Protokoly, identifikácia a autentizácia

Úvod do informačnej bezpečnosti
(LS 2015/2016)

Michal Rjaško
rjasko@dcs.fmph.uniba.sk

Obsah

- Protokoly
- Autentizácia na základe hesla
  - slabiny, prístupy, PIN, Passkey, jednorázové heslá
- Challenge-response autentizácia
  - zabezpečenie aktuálnosti, časové pečiatky
- Protokoly na autentizáciu s využitím 3. strany
- Zero-knowledge autentizácia
- Zdieľanie tajomstva
- Zraniteľnosti

Protokoly

- Rôzne typy protokolov (účel):
  - výmena (distribúcia/dohoda) kľúča
  - autentizácia subjektu
  - slepé podpisy, voľby, peniaze, ...
- Bezpečnosť závisí na schopnosti útočníka
  - odpočúvať / modifikovať ľubovoľné správy
  - legitímny subjekt prostredia / mimo
  - (Zvyčajne) chceme protokol odolný voči najsilnejšiemu útočníkovi
- Najznámejšie protokoly: SSL/TLS, IPSec, SSH
  - Viaceré varianty – algoritmy, spôsoby autentizácie
- Prostriedky autentizácie účastníka protokolu:
  - Zdieľaná tajná informácia (heslo/kľúč)
  - Znalosť súkromného kľúča k verejnému kľúču uvedenom v certifikáte

Diffieho-Hellmanov protokol

- Protokol na dohodnutie kľúča
  1. A $\rightarrow$ B: $p, g, g^x \bmod p$
  2. B $\rightarrow$ A: $g^y \bmod p$
  3. A vypočíta $K = (g^y)^x = g^{xy}$, B vypočíta $K = (g^x)^y = g^{xy}$
- Varianty DH protokolu použité v SSL/TLS (jedna z možností) a inde
- Bezpečnosť
  - pri pasívnom útočníkovi
  - nie je bezpečný pri aktívnom útočníkovi uprostred (MITM)
  - ochrana spočíva v zabezpečení autentickosti správ (napr. dig. podpismi)

Varianty DH protokolu

- TLS
  - DH_anon – anonymný DH (možný MITM útok)
  - DHE_RSA, DHE_DSS – server svoje parametre podpíše
  - DHE_RSA, DHE_DSS – parametre sú súčasťou certifikátu servera
- IPSec
  - Protokoly IKEv1, IKEv2 – dohodnutie kľúča
  - DH protokol – autentizácia šifrovaním, digitálnym podpisom, MAC
- SSH2
  - DH je jedna z metód, server podpisuje svoje parametre

SSL / TLS

- SSL – Secure Socket Layer (pôvodne Netscape)
- TLS – Transport Layer Security (TLS 1.0 ~ SSL v3.1)
- Protokol na transportnej vrstve (nad TCP/IP), zabezpečuje intergitu a dôvernosť
- V podstate ľubovoľný protokol nad SSL (FTP, SMTP)
- Najčastejšie: HTTP/SSL (https)

SSL / TLS

Základne charakteristiky SSL/TLS	
Autentizácia servera	povinná (znalosť súkromného kľúča k verejnému kľúču z certifikátu)
Autentizácia klienta	voliteľná (málokedy používané, obvykle riešené po vytvorení TLS spojenia)
Distribúcia kľúčov	viaceré protokoly (odvodenie kľúčov pre šifrovanie a autentizačné kódy)
Dôvernosť	symetrické šifrovanie (podpora rôznych algoritmov a módov)
Autentickosť	autentizačné kódy (podpora rôznych algoritmov)

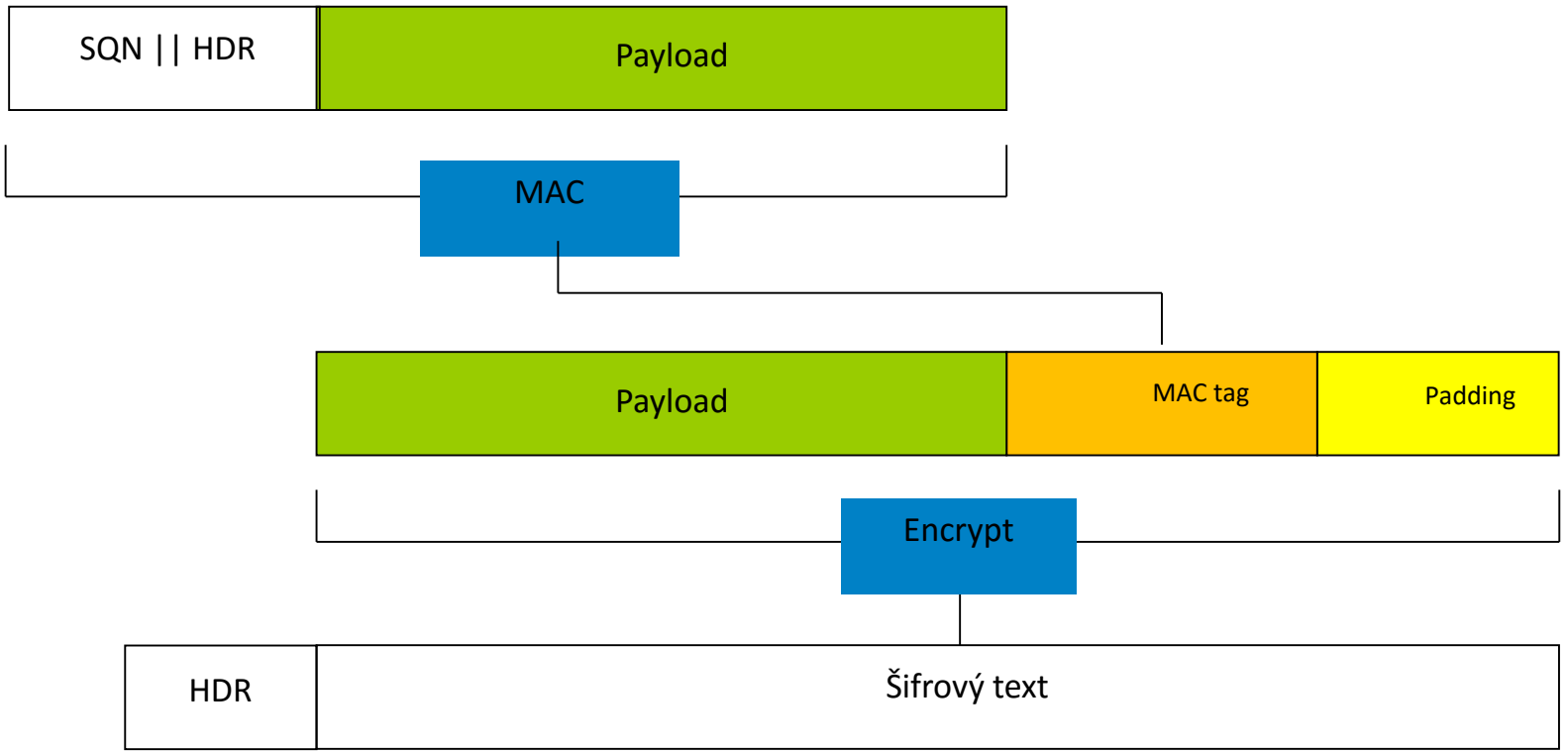
SSL / TLS

- SSL protokoly:
  - Record Protocol – spodná vrstva (šifrovanie, MAC, kompresia)
  - Handshake Protocol – autentizácia (jednostranná - server, alebo vzájomná - aj klient), dohoda o kryptografických algoritmoch, dohoda o šifrovacom kľúči a MAC kľúči
  - Alert Protocol - oznamovanie chybových hlášok (napr. `certificate_expired`)
  - Change Cipher Spec Protocol – „prepnutie“ aloritmov
- kryptografia v SSL, napr.:

_KeyExchange_WITH_Cipher_MAC
SSL_DHE_DSS_WITH_DES_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA

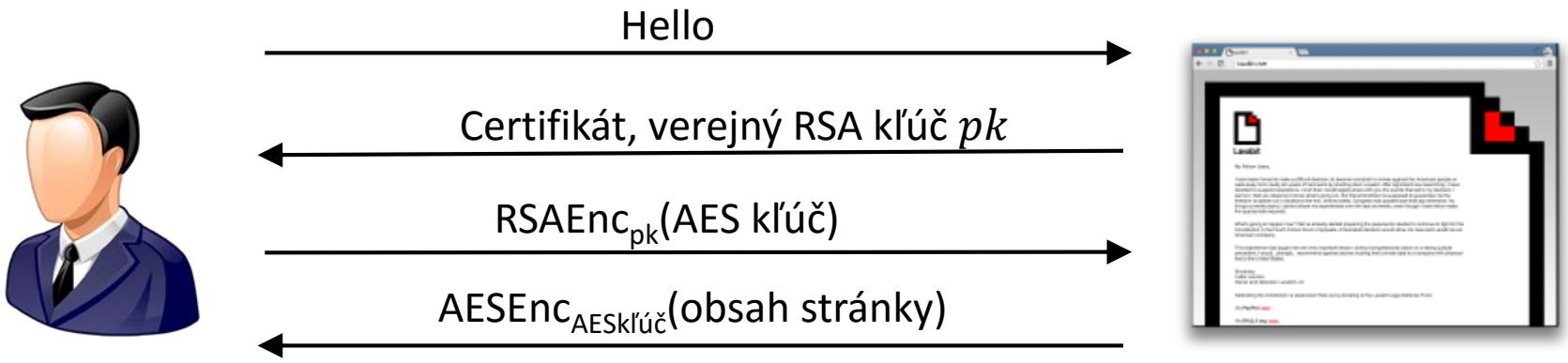
SSL / TLS

TLS Record Protokol: MAC-Encode-Encrypt



TLS RSA výmena kľúčov

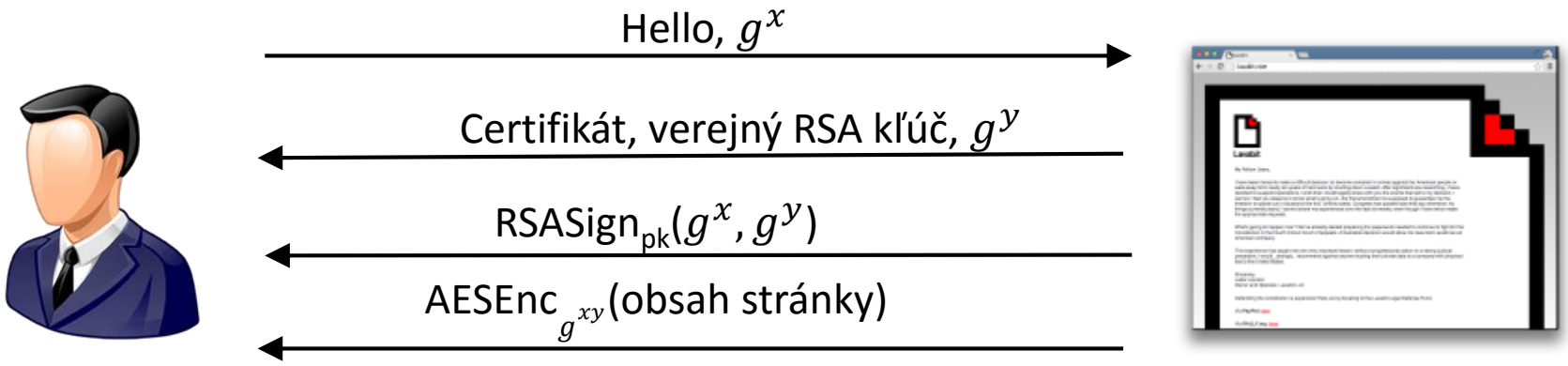
dôležitosť „forward secrecy“



- Útočník poznajúci tajný kľúč servera (Lavabitu) môže
  - Vystupovať ako daný server (Lavabit) pre kohokoľvek
  - Dešifrovať všetkú budúcu aj **minulú** komunikáciu

TLS Diffie-Hellman výmena kľúčov

dôležitosť „forward secrecy“



- Útočník poznajúci tajný kľúč servera (Lavabitu) môže
  - Vystupovať ako daný server (Lavabit) pre kohokoľvek
  - „Forward secrecy”: Nemôže dešifrovať predchádzajúcu komunikáciu (ak g^{xy} bolo zahodené)

IPSec

- Bezpečnostný „doplnok“ k IP vrstve
- Oblasti pôsobnosti: dôvernosť, autentickosť, správa kľúčov
- Výhody „nízkoúrovňového“ protokolu:
  - zabezpečená celá komunikácia nad IP
  - transparentné pre aplikácie
  - možnosť vytvoriť VPN
  - integrácia do sieťových zariadení (smerovače a pod.)
- Nevýhody:
  - SW implementácia (v operačnom systéme) zaťažuje server
  - Identita zariadenia, nie používateľa/aplikácie

IPSec

- základné protokoly – AH, EPS
- AH (Authentication Header) - len autentickosť/integrita
- ESP (Encapsulating Security Payload) - šifrovanie a voliteľne autentickosť/integrita
- Transportný mód (spracúvajú sa vybrané časti IP paketu)
- Tunelovací mód (zabalenie celého IP paketu do nového) protokolov – možnosť vytvoriť VPN
- Algoritmy: HMAC-MD5/SHA-1 (96), 3DES, Blowfish, ...
- Správa kľúčov: manuálna, automatizovaná (ISAKMP/Oakley)

Identifikácia a autentizácia

- Protokol medzi dvoma stranami
  - Dokazovateľ P
  - Overovateľ V
- Dokazovateľ P sa snaží dokázať svoju identitu overovateľovi V.
- Identifikácia: predstavenie identity P overovateľovi V
- Autentizácia: dôkaz / potvrdenie identity P
- Výstup protokolu:
  - akceptácia, t.j. identita P je pravá, komunikácia pokračuje
  - zamietnutie, ukončenie komunikácie
  - v niektorých prípadoch aj tzv. „session key“ – dočasný kľúč na šifrovanie danej relácie

Identifikácia a autentizácia - ciele

- **Korektnosť:**
  - V prípade poctivých strán P, V: V akceptuje identitu P
- **(Ne)prenositeľnosť:**
  - V nemôže zneužiť komunikáciu a vydávať za P pre tretiu stranu C
- **(Ne)falšovateľnosť:**
  - Žiadna tretia strana C sa nemôže vydávať za P pre V.
- **Robustnosť:**
  - Predchádzajúce vlastnosti zostávajú v platnosti aj v prípade veľkého množstva vykonaní protokolu.
- **“Real-time”:**
  - Autentizácia sa musí uskutočniť v realnom čase.

Autentizácia – základ

Autentizácia môže prebiehať na základe:

1. **Toho čo viem –** heslo, PIN, tajný kľúč
2. **Toho čo mám –** pas, kreditná karta, smart karta, token, mobil,…
3. **Toho čo som –** moje fyzikálne charakteristiky: odtlačok prsta, podpis, vzor dúhovky, hlas...

Využitie I & A

- Primárne využitie:
  - (Kontrolovaný) prístup k zdrojom
  - Logovanie / monitoring používateľov (kto čo robí)
  - Účtovanie (kto čo využíva)
- Ďalšie využitie:
  - Napr. vytvorenie session kľúča

Vlastnosti protokolov na I & A

- Reciprocita
  - jednostranná vs. vzájomná autentizácia
- Efektívnosť
  - Výpočtová náročnosť (# operácii)
  - Komunikačná zložitosť (# správ, prenesené bity)
- Zapojenie tretej strany
  - Dôveryhodná vs. nedôveryhodná 3. strana
  - Online vs. offline
- Bezpečnostné vlastnosti
  - Spôsob ukladania tajných hesiel / kľúčov
  - Dokázateľná bezpečnosť, Zero-knowledge

Heslá

- Poskytujú tzv. slabú autentizáciu
- Zdieľané tajomstvo medzi používateľom a systémom
  - UserID používateľa identifikuje, heslo slúži ako dôkaz identity
- Dôsledky:
  - Systém musí mať uložené heslá (v nejakej forme)
  - Používateľ musí systému ukázať svoje heslo (cez nejaký komunikačný kanál)

Uloženie hesiel

Súbor s heslami v otvorenom tvare:

- Bez akejkoľvek ochrany súboru
  - Zjavne nebezpečné - ktokoľvek môže získať heslo
- Read / write ochrana v operačnom systéme
  - Administrátor / root ma prístup ku všetkým heslám
  - Backup súboru nemusí byť chránený

Uloženie hesiel

- Zle:
  - Šifrované
  - Odtlačok s jednoduchou aplikáciou hašovacej funkcie
- Dobre: ireverzibilne + soľ + iterácie
- Soľ – (náhodný) individuálny reťazec
  - Pridávaná pri výpočte odtlačku
  - Znemožňuje útočníkovi predvýpočty, paralelné prehľadávanie rovnakých hesiel (vedú k rôznym odtlačkom)
- Iterácie – spomalenie výpočtu odtlačku
  - Spomalenie overenia hesla (nevadí), spomalenie útoku (vyhovuje)
- Vhodné algoritmy: PBKDF2, bcrypt, scrypt
- Zlé heslo je zlé bez ohľadu na uloženie (slovníkový útok)

Posielanie hesiel

- Používateľ pošle heslo v otvorenom tvare
  - Systém ho zahašuje (prípadne pridá soľ) a porovná s uloženým záznamom
    - Administrátor nevie získať žiadne heslo (ak neodpočúva komunikáciu)
    - Backup obsahuje iba haš hesiel
    - Heslo je možné odpočuť
- Alternatívne, používateľ heslo zahašuje a haš pošle na server
  - Útočník nevie odpočuť heslo
  - Soľ (ak sa využíva) musí byť uložená aj na klientovi
- V obidvoch prípadoch však útočník môže zopakovať odpočutú správu

Príklad autentizácie na základe hesla

Používateľ Adam
(prihlasovacie meno, heslo)

Prihlasovacie meno, heslo

Počítač / Server

Vypočítaj
H(heslo)

$meno_1$	$H(heslo_1)$
$meno_2$	$H(heslo_2)$
$meno_3$	$H(heslo_3)$
⋮	⋮
$meno_N$	$H(heslo_N)$

Eva sa snaží uhádnuť Adamove heslo

Heslá: útoky

- Opakovaním
  - Ak je možné odpočúvať komunikáciu
- Úplné preberanie
  - Útočník skúša všetky možné heslá
  - Ochrana: zvýšiť veľkosť hesiel a / alebo limitovať počet (online) pokusov
  - Offline útok:
    - útočník môže generované heslá porovnávať priamo so súborom (ak má prístup k súboru / databáze)

Heslá: útoky

- Slovníkový útok
  - Väčšina používateľov si volí heslá z malej podmnožiny všetkých hesiel
  - Útočník skúša iba heslá zo slovníka – aj najväčší slovník má iba 250 000 slov, čo je menej ako 26^4.
  - Existujú aj špeciálne slovníky na „heslá“
  - Využiteľné najmä pri offline útokoch
    - V súčastnosti na to existujú šikovné programy – heslo odhalia v priebehu niekoľkých minút až hodín

Heslá: útoky


THE TOP 20 PASSWORDS OF ALL TIME
1 123456
2 12345
3 123456789
4 Password
5 iloveyou
6 princess
7 rockyou
8 1234567
9 12345678
10 abc123
11 Nicole
12 Daniel
13 babygirl
14 monkey
15 Jessica
16 Lovely
17 michael
18 Ashley
19 654321
20 Qwerty

Náhodnosť používateľských hesiel

Dĺžka hesla	PIN (10 znaková abeceda)	Všeobecné heslá (94 znaková abeceda)
4	9	10
8	13	18
10	15	21
16	21	30
22	27	38

- Príklad: 2012, LinkedIn, 6,5 mil. používateľských účtov
- 4 hodiny + slovníkový útok → cca. 900 tisíc hesiel
- Pokračovanie slovníkového útoku → cca. 2 mil. hesiel

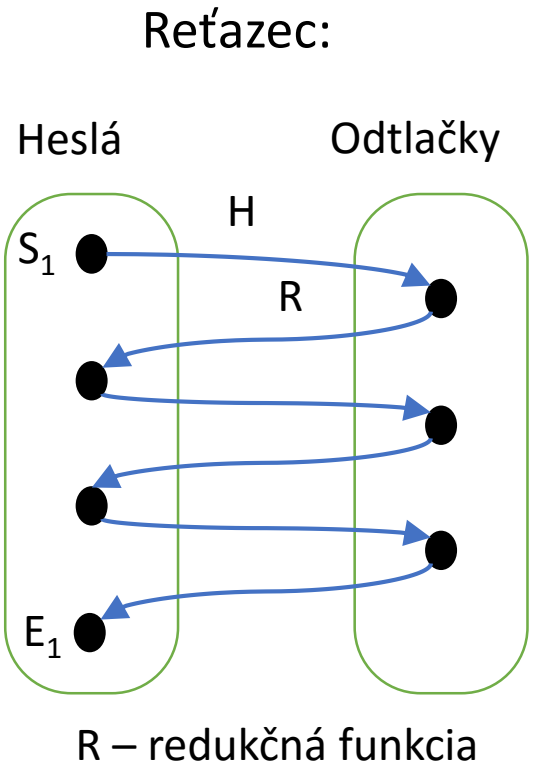
Hľadanie hesla – úplné preberanie

- Online úplné preberanie
  - skúšame všetky možné heslá, vypočítame ich haš
  - Výpočtová zložitosť: **N := |A|** (A množina všetkých hesiel)
  - Pamäť: **0**
  - Predvýpočet: **0**
- Úplné preberanie s predvýpočtom
  - Predvýpočítame si tabuľku všetkých možných hesiel a ich hašov
  - Online výpočtová zložitosť: **0**
  - Pamäť: **N**
  - Predvýpočet: **N**

Time-memory trade-off

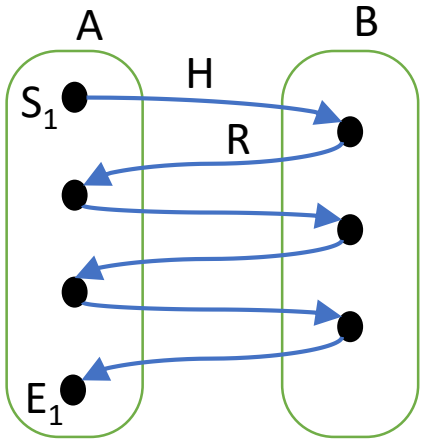
Hellmanove tabuľky, 1980

- Namiesto ukladania celej množiny možných hesiel a ich hašov počítame tzv. reťazce hašov
- Heslá sú organizované v reťazcoch hašov, iba prvý a posledný prvok reťazca je zapamätaný v tabuľke
- Pri predvýpočte vytvoríme m reťazcov dĺžky t



R – redukčná funkcia

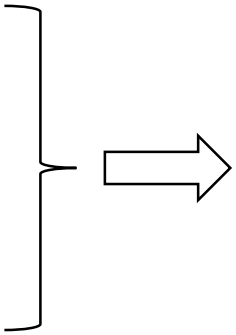
Predvýpočet, reťazce



- $H: A \to B$ – hašovacia funkcia (MD5, SHA1)
- $R: B \to A$ – „ľubovoľná“ funkcia (redukcia)
- $f: A \to A$, kde $f := R \circ H$

Reťazce:

$$S_1 = x_{1,1} \xrightarrow{f} x_{1,2} \xrightarrow{f} x_{1,3} \xrightarrow{f} \dots \xrightarrow{f} x_{1,t} = E_1$$
$$S_2 = x_{2,1} \xrightarrow{f} x_{2,2} \xrightarrow{f} x_{2,3} \xrightarrow{f} \dots \xrightarrow{f} x_{2,t} = E_2$$
$$\vdots$$
$$S_m = x_{m,1} \xrightarrow{f} x_{m,2} \xrightarrow{f} x_{m,3} \xrightarrow{f} \dots \xrightarrow{f} x_{m,t} = E_m$$

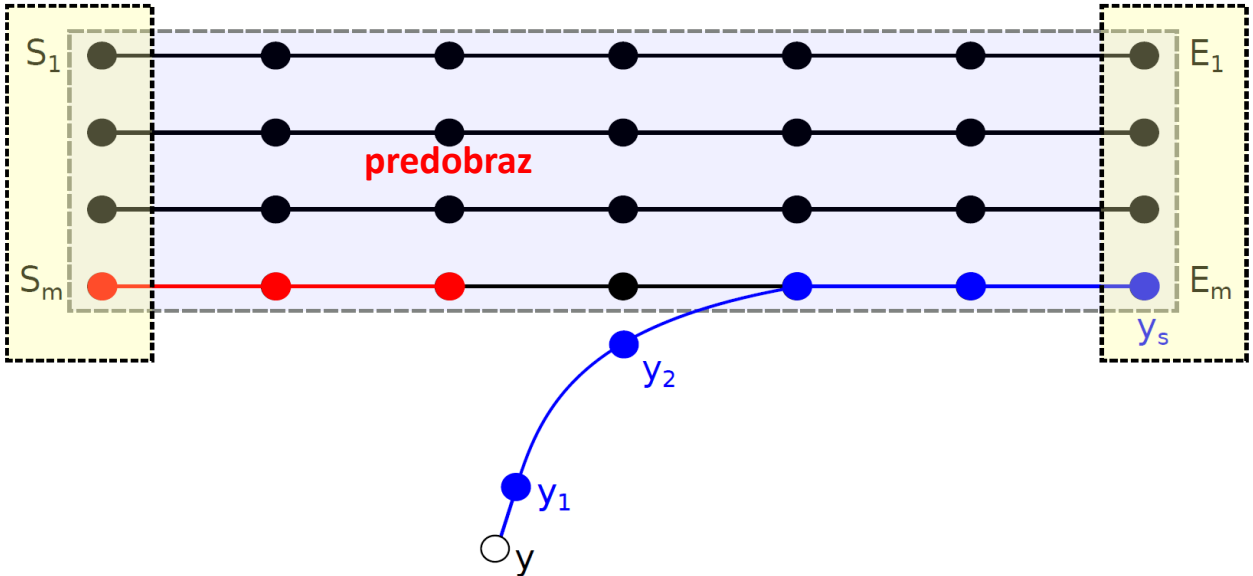


S_1	E_1
S_2	E_2
S_3	E_3
⋮	⋮
S_m	E_m

Online fáza

Pre daný haš $y \in B$, vypočítame $y_1 = R(y)$ a postupne generujeme $y_i = f(y_{i-1})$, pre $i = 2, \ldots, t$.

- Pre každé y_i kontrolujeme, či to nie je nejaké E_j (t.j. či sa nenachádza v tabuľke),
- Ak sme našli E_j, **môžeme** predobraz y získať postupným hašovaním z S_j



TMTO: Hellmanove tabuľky

- N – veľkosť množiny hesiel
- Predvýpočet: $P = t^2 m = N$
- Online zložitosť: $T = t^2$
- Pamäť: $M = tm$

- Ak chceme minimalizovať $T + M$, potom optimálna voľba je

$$T = M = N^{2/3}$$

TMTO: Dúhové tabuľky

Optimalizácia Hellmanových tabuliek

- N – veľkosť množiny hesiel
- Predvýpočet: $P = N$
- Online zložitosť: $T = \frac{t(t-1)}{2}$
- Pamäť: $M = tm$
- Výhody oproti Hellmanovým tabuľkám
  - Polovičná zložitosť online fázy $t(t-1)/2$ vs t^2

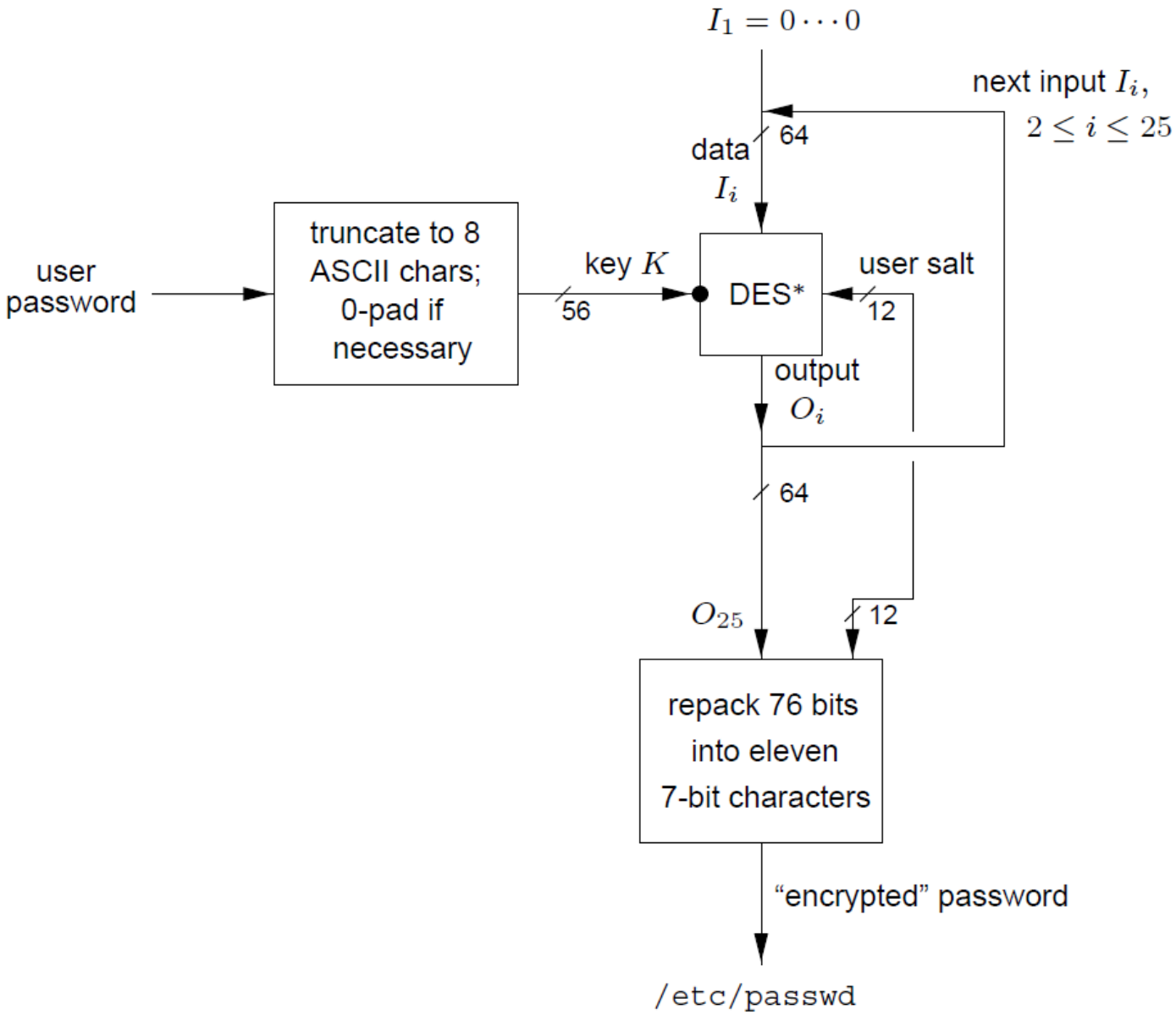
Time-memory trade-off útoky

- TMTO útok nie je nikdy celkovo rýchlejší ako brute-force
- TMTO má význam v nasledovných prípadoch
  - Útok sa opakuje niekoľko krát
  - „Útok počas obednej prestávky“
  - Útočník nie je veľmi výkonný, avšak má možnosť stiahnuť si tabuľky
- Podmienky, aby bolo možné TMTO použiť
  - Problém rozumnej veľkosti
  - Jednosmerná funkcia (alebo CPA útok na šifrovom texte)
- Použitie soli pri ukladaní hesiel zabraňuje TMTO útokom

Heslá: ochrana voči útokom

- Kontrola sily hesla
  - Zabrániť používateľom zvoliť si slovníkové heslo.
  - max. doba platnosti hesla, min. doba platnosti hesla
- Použiť pomalú hašovaciu funkciu
  - Napr. iterovať štandardnú hašovaciu funkciu niekoľko krát
- Pridanie náhodnej soli
  - Pred zahašovaním hesla P k nemu pridáme náhodnú soľ S
  - C = h(S, P), zapamätáme si S, C
  - Dve rovnaké heslá majú rôznu soľ, t.j. rôzne šifrovanie
  - Zvačší sa zložiťosť slovníkového útoku (ale nie pre daného používateľa)
- Frázové heslá
- Expirácia hesiel
- Blokovanie prístupu po x neúspešných prihláseniach
- Spomaľovanie odozvy po neúspešnom prihásení, atď.

Heslá v UNIXe



Manažment hesiel

- Ako identifikovať používateľa ak ešte nemá heslo?
  - Ako ste dostali Vaše heslo pri nástupe na FMFI?
- Zabudnuté heslá
  - Zaslanie hesla nesprávnej osobne
  - Neposkytovať zabudnuté heslo volajúcemu, ale zavolajte naspäť na overené číslo používateľa
- Phishing
  - Dostanete email od banky vyžadujúci zmenu hesla

Manažment hesiel

- K obmedzeniam hesiel treba pristupovať rozumne
  - Ak heslo musí byť príliš zložité
    - používateľ si ho zapíše
  - Ak si heslo musí používateľ často meniť
    - zvolí si jednoduchšie heslo
  - Veľa systémov vyžadujúcich heslo
    - Koľko máte rôznych hesiel?
- Treba nájsť rovnováhu medzi bezpečnosťou a prívetivosťou pre používateľov

PIN

PIN – Personal Identification Number

- Používané spolu s nejakým tokenom, smart-kartou, kreditkou a pod.
- PIN je malý, zvyčajne 4-8 číslic
  - Môže a nemusí byť uložený v tokene (online vs. offline)
  - Môže byť odvoditeľný (hašovaním) z tajného kľúča a identity uloženej v tokene
  - Token obsahuje údaje na identifikáciu, PIN slúži na overenie vlastníctva tokenu – **dvojstupňová autentizácia**
- Na zamedzenie online útoku preberaním sa limituje počet nesprávnych pokusov.

Passkey

Password derived key

- Z PINu / hesla sa pomocou jednosmernej funkcie vygeneruje kľúč
- Kľúč je následne použítý na zabezpečenie komunikácie
- Overovateľ pozná PIN / heslo, môže si teda vygenerovať kľúč
- Možné skombinovať heslo so soľou – zakaždým nový kľúč
- Podobné slabiny ako v prípade fixných hesiel
  - Nutnosť pamätať si heslá na serveri

Jednorázové heslá

- Snaha o elimináciu útoku opakovaním
- Zdieľaný zoznam hesiel
  - Každý prvok použitý iba raz
  - Variácia: Tabuľka challenge-response dvojíc
    - Overovateľ pošle challenge, používateľ odpovie príslušným párom z tabuľky
- Sekvenčne aktualizované heslá
  - Začíname so zdieľaným heslom
  - Pri autentizácii s použitím hesla *i,* používateľ pošle nové heslo *i+1,* zašifrované heslom *i*

Jednorázové heslá
Lamportova schéma

- Sekvencie hesiel s využítím jednosmernej funkcie: **Lamportova schéma**
- **Set-up:**
  - Dokazovateľ *P* má tajné heslo *w*. H je hašovacia funkcia
  - Určíme konštantu *t* – počet možných autentizácii
    - Po *t* autentizáciách je potrebné znovu vygenerovať *w*
  - *P* pošle *V* cez **autentický kanál** $w_0 = H^t(w)$
    - *V* inicializuje počítadlo pre *P,* napr. $l_P = 1$
    - H^t znamená t iterácii H, t.j. $H \circ H \circ \cdots \circ H$

Jednorázové heslá
Lamportova schéma

i-ta iterácia Lamportovej schémy

- P vypočíta $w_i = H^{t-i}(w)$ a pošle to *V*
- V overí, či platí
  - $i = i_A$
  - $H(u) = w_{i-1}$, kde u je prijatá správa od *A*
- Ak je overenie úspešné
  - V akceptuje heslo, zvýší i_P o 1
  - V uloží *u* ako w_i

Jednorázové heslá

Lamportova schéma

- Útok opakovaním nie je možný, avšak
  - Schéma je zraniteľná v prípade ak útočník získa *w* pred uskutočnením protokolu
    - Potrebujeme zabezpečiť autentický prenos $H^t(w)$
  - Problémy robia straty spojenia
- Výhoda
  - Malé komunikačné nároky
- Alternatívna schéma (vyžaduje si uloženie hesla na serveri)
  - P pošle serveru dvojicu *(r, H(r, p))*, kde *r* je zakaždým iné (napr. poradové číslo), *p* je zdieľané heslo

Challenge-response autentizácia

- Tzv. silná autentizácia
- Dokazovateľ dokáže znalosť nejakého tajomstva cez challenge-response protokol
  - Bez toho, aby tajomstvo počas behu protokolu odhalil (v niektorých prípadoch ho však overovateľ pozná)
- Dokazovateľ odpovedá na časovo závislý „challenge“
- Môže využívať
  - Symetrické šifrovanie
  - Asymetrické šifrovanie

Časovo závislé parametre

- Zamedzujú útokom opakovaním
- 3 základné typy:
  - Náhodné hodnoty
  - Sekvenčné číslovanie
  - Časové pečiatky
- “New and once” - nonce
  - Hodnota parametra musí byť zakaždým iná
  - Je potrebné zabezpečiť integritu a autentickosť parametrov – naviazať ich na ostatné posielané správy

Náhodné hodnoty

- Overovateľ V vygeneruje náhodnú hodnotu *r*
  - Pošle ju P ako „challenge“
- P odpovie správou, ktorá je „zviazaná“ s *r*
  - „zviazanosť“ s *r* zabezpečuje čerstvosť
- Problémy:
  - Opakovanie hodnoty *r* (narodeninový paradox)
  - Predvídateľnosť *r* – generovanie náhodných čísel nie je jednoduché
  - Komunikačná zložitosť
    - oproti časovým pečiatkam a sekvenčným číslam sa vyžaduje jedna správa naviac

Sekvenčné číslovanie

- Číslovanie správ poslaných medzi P a V
  - Monotónne rastúce číslovanie
- Problémy:
  - Potreba dlhodobo uchovávať aktuálne poradové číslo správy
  - Synchronizácia
  - Potreba riešiť výpadky spojenia a pod.
  - Nemožnosť detekovať „forced delay“ útok

Časové pečiatky

- Do každej posielanej správy zakomponujeme časovú pečiatku
  - Akceptujeme len správy s časovou pečiatkou, ktorá je v rámci nejakého akceptovateľného časového okna
  - Môžu sa využívať aj na časové obmedzenie prístupu
  - Umožňujú detekciu „forced delay“ útokov
- Nevýhody
  - Nutná synchronizácia hodín
    - Ak je synchronizácia vykonaná po sieti, je potrebné komunikáciu zabezpečiť – zase s využitím časových pečiatok?
  - Potreba ukladať prijaté časové pečiatky v rámci daného časového okna
    - Aby sme vedeli zabrániť útokom opakovaním
  - Čas sa stáva kritickým prvkom systému

Challenge-response autentizácia

s využitím symetrického šifrovania

- Obe strany A,B zdieľajú nejaký tajný kľúč k
- Základné (jednoduché) protokoly ISO/IEC 9798-2:
  - S využitím časových pečiatok:
    - $A \rightarrow B : E_k(t_A, B)$
    - Po prijatí, B správu dešifruje a overí časovú pečiatku
    - Posielanie identity druhej strany zamedzuje použitia rovnakej správy na autentizáciu B do A
  - S využitím náhodných čísel
    - $B \rightarrow A : r_B$
    - $A \rightarrow B : E_k(r_B, B)$
    - Po prijatí, B správu dešifruje a overí r_B (nemalo by sa opakovať).

Challenge-response autentizácia

s využitím symetrického šifrovania

- Vzájomná autentizácia
  - B → A : r_B
  - A → B : $E_k(r_A, r_B, B)$
    - Po prijatí B správu dešifruje a skontroluje r_B
  - B → A : $E_k(r_B, r_A)$
    - Po prijatí A správu dešifruje a skontroluje r_A
- S využitím hašovacích funkcií (ISO/IEC 9798-4):
  - B → A : r_B
  - A → B : $r_A, h_k(r_A, r_B, B)$
    - Po prijatí B zahašuje r_A, r_B, B a porovná s prijatou správou
  - B → A : $h_k(r_B, r_A)$
    - Po prijatí A zahašuje r_A, r_B a porovná s prijatou správou

Challenge-response autentizácia
s využitím **asymetrických** techník

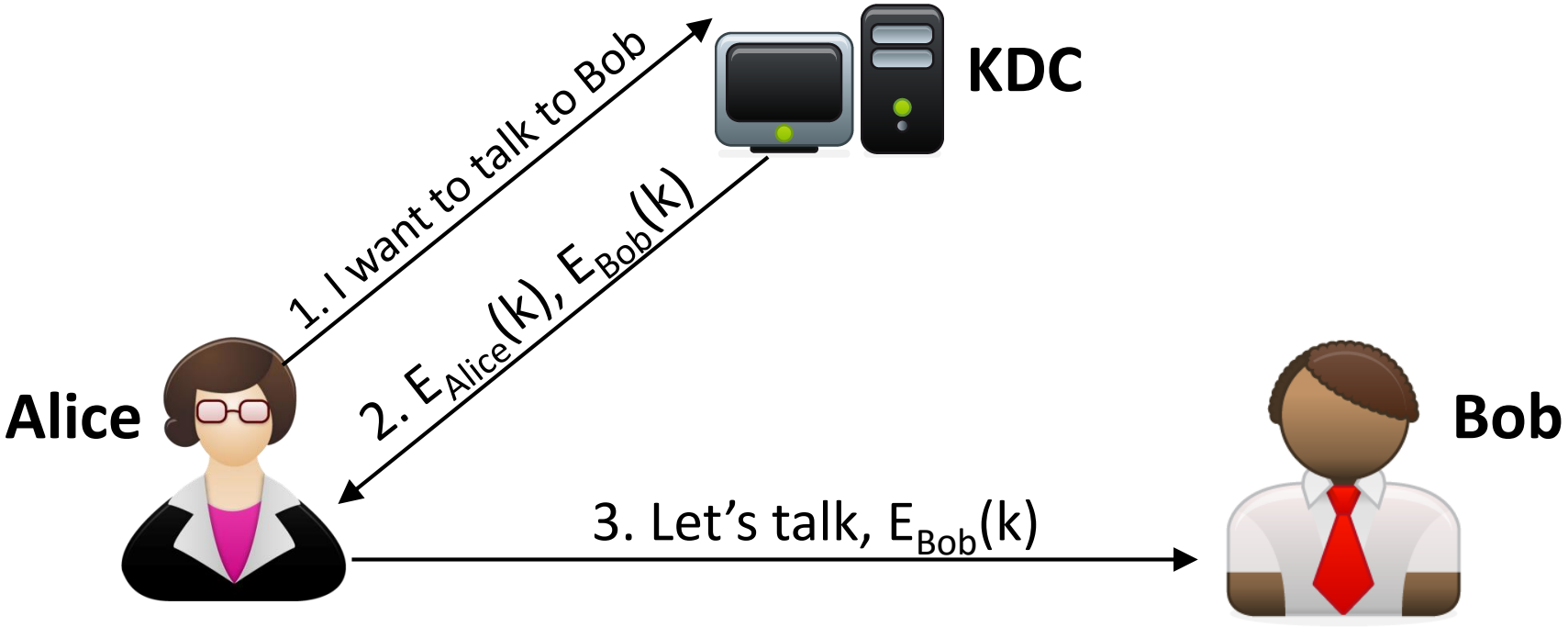
- Vzájomná autentizácia, asymetrické šifrovanie
  - B → A : $Pub_A(r_B, A)$
    - Po prijatí, A dešifruje a získa r_B
  - A → B : $Pub_B(r_A, r_B)$
    - Po prijatí, B dešifruje, získa r_A, r_B a porovná r_B
  - B → A : r_A

Základné challenge-response protokoly

- Všetky uvedené protokoly
  - Sú dvojstranné protokoly, t.j. bez tretej strany
  - Dokazovateľ aj overovateľ si dôverujú
  - Predpokladajú distribúciu kľúčov medzi komunikujúcimi stranami
    - T.j. strany sa navzájom poznajú, zdieľajú tajný kľúč / poznajú verejný kľúč druhej strany
    - Problematické v prípade veľkého množstva komunikujúcich párov
- Ak posledný predpoklad nie je platný
  - Je potrebné využiť 3. stranu na výmenu / distribúciu kľúčov – napr. KDC (Key Distribution Center)

Key distribution center

- Server na distribúciu kľúčov
- Každý používateľ zdieľa so serverom tajný kľúč



Needhamov-Schroederov protokol

- Autentizácia s využitím 3. strany
  1. A → S: A, B, r_A
  2. S → A: $E_{K_{AS}} (r_A, B, K_{AB}, E_{K_{BS}}(K_{AB}, A))$
     - A dešifruje. Overí aktuálnosť r_A. Získa kľúč K_{AB}
  3. A → B: $E_{K_{BS}} (K_{AB}, A)$
     - B dešifruje, získa kľúč K_{AB}
  4. B → A: $E_{K_{AB}}(r_B)$
     - A dešifruje a získa r_B.
  5. A → B: $E_{K_{AB}}(r_B-1)$
     - B dešifruje, overí aktuálnosť cez r_B-1

Needhamov-Schroederov protokol

- Úloha servera S
  - Distribúcia kľúčov
  - Nemusí byť plne online, keďže po vykonaní protokolu, A ani B už nepotrebujú S
- Cvičenie: modifikuje protokol tak, aby využíval asymetrické šifrovanie
  - Aké to bude mať výhody?
- Needhamov-Schroederov protokol sa v súčasnosti neodporúča používať
  - Alternatíva: Kerberos protokol – veľmi rozšírený

Útoky na autentizačné protokoly

Needhamov-Schroederov protokol

- Predpokladajme, že kľúč K_{AB} bol kompromitovaný
  1. A → S: A, B, r_A
  2. S → A: $E_{K_{AS}}$ (r_A, B, K_{AB}, $E_{K_{BS}}$(K_{AB}, A))
  3. Z(A) → B: $E_{K_{BS}}$ (K_{AB}, A)
  4. B → Z(A): $E_{K_{AB}}$(r_B)
  5. Z(A) → B: $E_{K_{AB}}$(r_B-1)
- Z sa úspešne autentizovalo ako A
- Medzi krokmi 2 a 3 nie je žiadne previazanie pomocou časovo-závislého parametra

Útoky na autentizačné protokoly

- Paralelný beh protokolov
  - Odpočuté / prijaté správy v jednom protokole využijeme pri paralelnom behu druhého protokolu
- Útok zrkadlením
  - Špeciálny prípad predošlého útoku
  - Príklad: 2 paralelné šachové partie – raz som čierny, raz biely
- “Chosen text” útok
  - Útočník si volí hodnoty parametrov tak, aby jednoduchšie odhalil informácie o tajnom kľúči
  - CPA / CCA útok na šifrovaciu schému
- „Forced delay“ útok
  - Útočník odpočuje správu (zvyčajne obsahujúcu sekvenčné číslo) a použije ju neskôr (po prípadnom dešifrovaní hrubou silou)

Protokoly na I & A
zhrnutie

- Cieľ: Dokázať (vzájomnú) identitu
  - Počas behu protokolu, dokazovateľ by nemal odhaliť svoje tajomstvo útočníkovi
- Fixné heslá
  - Ak je heslo posielané v otvorenom tvare, útočník ho odpočuje
  - Ak je heslo posielané šifrovane, útočník ho môže zopakovať
- Challenge-response protokoly
  - Zabraňujú útokom opakovaním s využitím časovo závislých parametrov
  - Útočník však môže získať nejakú informáciu o tajomstve
    - „Chosen-text“ útoky,
  - Overovateľ môže poznať tajomstvo

Zero Knowledge protokoly

- Dokazovateľ dokáže overovateľovi znalosť tajomstva bez toho, aby odhalil akúkoľvek informáciu o tajomstve
- Postavené na interaktívnych dôkazoch:
  - Pravdepodobnostná verzia „dôkazu“
  - Úlohou dokazovateľa je presvedčiť overovateľa o pravdivosti nejakého tvrdenia cez výmenu správ
- Interaktívne dôkazy na autentizáciu
  - Dôkaz znalosti nejakého tajomstva na základe odpovedania na otázky, pričom správne odpovede vyžadujú znalosť tohto tajomstva

Interaktívne dôkazy

- Úplnosť: interaktívny dôkaz je úplný:
  - Ak sú obidve strany čestné, dôkaz (protokol) skončí úspešne s veľkou pravdepodobnosťou
- Korektnosť: interaktívny dôkaz je korektný, ak existuje efektívny algoritmus M, ktorý
  - Ak je útočník schopný úspešne prebehnúť protokol (presvedčiť overovateľa),
  - potom M môže byť použité na extrakciu informácie z daného útočníka, ktorá môže byť použitá na ďalšie úspešné absolvovanie protokolu
    - Inak povedané, útočník pozná to tajomstvo

Zero-knowledge protokoly

- Protokol ma vlastnosť „zero-knowledge“ ak
  - Existuje efektívny algoritmus - „simulátor“ S, ktorý
    - dostane na vstupe tvrdenie, ktoré má dokázať
    - bez interakcie s dokazovateľom je schopný generovať transcript neodlíšiteľný od skutočného behu protokolu
- T.j. dokazovateľ neodhalí žiadnu informáciu o svojom tajomstve, okrem tej, ktorá je vypočítateľná z verejne dostupných údajov
  - Aj keď komunikuje s nečestným overovateľom

Zero-knowledge protokoly

V porovnaní s ostatnými protokolmi na I & A:

- Dlhodobé opakovanie protokolu neznižuje bezpečnosť
  - Odolnosť vočí „chosen-text“ útokom
- Nevyžadujú šifrovanie (politické dôvody)
- Zvyčajne menej efektívne
  - Väčšia komunikačná aj výpočtová náročnosť
- Postavené na nedokázaných predpokladoch
  - Podobne ako v prípade asymetrických techník, napr. problém faktorizácie
- „Asymptotické“ dôkazy ZK vlastnosti

Fiatov-Shamirov protokol

- Postavený na probléme počítania odmocnín modulo veľké n = p.q
  - Ekvivaletné problému faktorizácie
- Setup:
  - Dôveryhodný server T vyberie *n = p.q*, prvočísla *p* aj *q* ostávajú utajené.
  - Každý dokazovateľ A si vyberie tajomstvo *s = 1…n-1,* ktoré je nesúdeliteľné s *n*
    - Vypočíta $v = s^2 \pmod n$
    - *v* je verejný kľúč, A ho registruje na serveri T

Fiatov-Shamirov protokol

- Komunikácia počas behu protokolu (A dokazovateľ, B overovateľ)
  - A → B: $x = r^2 \bmod n$, kde r je náhodné $1 \leq r \leq n-1$
  - B → A: náhodný bit e
  - A → B: $y = r * s^e \pmod n$
- B zamietne, ak y = 0, inak
  - akceptuje, ak $y^2 = x.v^e \pmod n$

Fiatov-Shamirov protokol

- Útočník C si môže na začiatku tipnúť e:
  - Pozorovanie (C tipuje e=1):
    - C môže zvoliť $x = r^2/v \pmod n$, vtedy dokáže odpovedať správne pre e=1
    - Pre e = 0 musí poznať odmocinu z $x=r^2/v$
    - Pravdepodobnosť úspechu ½
  - Pozorovanie (C tipuje e=0):
    - Útočník C nepozná s, pravdepodobnosť, že odpovie správne je ½ (keďže nevie spočítať *s* z *v*)
- t-násobným opakovaním protokolu dosiahneme pravdepodobnosť podvádzania 2^{-t}

Fiatov-Shamirov protokol

- Odhalená informácia o tajnom kľúči
  - $y = r \pmod n$ – žiadna informácia o s
  - $y = rs \pmod n$ – žiadna informácia o s, keďže r je náhodné a neznáme pre B resp. útočníka
    - Inak by vedeli počítať odmocniny

Identifikácia a autentizácia: Záver

Rôzne autentizačné schémy

- Heslá
  - Jednoduchý útok opakovaním
  - Slovníkový útok
- Jednoduché challenge-response protokoly
  - Poskytujú ochranu voči útokom opakovaním
  - Vyžadujú zdieľané tajomstvo, resp. dôveryhodnú distribúciu verejných kľúčov
- Key distribution center
  - Využitie dôveryhodnej 3. strany na distribúciu kľúča
  - Užitočné v prípade veľkého množstva komunikujúcich strán
- Zero-knowledge protokoly
  - Neposkytujú žiadnu informáciu o tajomstve
  - Nie je potrebné pamätať si tajné kľúče na serveri

Zdieľanie tajomstva

Shamirova schéma

Zdieľanie tajomstva

motivácia

- Svedomitý bankový manažér
  - Má pod sebou 8 zamestnancov
  - Chce, aby mohol byť trezor otvoreny iba keď je prítomných aspoň polovica zamestnancov
- Ako to urobiť?

Zdieľanie tajomstva

- Máme tajomstvo S
- n ľuďom chceme rozdeliť podiely tak, aby
  - Ak poznáme aspoň t z n podielov, vieme rekonštruovať tajomstvo S
  - Ak poznáme najviac t−1 podielov, nevieme o tajomstve nič povedať
- (t, n)-prahová schéma na zdieľanie tajomstva

→ Každý podiel musí byť aspoň tak dlhý ako tajomstvo S

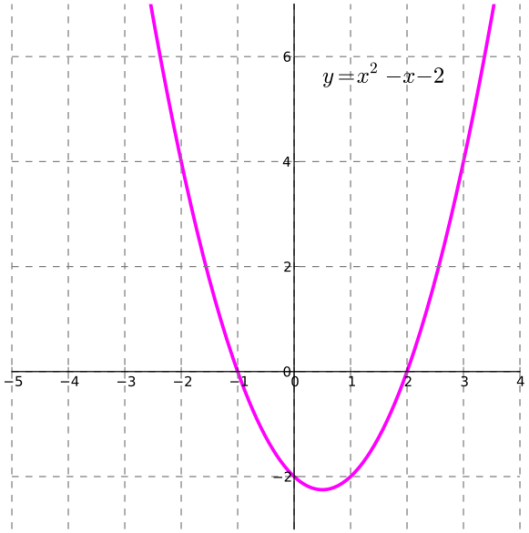
- Z $t - 1$ podielov nevieme nič o S => posledny t-ty podiel musí obsahovať toľko informácie ako samotné tajomstvo S

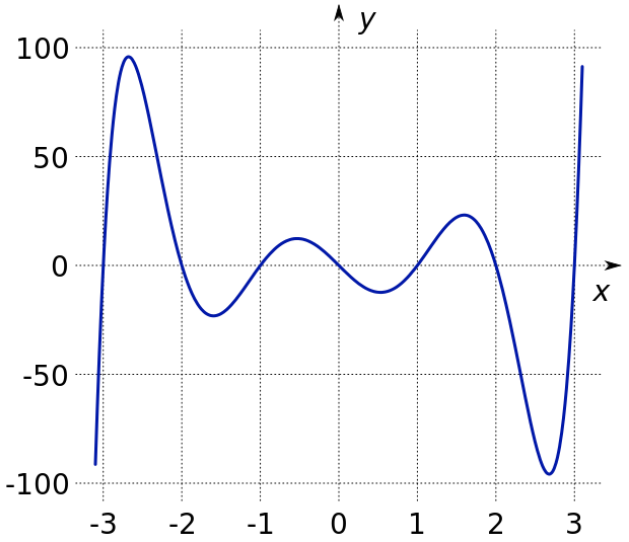
→ Všetky schémy na zdieľanie tajomstva využívajú náhodnosť

- Distribúcia 1-bitového tajomstva S medzi t ľudí
- $t - 1$ podielov nesmie nič prezradiť o bite S => podiely musia byť „náhodné“

Shamirova schema - idea

- Polynóm f stupňa n môžeme popísať $n + 1$ bodmi
  - T.j. $n + 1$ dvojicami $(x, f(x))$





Shamirova schema (t, n) schéma

Inicializácia a rozdelenie tajomstva (vykonáva dôveryhodná autorita):

1. Zvolíme prvočíslo $p \geq n + 1$ a tajnú informáciu $S \in \mathbb{Z}_p$
2. Zvolíme náhodný polynóm $f(x)$ stupňa najviac $t - 1$, tak aby $f(0) = S$

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + \mathrm{a}_0,$$

kde $a_{t-1}, \ldots, a_1 \in_R \mathbb{Z}_p$ a $a_0 = S$

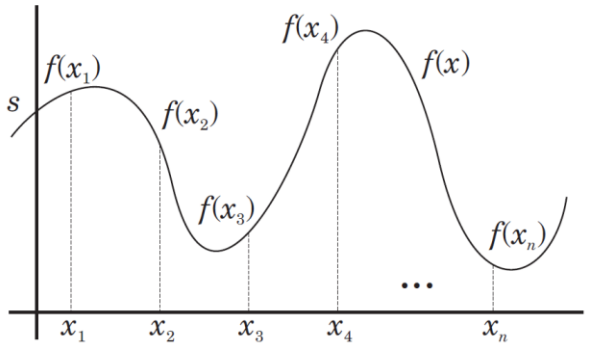
3. Účastník P_i dostane podiel $f(i)$, pre $i = 1, \ldots, n$

Shamirova schema (t, n) schéma

Rekonštrukcia tajomstva S

- t účastníkov má k dispozícii

$$f(x_1) = a_{t-1}x_1^{t-1} + \cdots + a_1 x_1 + \mathrm{a}_0,$$
$$f(x_2) = a_{t-1}x_2^{t-1} + \cdots + a_1 \mathrm{x}_2 + \mathrm{a}_0,$$
$$\vdots$$
$$f(x_t) = a_{t-1}x_t^{t-1} + \cdots + a_1 x_t + \mathrm{a}_0,$$



- Sústava t lineárnych rovníc o t neznámych má práve jedno riešenie
- Vypočítame $a_{t-1}, \ldots, a_0$ a následne rekonštruujeme $S = f(0)$

Shamirova schema (t, n) schéma

bezpečnosť

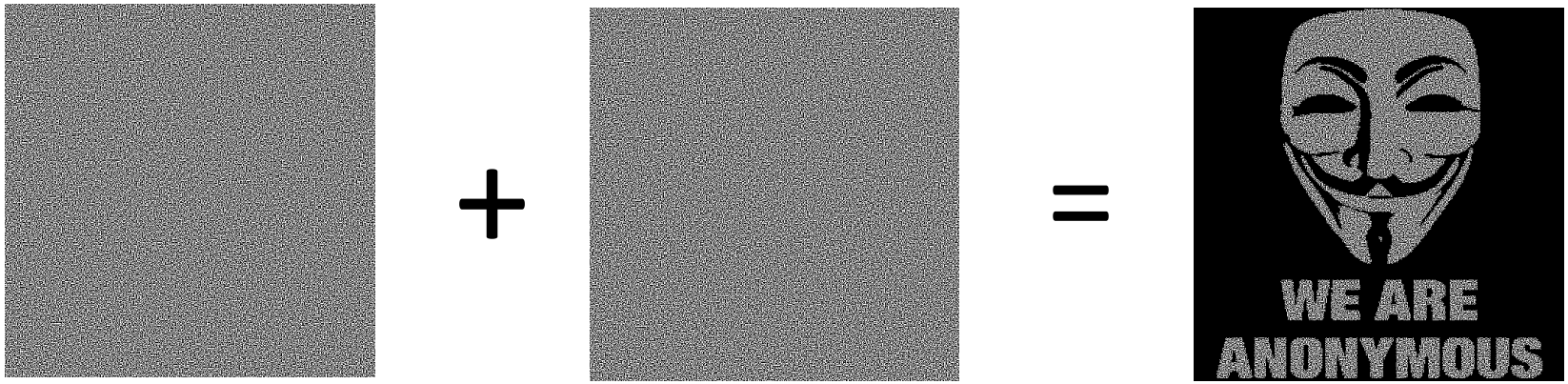
- Veľkosť podielu je rovnaká ako veľkosť tajomstva
- Skupina $t-1$ účastníkov nevie o tajomstve vypočítať nič

$$f(x_1) - s' = a_{t-1}x_1^{t-1} + \cdots + a_1 x_1,$$
$$f(x_2) - s' = a_{t-1}x_2^{t-1} + \cdots + a_1 \mathrm{x}_2,$$
$$\vdots$$
$$f(x_{t-1}) - s' = a_{t-1}x_{t-1}^{t-1} + \cdots + a_1 x_{t-1},$$

pre každé s' existuje f', také že $f'(x_i) = f(x_i)$ pre $i = 1, \ldots, t-1$

- Nečestný účastník
  - Čo ak účastník podhodí falošný podiel?
  - Čo ak účastník po odhalení $t-1$ podielov svoj podiel neodhalí?

Vizuálne zdieľanie tajomstva



Kryptografia a zraniteľnosti

Niektoré zraniteľnosti súvisiace kryptografiou

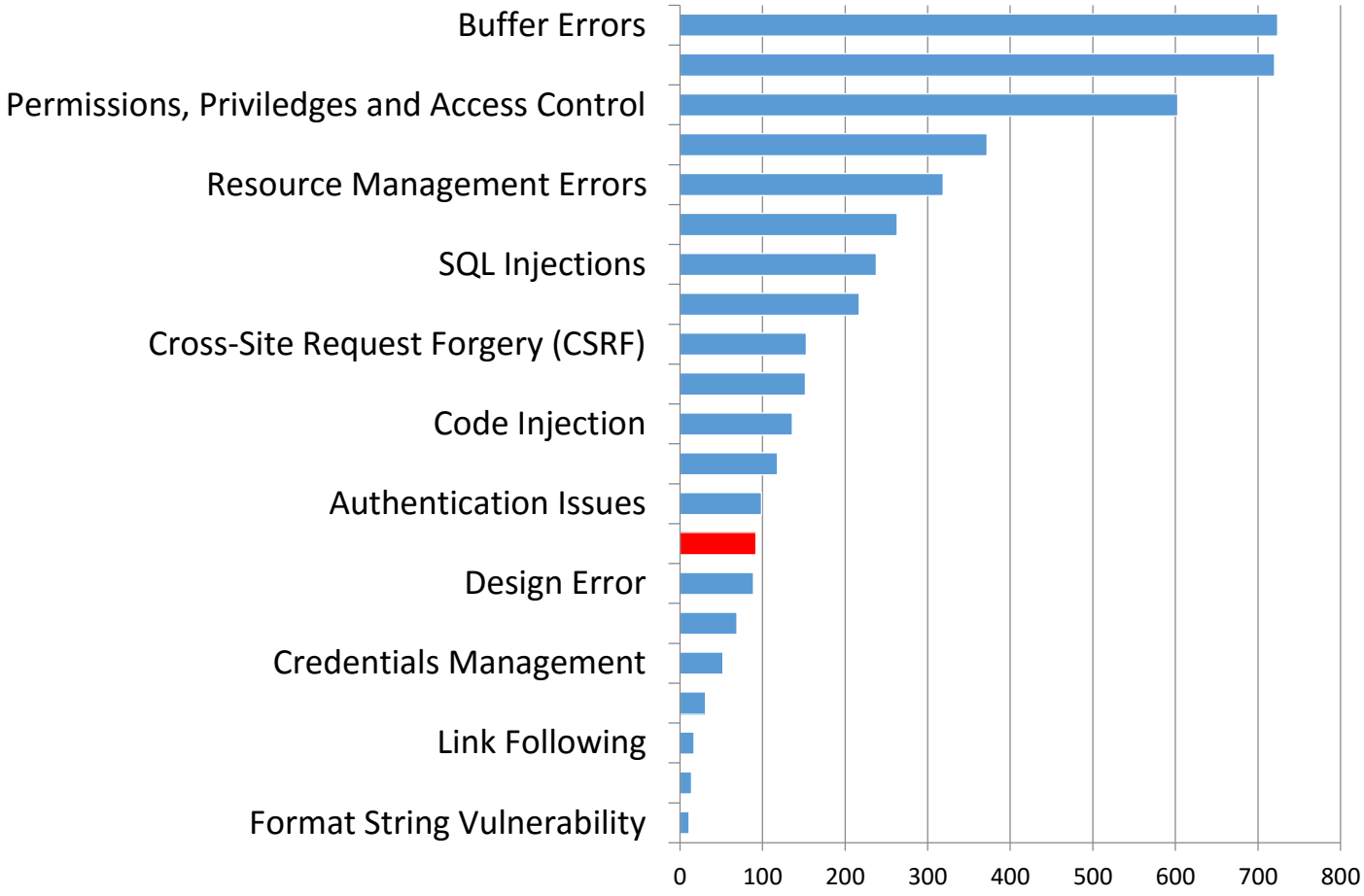
Kryptografia a zraniteľnosti

- Útoky na kryptografické mechanizmy
  - Obvykle sú slabiny v správe kľúčov a v implementácii
  - Protokoly – zvyčajne slabiny v protokole, bez ohľadu na algoritmy
- Niektoré implementačné slabiny/útoky
  - Útok postrannými kanálmi (napr. timing útok)
  - Nesplnenie bezpečnostných predpokladov (napr. náhodnosť)
  - Slabiny v protokoloch (napr. útoky na SSL/TLS)
  - Slabé algoritmy (napr. proprietárne algoritmy ako CCS)

Kryptografia a zraniteľnosti

- NIST: NVD (National Vulnerability Database)
  - SW zraniteľnosti a ich klasifikácia (typ, závažnosť a pod.)
- Najčastejšie zraniteľnosti v „Cryptographic Issues“:
  - použitie nekvalitného zdroja náhodnosti pri generovaní kľúčov,
  - nedostatočná (neúplná) kontrola certifikátov,
  - nekorektná implementácia kryptografických algoritmov alebo protokolov,
  - fixné heslá servisných účtov alebo heslá odvodené z verejne známych údajov

Počty zraniteľností publikovaných v roku 2012 podľa NVD



Generátory náhodných čísel

- Častá príčina zlyhania kryptografických systémov
  - Generovanie skutočne náhodných čísel je ťažké
- (Ne)kryptografické generátory pseudo-náhodných čísel bývajú predvídateľné
- Určite nepoužívajte “rand()” funkciu zabudovanú v programovacom jazyku
  - srand(seed) inicializuje generátor, nastaví `state=seed`
  - `rand()`:
    - `state=f(state)`, kde f je nejaká lineárna funkcia
    - `return state;`
  - Generovanie 128-bitového kľúča KEY=`rand()‖rand()‖rand()‖rand()`
  - Entropia kľúča je iba 32 bitov!

Generátory náhodných čísel

- Generátor pseudo-náhodných čísel použiteľný v kryptografii
  - Výstup neodlíšiteľný od úplne náhodného akýmkoľvek efektívnym algoritmom
  - Pokiaľ možno, zakaždým reinicializovaný novým zdrojom entropie
  - Malo by byť ťažké uhádnuť interný stav generátora
    - Napr. entropia by nemala pochádzať iba z času (súborov)

- „Cold boot“ problémy
  - Server práve naštartoval a potrebuje zdroj náhodnosti ... je možné získať dosť entropie, ak server beží len pár sekúnd?

Generátory náhodných čísel

najznámejšie zraniteľnosti

- Netscape, implementácia SSL, 1995
  - Pseudonáhodný generátor inicializovaný na základe času, ID procesu a ID nadradeného procesu – všetko ľahko predvídateľné hodnoty
  - Generátor nebol verejne dostupný („security through obscurity“), na analýzu využili reverzné inžinierstvo
- Windows 2000 / XP, 2007
  - Leo Dorrendorf - *Cryptanalysis of the Random Number Generator of the Windows Operating System,* http://eprint.iacr.org/2007/419.pdf
  - Vážne nedostatky vstavaného generátora
  - Ak sa útočníkovi podarilo získať stav generátora (napr. cez buffer overflow), mohol predpovedať všetky predchádzajúce aj nasledujúce vygenerované hodnoty (napr. SSL šifrovacie kľúče)
  - Opravené v XP SP3

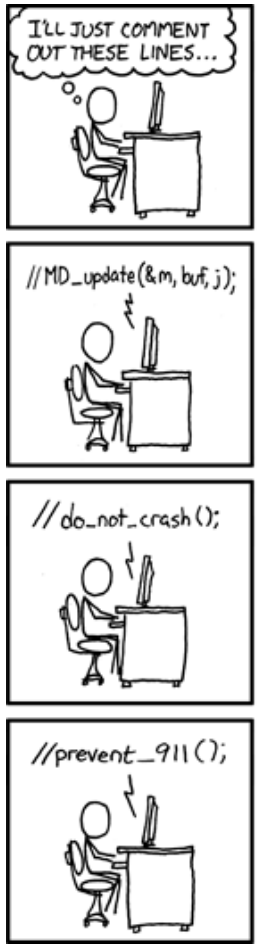
Generátory náhodných čísel

najznámejšie zraniteľnosti

- Debian OpenSSL, 2008
  - Debian distribúcie Linuxu
  - Zmeny v kóde pseudo-náhodného generátora drasticky znížili entropiu
  - Chyba bola spôsobená vývojárom, ktorý na základe upozornení kompilátora odstránil na pohľad zbytočný kód

```
MD_Update(&m,buf,j); /* neinicializovaná hodnota */
[ .. ]
MD_Update(&m,buf,j); /* neinicializovaná hodnota */
```

    - Odstránený kód zabezpečoval zvýšenie entropie
    - Po jeho odstránení bol generátor inicializovaný len na základe ID procesu (max. 32 768 hodnôt)
  - Chyba umožnila odhaliť vygenerované súkromné kľúče
  - Veľké množstvo kľúčov a certifikátov muselo byť vygenerovaných znovu



Generátory náhodných čísel

najznámejšie zraniteľnosti

- PlayStation 3, 2010
  - Sony využíva ECDSA algoritmus na podpisovanie softvéru pre PlayStation 3
  - ECDSA vyžaduje dobrý PRNG
  - Opakované použitie k vedie k odhaleniu súkromného kľúča
  - SONY použilo zakaždým tú istú hodnotu k

Podpisovanie v ECDSA $\mathrm{Sig}_x(m)$:

1. $k \xleftarrow{\$} \{1, \ldots, n-1\}$
2. $(x_1, y_1) = k \times G$
3. $r = x_1 \bmod n$
4. $s = k^{-1}(H(m) +$

Generátory náhodných čísel

najznámejšie zraniteľnosti

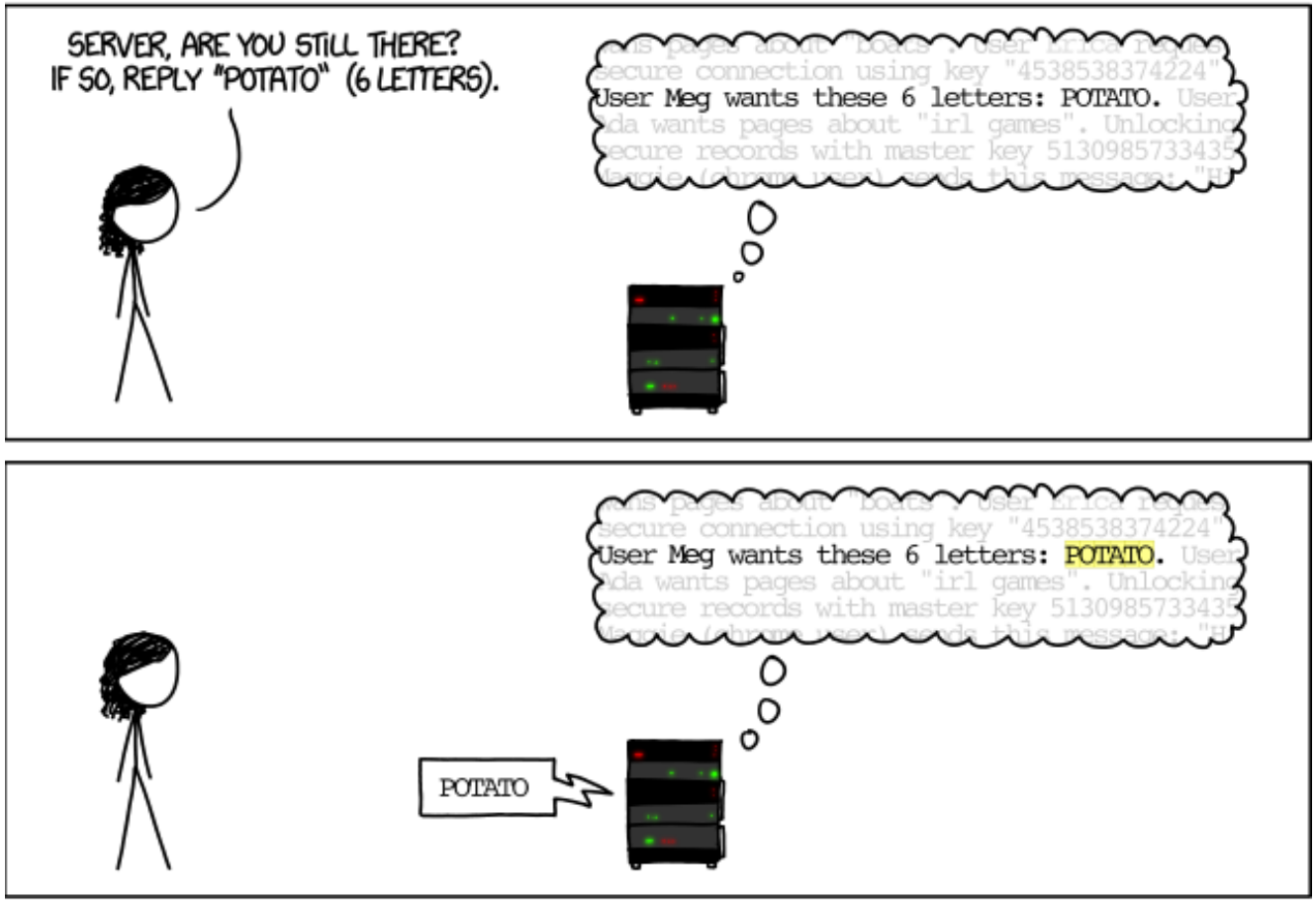
- Implementácia Bitcoinov v Androide, 2013
  - Chyba v Java triede SecureRandom - možné kolízie v hodnote k pri použití ECDSA
  - Kolízia vedie k odhaleniu súkromného kľuča – možnosť ukradnúť Bitcoiny z Androidovej peňaženky
- DUAL_EC_DRBG, 2007, 2013
  - NIST Special Publication 800-90 – kolekcia pseudo-náhodných generátorov
  - DUAL_EC_DRBG – odporúčaný / navrhovaný aj NSA
    - Kryptografia nad eliptickými krivkami - štandard obsahuje aj sadu odporúčaných kriviek / konštánt
    - 2007, Shumow, Ferguson ukázali, že konštanty mohli byť skonštruované tak, aby umožňovali „zadné vrátka“ k náhodnému generátoru
  - 2013, REUTERS – Snowden: NSA zaplatilo firme RSA $10 mil., aby bol predvolený generátor práve DUAL_EC_DRBG

Heartbleed

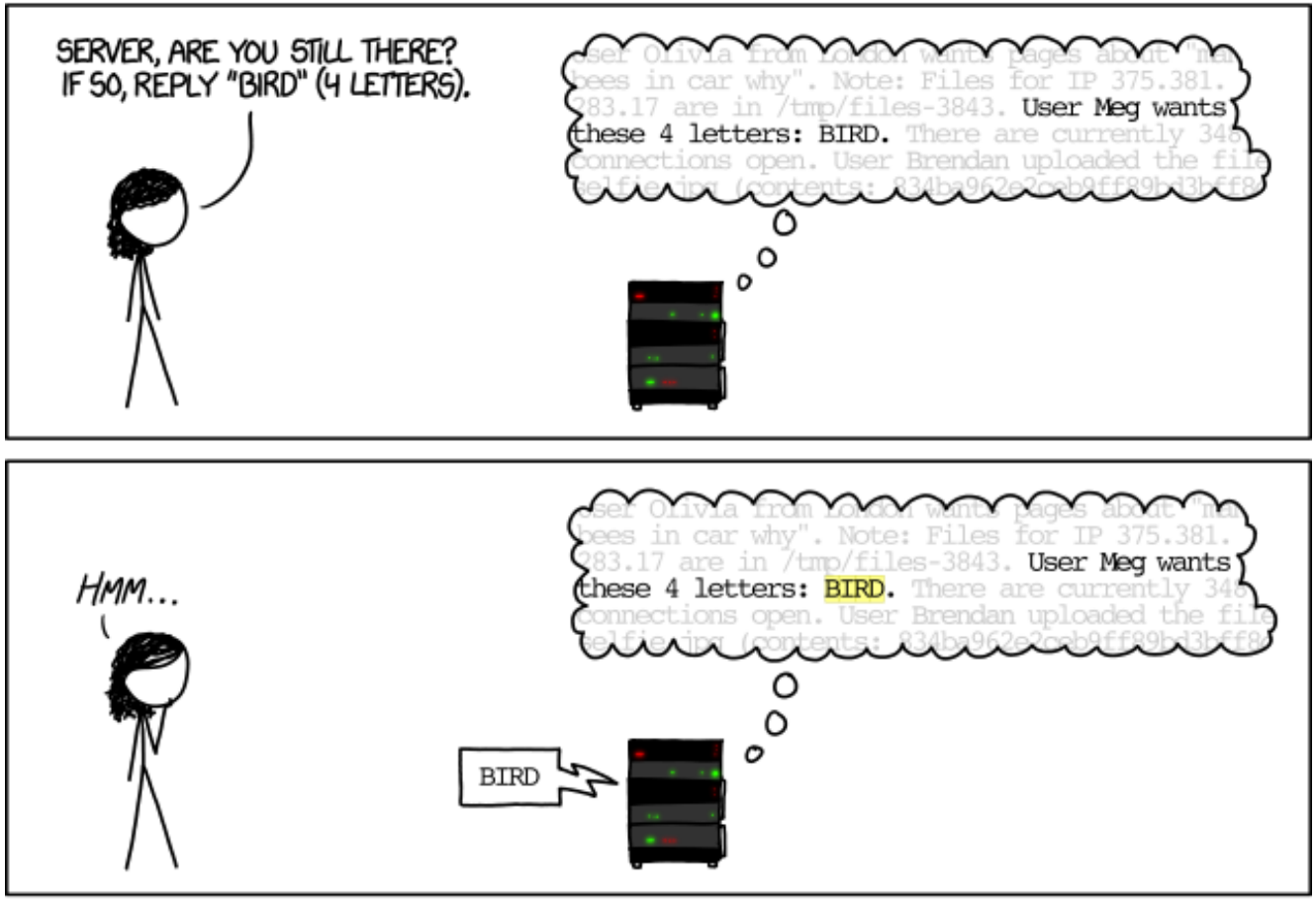
- Apríl 2014
- Zraniteľnosť v rozšírení „Hearbeat“ pre OpenSSL
  - „keep alive“ pre TLS
  - Každá stránka obsahuje veľa súčastí (obrázky, skripty, štýly)
  - Rozšírenie heartbeat zabezpečovalo, aby nebolo potrebné zakaždým negociovať nové kľúče

- Chyba v implementácii umožnila na diaľku čítať pamäť servera

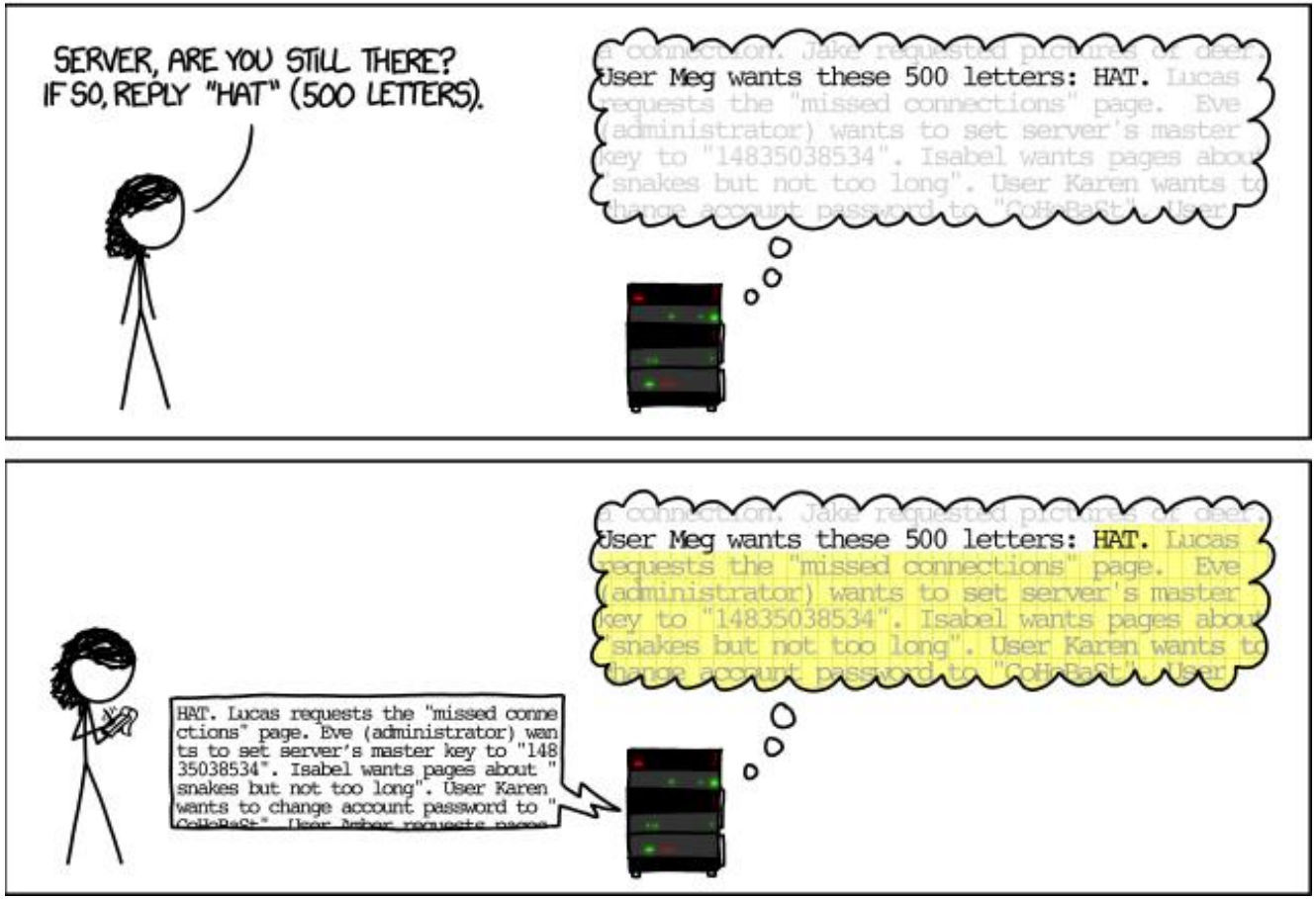
HOW THE HEARTBLEED BUG WORKS:



http://xkcd.com/1354/



http://xkcd.com/1354/



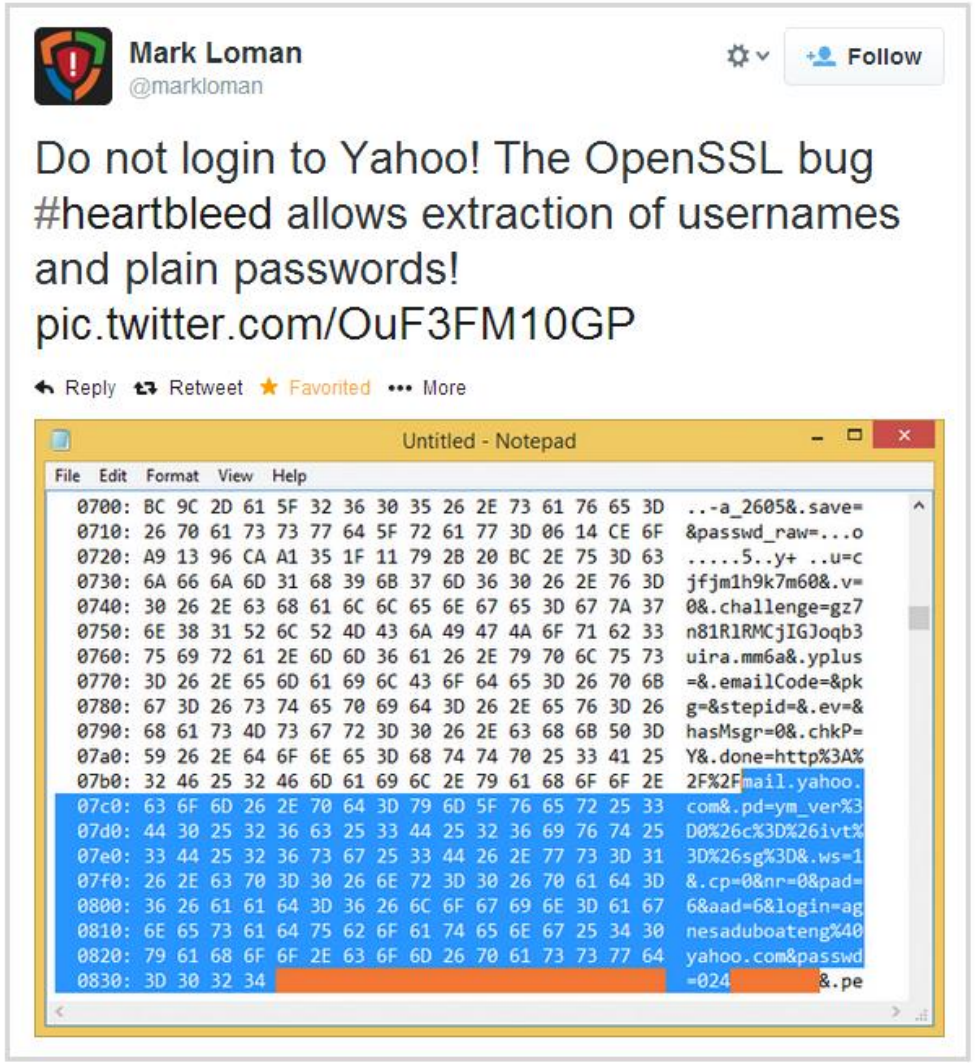
http://xkcd.com/1354/

Heartbleed

```
buffer
  = OPENSSL_malloc(
        1 + 2 + payload
        + padding
  );
```

`Payload + padding` – hodnota ovládaná klientom

Zakaždým bolo možné získať max. 64kb obsahu pamäte servera



Apple goto fail

- Február 2014
- iOS < 7.0.5
- OS X < 10.9.2
- Chyba v implementácii SSL/TLS klienta
- Nedochádza k správnemu overeniu digitálneho podpisu servera
- Možný „man in the middle útok“

Prvé kroky TLS:

C $\rightarrow S$: Zoznam podporovaných šifrovacích algoritmov

$S \rightarrow$ C: S vyberie „Ephemeral Diffie Hellman“, vygeneruje DH parameter g^a pošle to podpísané svojim tajným kľúčom klientovi

$C \rightarrow S$: C overí podpis parametrov, vygeneruje g^b, pošle to S

$C \leftrightarrow S$: „session“ kľuč g^{ab}

- ďalšia komunikácia je šifrovaná týmto kľúčom

Apple goto fail

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
        OSStatus        err;
        ...

        if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
                goto fail;
        if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
                goto fail;
                goto fail;
        if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                goto fail;
        ...

fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
}
```

Tento riadok tu nemá byť, zakaždým sa vykoná

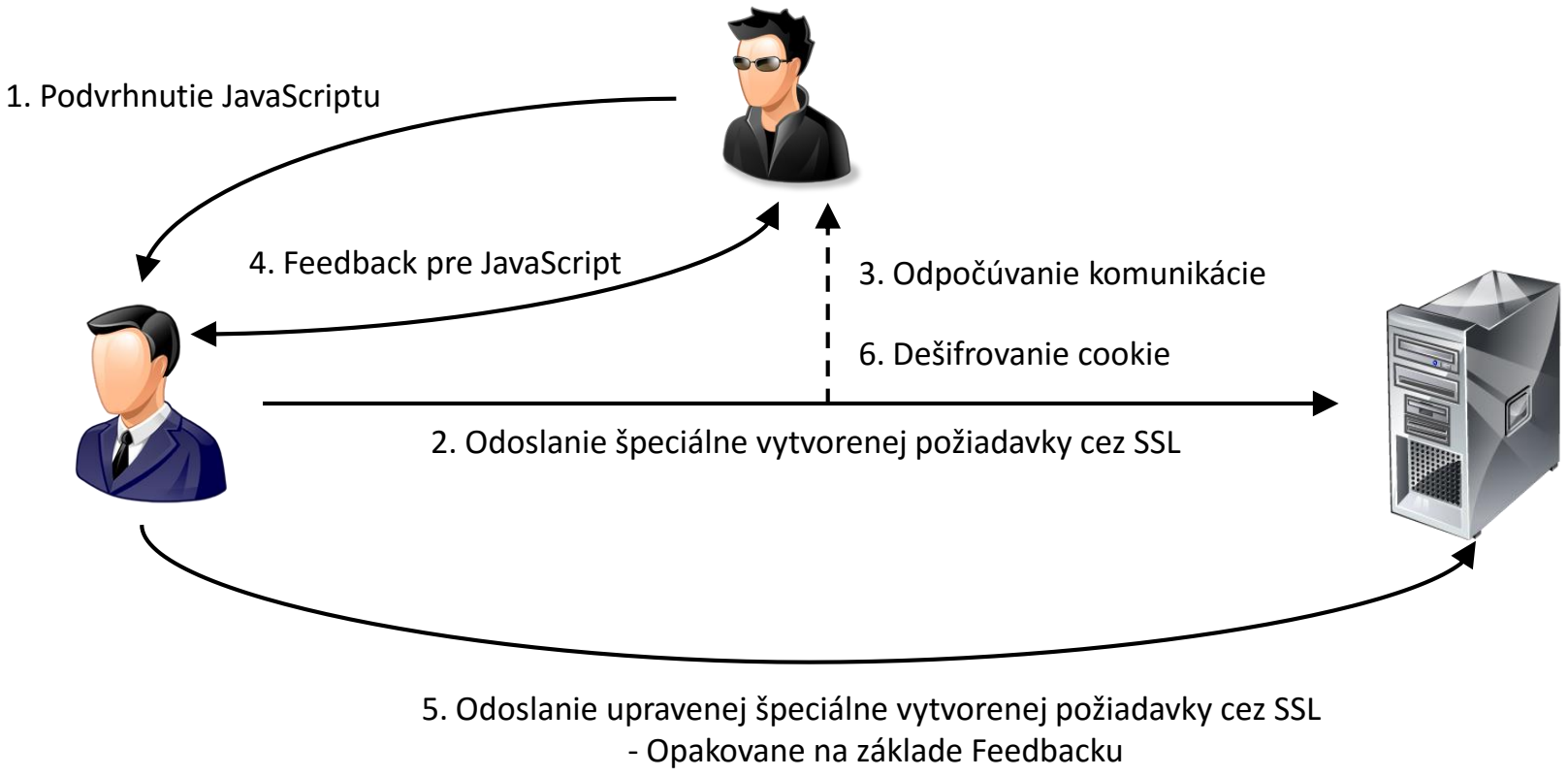
Kód zakaždým preskočí sem a err obsahuje hodnotu reprezentujúcu úspešné overenie podpisu

BEAST útok, 2011

- Aplikovateľný na šifrovacie schémy bežiace v CBC móde v rámci SSL 3.0 a TLS 1.0
- Využíva nedostatočnú náhodnosť IV v týchto šifrovacích schémach
  - Teoretický útok navrhnutý už v roku 1995 (Rogaway)
  - IV inicializovaný posledným blokom šifrového textu predošlého packetu
- Predpoklady:
  - Zapnutý JavaScript – „Man in the browser“ – útočník cez JavaScript posiela na server dotazy
  - Možnosť byť „Man in the middle“ – odpočúvať a posielať packety v sieti
- Útočník uprostred môže odhaliť „session cookie“ obete (napr. Do stránky PayPal.com)

  => t.j. pracovať so systémom v mene obete

BEAST útok, 2011



BEAST útok, 2011

- Prehliadače hneď vydali záplaty
- TLS 1.1 a 1.2 nie sú ovplyvnené – IV je generovaný náhodne
- Teoretická možnosť odhalená v roku 1995 sa stala praktickou
  - Útoky sa časom zlepšujú!
- Útok využíval nástroje z viacerých oblastí bezpečnosti
  - Man-in-the-browser cez JavaScript
  - Nedostatočnú náhodnosť IV
  - Možno sa budú dať tieto nástroje využiť aj pri iných útokoch

CRIME útok, 2012

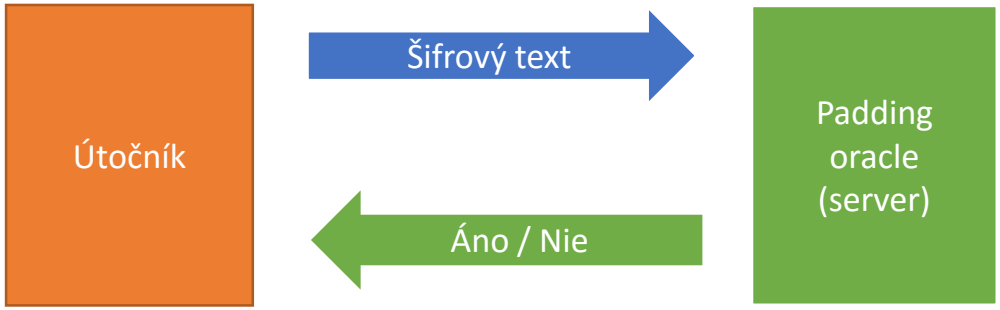
- Zneužíva možnosť kompresie posielaných údajov cez TLS
- Teoretický útok známy od roku 2004 (Kelsey)
- Idea:
  - Dĺžka šifrového textu určuje dĺžku otvoreného textu
  - Ak poznáme dĺžku otvoreného textu, vieme pomer kompresie
  - Pomer kompresie odhaľuje niečo (hoci málo) o otvorenom texte
- Útok umožňuje odhalenie „session cookie“
  - Predpoklady podobné ako v BEAST útoku
    - „man in the browser“ – JavaScript pošle na server vhodne zvolenú správu (CPA útok)
    - „man in the middle” – Možnosť odpočúvať packety v sieti
- Zabránime mu vypnutím kompresie v prehliadači / na serveri

CRIME útok, 2012

- Teoretická zraniteľnosť z roku 2004, praktický útok v roku 2012
  - Útoky sa časom zlepšujú!
- Nástroje vyvinuté pre BEAST útok boli využíté aj v CRIME útoku
  - A možno sa budú dať použiť aj inde?
- Útoky podporili rozšírenie novších (a lepších) štandardov TLS 1.1 resp. 1.2

Error message attack

- Server implementujúci kryptografický protokol môže reagovať rôzne, ak (zašifrované) dáta ktoré prijal majú
  - **správny** tvar (napr. otvorený text má správny padding)
  - **nesprávny** tvar (napr. otvorený text má nesprávny padding)
- Tzv. padding oracle útoky



Error message attack

- 2002, Vaudenay – útok na sym. šifry v CBC móde
  - Útočník môže s využitím „timing padding“ orákula dešifrovať správy (resp. vytvoriť správne zašifrovaný text)
- ...
- 2013, Peterson a kol. „Lucky 13“ útok na TLS
  - “Timing padding oracle útok”
  - TLS síce nedáva rôzne chybové hlásenia, avšak
  - ak má správa zlý padding – čas, odpovede je iný ako v prípade správneho paddingu

Lucky 13

Špecifikácia TLS 1.2:

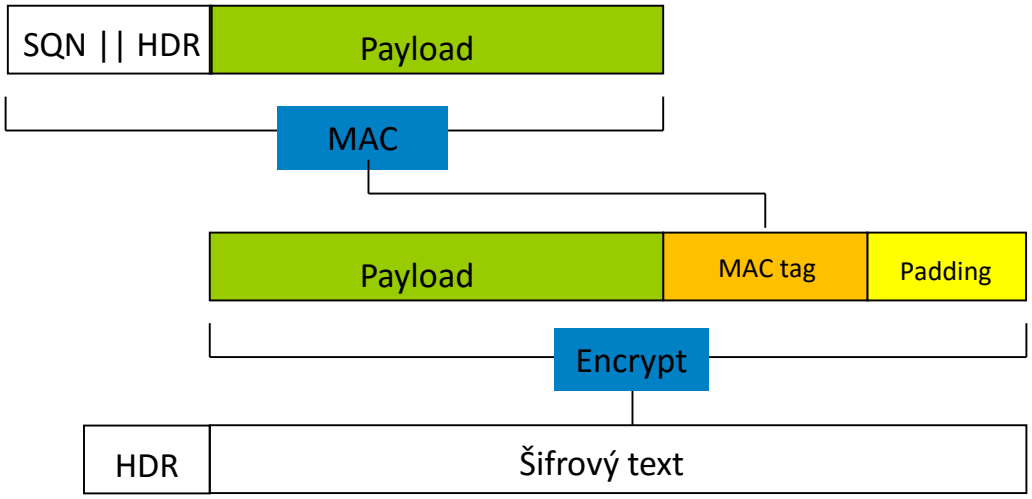
…implementations MUST ensure that record processing time is essentially the same whether or not the padding is correct.

In general, the best way to do this is to compute the MAC even if the padding is incorrect, and only then reject the packet.

- Avšak ak je zlý padding, aký padding máme použiť?

Lucky 13

TLS Record Protokol: MAC-Encode-Encrypt



Problém je, ako v prípade zlého paddingu určiť, čo je Payload, MAC tag a Padding

Lucky 13

For instance, if the pad appears to be incorrect, the implementation might assume a zero-length pad and then compute the MAC.

- Tento prístup využívalo viacero implementácií vrátate OpenSSL, NSS (Chrome, Firefox), BouncyCastle, OpenJDK, …

… This leaves a small timing channel, since MAC performance depends to some extent on the size of the data fragment, but it is not believed to be large enough to be exploitable, due to the large block size of existing MACs and the small size of the timing signal.

Lucky 13

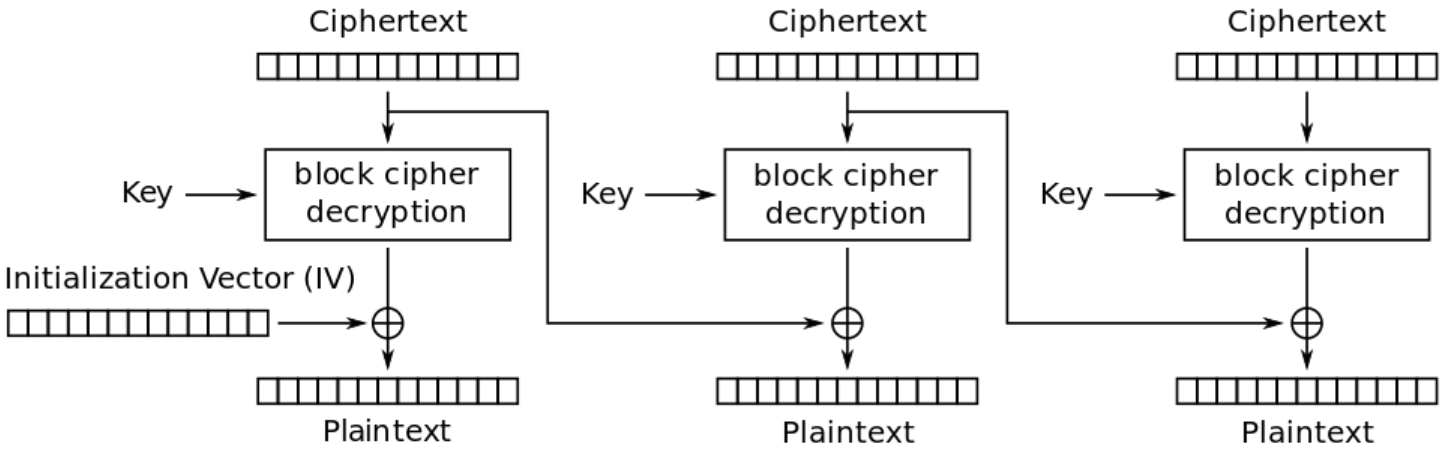
- Útok voči TLS-CBC šifrám implementovaných podľa odporúčania špecifikácie TLS 1.2
  - Útočník môže dešifrovať komunikáciu - napr. „session cookie“
- Aplikovateľný na všetky verzie SSL/TLS
  - SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2 pri použití CBC šifry
  - TLS 1.2 podporuje aj šifry v GCM a CCM móde, ktoré sú odolné
- Využitie BEAST prístupu
  - „man in the browser“ cez JavaScript

Poodle útok, 2014

- Ďalší padding oracle útok na CBC šifry v SSL 3.0
- Požiadavka klienta na server (rozdelená na 8 bajtové bloky):

- Posledný blok obsahuje padding a jeho dĺžku
  - V našom prípade 7 bajtov
- SSL 3.0 nešpecifikuje ako má vyzerať padding
  - T.j. je akceptovaný vtedy a len vtedy ak posledný bajt je 7
- Útočník odpočuje šifrovanú podobu vyššie uvedenej požiadavky
  - Duplikuje blok obsahujúci cookie a nahradí nim posledný blok

Poodle útok, CBC dešifrovanie

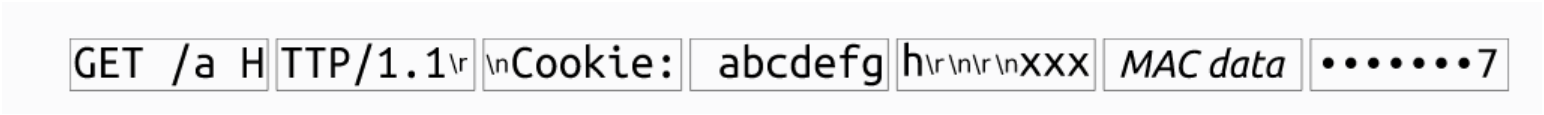


- Ak SSL 3.0 server správu akceptuje (t.j. má správny padding)
  - cookie blok $\oplus$ predchádzajúci šifrový blok = xxxxxx7

=> Útočník pozná posledný znak cookie

Poodle útok, 2014

- Následne útočník vytvorí novú požiadavku, kde je cookie blok posunutý:



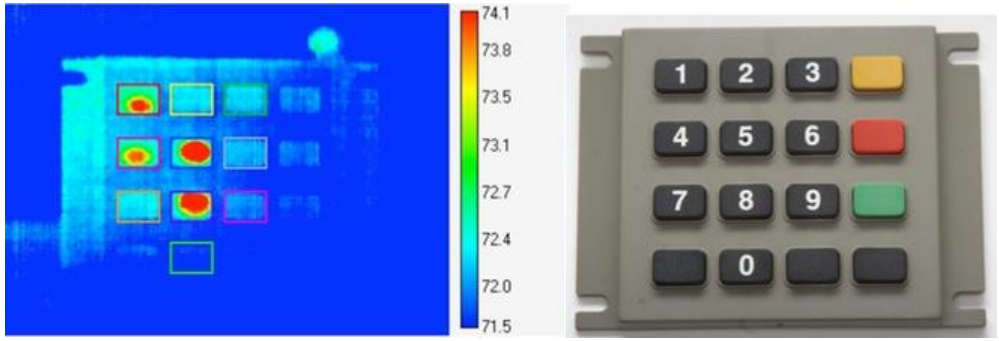
=> útočník získa predposledný znak cookie

⋮

=> Útočník získa celý cookie

Ďalší útok: Bankomaty a PIN

- Pomocou termokamery je možné čítať PIN



- Niekedy je dokonca možné určiť poradie stlačenia kláves
- Kovové klávesy sú náchylnejšie