPSec, SSL/TLS, …
- Při špatné výměně dochází k úniku informací o klíči.

Správa klíčů

→ Nepoužívejte vlastní algoritmy pro výměnu klíčů.2017 — prolomeno WiFi WPA2, nezávisle na použité šifře

VII

- Používejte bezpečné algoritmy pro výměnu klíčů:
  - Diffie-Hellman, ECDH,
  - RSA výměna klíčů,
  - Kvantová výměna klíče, …
- Pamatujte, že některé klíče se nikdy nevyměňují: soukromý
  - klíč z páru veřejný-soukromý klíč.
-
  -

Správa klíčů

Výměna klíčů může probíhat i jinak než po síti:

- DVD s náhodnými daty pro Vernamovu šifru

Integrita dat

Úvod

- Data je nutno ochránit před jejich neautorizovanou změnou (STRIDE/Data Tampering).
- Daty nemyslíme jen soubory a pakety v síti, ale např. i cookie, které si můžete ukládat na uživatelův počítač, parametry v URL vaší webové aplikace, atd.

- V úvahu připadá použití:
  - různé kontrolní součty (CRC, …)
  - digitální otisky dat (Hashe, …)
  - klíčované hashe (HMAC, …)
  - digitální podpisy (DSA, ECDSA, …)

Nejen, že bychom měli být schopni určit, zda došlo ke změně dat, ale také musíme mít jistotu, že nikdo jiný není schopen kontrolní informaci přepočítat.

Integrita dat

Úvod

- Data je nutno ochránit před jejich neautorizovanou změnou (STRIDE/Data Tampering).
- Daty nemyslíme jen soubory a pakety v síti, ale např. i cookie, které si můžete ukládat na uživatelův počítač, parametry v URL vaší webové aplikace, atd.

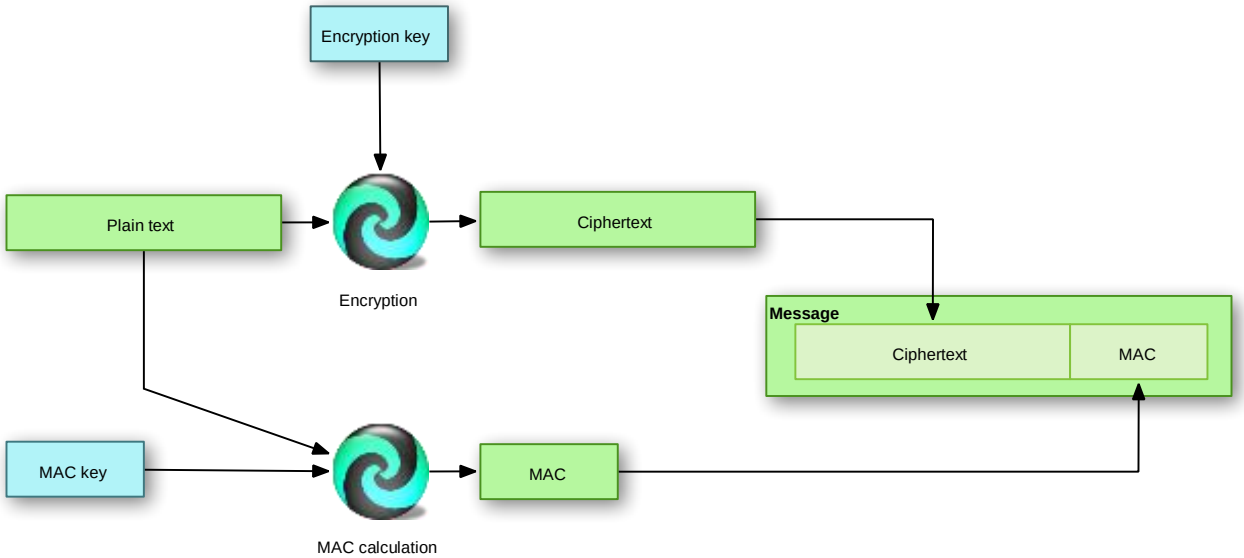
V úvahu připadá použití:

~~různé kontrolní součty (CRC, …)~~
~~digitální otisky dat (Hashe, …)~~
klíčované hashe (HMAC, …)
digitální podpisy (DSA, ECDSA, …)

- Nejen, že bychom měli být schopni určit, zda došlo ke změně dat, ale také musíme mít jistotu, že nikdo jiný není schopen kontrolní informaci přepočítat.

Klíčované hashe (Keyed hash, HMAC) I

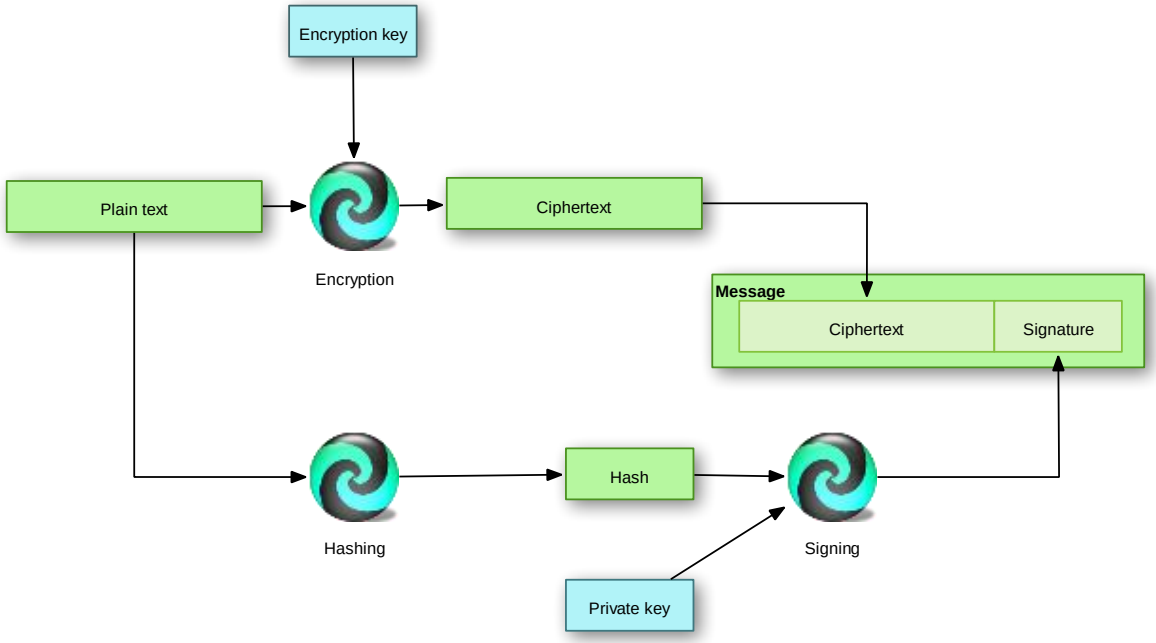
- Šifrovacím klíčem zašifrujeme zprávu a připojíme k ní autentizační kód (Message Authentication Code, MAC) jako klíčovaný hash.
- Klíčovaný hash je hash, který je spočítán z dat a přidané tajné informace (klíč pro MAC). Bez její znalosti nelze MAC spočítat.



Obrázek: Šifrování dat s vytvářením autentizačního kódu pro zprávu (MAC)

Digitální podpisy I

- Digitální podpis vytvoříme tak, že hash zprávy zašifrujeme (= podepíšeme) soukromým klíčem a připojíme ho ke zprávě.
- Digitální podpis ověříme tak, že srovnáme hodnoty hashe zprávy s hodnotou připojeného hashe odšifrovaného pomocí veřejného klíče.



Obrázek: Digitální podpis a šifrování zprávy.

MAC nebo digitální podpis?

MAC

– Pro výpočet klíčovaného hashe se používá hashovací klíč, který musíbýt sdílen.

+ Velmi rychlé.

Digitální podpis

+ Hash se podepisuje soukromým klíčem, podpis se ověřuje veřejným klíčem.

\+ Lze použít pro nepopiratelnost — podpis umí vygenerovat jen vlastník soukromého klíče, ověřit ho však může kdokoliv.

– Pomalejší.

– Problém distribuce veřejného klíče.

Přehled

Hrozba	**Technika vedoucí k nápravě**	**Použitelné algoritmy**
Information Disclosure	Šifrování symetrickou šifrou	~~RC4~~, ~~3DES~~, AES

Tampering	Zajistit integritu dat pomocí hashovacích funkcí, MAC nebo digitálních podpisů.	~~SHA-1~~, ~~SHA-224~~, SHA-256, SHA-384, SHA-512, HMAC, RSA digitální podpis, DSA, XML DSig.
Spoofing	Autentizace dat od přijemce	Certifikáty veřejného klíče a digitální podpisy.

Tabulka: Řešení hrozeb pomocí kryptografie podle jejich typu.

Poznámka

Vyhláška NÚKIB č. 82/2018 Sb. v §26 stanoví, že "[Povinná osoba] používá aktuálně odolné kryptografické algoritmy a kryptografické klíče".[6]

Literatura

I

Howard M., LeBlanc D.: *Writing Secure Code*, 2nd edition, Microsoft Press, 2003.

ISO/IEC 9899: *Information technology — Programming Language C*, 1999.

U.S. Department of Commerce/National Institute of Standards and Technology: *Digital signature standard (DSS)*, https://csrc.nist.gov/publications/fips/archive/fips186-2/fips1862.pdf, 2000.

Literatura

Barker E. et al.: *Recommendation for Key Management – Part 1: General (Revised)*, NIST Special Publication 800-57, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.80057pt1r4.pdf, 2016.

II

Schwartz M.: *DVD encryption hacked*, http://www.cnn.com/TECH/computing/9911/05/dvd.hack.idg/, CNN, 1999.

Literatura

Národní úřad pro kybernetickou a informační bezpečnost: *Vyhláška o bezpečnostních opatřeních, kybernetických bezpečnostních incidentech, reaktivních opatřeních, náležitostech podání v oblasti kybernetické bezpečnosti a likvidaci dat (vyhláška o kybernetické bezpečnosti)*. Vyhláška č. 82/2018 Sb., https://www.govcert.cz/download/kii-vis/NovaVKB/VKB_822018sb.pdf, 2018.

Passware: *Password Trends 2019*, https://www.forensicfocus.com/News/article/sid=3752/, 2019.

Literatura

III

U D.: *Passwords, passwords everywhere*, https://www.ncsc.gov.uk/blog-post/passwords-passwords-everywhere, National Cyber Security Centre, 2019.

Password Hashing Competition: *Password Hashing Competition and our recommendation for hashing passwords: Argon2*, https://password-hashing.net/, 2015.

Biryukov A., Dinu D., Khovratovich D.: *Argon2: the memory-hard function for password hashing and other applications*, version 1.3,

Literatura

https://www.cryptolux.org/images/0/0d/Argon2.pdf, University of Luxembourgh, 2016.

Grassi P. A., Fenton J. L., Newton M. E., et al: *Digital Identity Guidelines*, https://pages.nist.gov/800-63-3/sp800-63b.html, National Institute of Standards and Technology, 2017.

IV

Špaček M.: *Změna hashování existujících hesel*, https://www.zdrojak.cz/clanky/zmena-hashovani-existujicich-hesel/, Zdroják.cz, 2017.