Útok na privátní podpisové klíče formátu OpenPGP, programů PGP™ a dalších aplikací kompatibilních s OpenPGP

Vlastimil Klíma [1] a Tomáš Rosa [2]

[1] Decros spol. s r.o., člen skupiny ICZ a.s., Praha, v.klima@decros.cz
[2] Decros spol. s r.o., člen skupiny ICZ a.s., Praha a ČVUT - FEL Praha, t.rosa@decros.cz

Abstrakt. V příspěvku je popsán útok na formát OpenPGP, který vede k odhalení privátních podpisových klíčů algoritmů DSA a RSA. Formát OpenPGP je použit v řadě aplikací, včetně programů PGP, GNU Privacy Guard a dalších, uvedených na seznamu produktů, kompatibilních s OpenPGP, které jsou uvedeny na stránce http://www.pgpi.org/products. Proto všechny tyto aplikace musí projít stejnou revizí jako vlastní program PGP™. Úspěšnost útoku byla prakticky ověřena a demonstrována na programu PGP™(*) verze 7.0.3 s kombinací algoritmů AES a DH/DSS. Protože privátní podepisovací klíč je základní utajovanou informací celého systému, je zašifrován silnou šifrou. Ukazuje se však, že tato ochrana je iluzorní, protože útočník nemusí útočit ani na tuto šifru, ani na tajné přístupové heslo uživatele. K útoku stačí určitým způsobem modifikovat soubor s privátním klíčem a následně zachytit jednu podepsanou zprávu. Nedostatečné zajištění integrity veřejných i privátních částí podepisovacích klíčů ve formátu OpenPGP je analyzováno u algoritmů DSA a RSA a na základě toho je ukázán postup útoků na oba privátní podpisové klíče. Útoky se týkají všech délek parametrů (modulů, klíčů) RSA a DSA. V závěru se navrhují kryptografická opatření pro opravu formátu OpenPGP i programu PGP™.

1 Úvod

Formát OpenPGP je definován v dokumentu [1] z roku 1998. Jeho cílem bylo publikovat všechny nezbytné informace o formátu OpenPGP, aby na jeho bázi mohly být vytvářeny různé interoperabilní aplikace. Popisuje formáty zpráv (datových struktur) a způsob, jak mají být vytvářeny. V tomto příspěvku poukazujeme na závažnou chybu formátu OpenPGP, která spočívá v nedostatečném zajištění integrity veřejných i privátních částí podepisovacích klíčů algoritmů DSA a RSA. Ukazujeme, že tato chyba může být využita k odhalení privátního klíče. Aby byl privátní klíč chráněn, je jeho hodnota uložena do privátní klíčenky (souboru) secring.skr v zašifrovaném tvaru. K tomu je použita uživatelem zvolená silná symetrická šifra (AES, CAST5, IDEA) s dostatečně dlouhým klíčem, který je odvozen od tajného přístupového hesla (password, passphrase), které zná jen uživatel. V příspěvku ukazujeme, že pokud útočník má přístup k tomuto souboru (nebo záznamu), může za určitých okolností získat privátní podpisový klíč uživatele, aniž by znal jeho přístupové heslo nebo proti němu útočil. Útok spočívá ve speciální modifikaci parametrů podpisového algoritmu a získání podpisu libovolného souboru (e-mailové zprávy) takto modifikovaným podpisovým klíčem. Ukazujeme, že na základě toho je útočník schopen vypočítat privátní podepisovací klíč. Protože může narušený záznam nebo soubor (secring.skr) uvést zpět do původního stavu, je tento útok velmi nebezpečný. Ve stejném ohrožení jsou i privátní klíče, přenášené v zašifrovaném tvaru na disketách nebo po síti. Útok byl ověřen prakticky na nejnovější verzi programu PGP™ 7.0.3 s kombinací algoritmů AES a DH/DSS a jeho výsledkem bylo získání privátního podpisového klíče algoritmu DSA. V závěru navrhujeme určitá bezpečnostní a kryptografická opatření pro opravu formátu OpenPGP a změny v programu PGP™.

Podrobnou revizí musí projít všechny další aplikace, které jsou kompatibilní s formátem OpenPGP. Jedná se například o GNU Privacy Guard a další, uvedené na seznamu aplikací, kompatibilních s OpenPGP, které jsou uvedeny na stránce http://www.pgpi.org/products.
Následující text je organizován takto: nejprve si připomeneme definici podpisového schématu DSA a formát uložení privátních klíčů podle OpenPGP [1] . Poté popíšeme ideu útoku na privátní podpisový klíč DSA a konkrétní postup útoku tak, jak jsme ho provedli v programu PGP™. Dále popíšeme ideu útoku na podpisový klíč RSA, uložený podle formátu OpenPGP. Poté uvedeme základní přechodná opatření pro ochranu privátních klíčů v programu PGP™ a návrhy na revizi formátu OpenPGP. V dodatcích uvádíme technické podrobnosti útoků.

2 Podpisový algoritmus DSA

Pro potřeby tohoto článku si v tomto odstavci připomeneme postup vytváření klíčového páru a podpisu pomocí 1024bitového algoritmu DSA (viz např. [2], str. 452) a postup verifikace podpisu a zavedeme potřebné proměnné.

2.1 Vytváření klíčového páru

Označme h hašovací funkci SHA-1. Každý uživatel vygeneruje privátní a veřejný klíč následujícím postupem:

1. Zvol 160bitové prvočíslo q tak, že $2^{159} < q < 2^{160}$.
2. Zvol 1024bitové prvočíslo p tak, že q dělí (p-1) a $2^{1023} < p < 2^{1024}$.
3. Vyber generátor g cyklické podgrupy řádu q v $\mathbf{Z}_p^*$ (tj. zvolí se prvek $h \in \mathbf{Z}_p^*$ tak, že $g = h^{(p-1)/q} \bmod p$ a $g \neq 1$, jinak se volí jiné h).
4. Vyber náhodné číslo x tak, že $1 \leq x \leq q-1$.
5. Vypočti $y = g^x \bmod p$.
6. Veřejný klíč je y, veřejné parametry jsou (p, q, g), privátní klíč je x.

2.2 Vytváření digitálního podpisu

Uživatel při vytváření podpisu zprávy m (resp. její hašovací hodnoty h(m)) používá svůj privátní klíč x a veřejné parametry podle následujícího postupu:

1. Vyber náhodné tajné číslo k, $0 < k < q$.
2. Vypočti $r = (g^k \bmod p) \bmod q$.
3. Vypočti $kInv = k^{-1} \bmod q$.
4. Vypočti $s = [kInv * (h(m) + x*r)] \bmod q$.
5. Digitální podpis zprávy m je dvojice (r, s).

Poznamenejme, že r, s, q jsou obecně 160bitová čísla, zatímco p, g, y jsou 1024bitová.

2.3 Verifikace digitálního podpisu

Při verifikaci digitálního podpisu zprávy m použijeme veřejný klíč signatáře (p, q, g, y) podle následujícího postupu.

1. Ověř, že $0 < r, s < q$. V opačném případě je podpis neplatný.
2. Vypočti $sInv = s^{-1} \bmod q$ a hašovací hodnotu h(m).
3. Vypočti $u1 = sInv * h(m) \bmod q$, $u2 = sInv * r \bmod q$.
4. Vypočti $v = (g^{u1} * y^{u2} \bmod p) \bmod q$.
5. Podpis je platný, právě když v = r.

3 Popis datové struktury Secret Key Packet pro uložení privátního podpisového klíče podle OpenPGP

Zde popíšeme datovou strukturu (Tag) "Secret Key Packet", ve které je uložen privátní podpisový klíč (RSA, DSA). Existují dvě verze tohoto formátu. Verze 3 se týká jen klíčů RSA, verze 4 může zahrnovat klíče DSA i RSA. Na podpisový klíč RSA budeme útočit prostřednictvím obou verzí formátu a na podpisový klíč DSA prostřednictvím verze 4 formátu. Verzi 4 tohoto formátu používá i program PGP™ 7.0.3, který preferuje DSA oproti RSA. V praxi se proto budeme setkávat zejména s klíči RSA ve formátu 3 a s klíči DSA ve formátu 4. Poznamenejme, že verze 3 a 4 se liší ve způsobu šifrování privátních dat, a proto se liší i útoky na oba algoritmy. V obou verzích formátu ale struktura Secret Key Packet obsahuje na počátku nejprve data ze struktury Public Key Packet, týkající se veřejného klíče, a poté data, týkající se privátního klíče. Popis a obsah jednotlivých položek pro algoritmus DSA znázorňuje tabulka 1 a pro algoritmus RSA tabulka 2. K obsahu tabulky připomeňme, že formát MPI (multiprecision integer) obsahuje prefix a poté vlastní (velké) celé číslo v zápisu BIG ENDIAN. Prefix tvoří dva oktety v BIG ENDIAN a označuje počet platných *bitů* následného čísla [1] .

Public Key Packet	1 oktet označující číslo verze	Oblast veřejně přístupných dat
	4 oktety označující čas, kdy byl klíč vytvořen	
	1 oktet označující algoritmus pro vytváření digitálního podpisu. Následují pole (čísla ve formátu MPI), obsahující veřejné parametry a veřejný klíč pro 1024bitový algoritmus DSA:	
	prvočíslo p (2 + 128 oktetů v obrázku 1)	
	prvočíslo q (2 + 20 oktetů v obrázku 1)	
	číslo g (2 + 128 oktetů v obrázku 1)	
	veřejný klíč uživatele y (2 + 128 oktetů v obrázku 1)	
1 oktet (string-to-key usage), indikující, zda a jak je šifrován privátní klíč. Je preferována hodnota 0xFF, znamenající, že následující tři volitelné položky jsou vyplněny.		
[Volitelné] V případě, že string-to-key usage je 0xFF, je zde 1 oktet, identifikující symetrický šifrovací algoritmus pro ochranu privátního klíče.		
[Volitelné] V případě, že string-to-key usage je 0xFF, je zde "string-to-key specifier", který říká, jak je password uživatele zpracován na symetrický klíč. Je preferována hodnota 0x03, označující tzv. iterovaný a solený "string-to-key specifier". Typicky jsou zde uložena následující data s tímto významem: *1 oktet: 0x03 (iterated and salted string-to-key identifier)* *1 oktet: identifikátor hašovacího algoritmu (pro SHA-1 je to 0x02)* *8 oktetů: sůl (náhodná data, která se hašují společně s přístupovým heslem uživatele a diverzifikují tak derivovaný symetrický klíč)* *1 oktet: počet hašovaných oktetů dat (tzv. "count").*		
[Volitelné] Jestliže je privátní klíč šifrován, je zde uložen inicializační vektor (IV). Jsou to náhodná data v délce bloku použité blokové šifry (8 oktetů pro 64bitové blokové šifry, 16 oktetů pro algoritmus AES).		
Algoritmicky závislá čísla ve formátu MPI. Pro DSA je zde pouze privátní exponent x:		Oblast citlivých dat
2 oktety, prefix čísla x (ve verzi 4 šifrováno, ve verzi 3 nešifrováno)		
20 oktetů, číslo x (ve verzi 3 i 4 šifrováno)		
2 oktety, checksum, aritmetický součet předchozích 22 oktetů v otevřeném tvaru, modulo 65536 (ve verzi 4 šifrováno, ve verzi 3 nešifrováno).		

Tabulka 1: Obsah struktury Secret Key Packet pro 1024bitový algoritmus DSA

Nyní se ještě zastavíme u souborů typu secring.skr v programech PGP™, kam program PGP™ ukládá strukturu Secret Key Packet. V tomto souboru je kromě pole Secret Key Packet uloženo obvykle ještě několik dalších záznamů jako jsou například

- UserID Packet (obsahuje identifikátor uživatele, tj. jeho jméno a e-mailovou adresu),
- Signature Packet (obsahuje digitálně podepsané hodnoty důvěry, času podpisu, expiraci klíče a pod.),
- Secret Subkey Packet (obsahuje podobné údaje jako Secret Key Packet, ale tentokrát o asymetrickém klíči a algoritmu pro šifrování dat),
- další Signature Packet (obsahuje údaje jako například čas podpisu tohoto klíče podpisovým klíčem, velikost důvěry k němu a pod.).

Tyto další záznamy ale neobsahují žádnou kontrolu integrity záznamu Secret Key Packet. To umožní úspěšně na Secret Key Packet útočit.
Nyní si popíšeme útok zvlášť na RSA a zvlášť na DSA.

4 Útok na podpisový algoritmus DSA

Povšimněme si, že integrita pole "Public Key Packet" v rámci struktury "Secret Key Packet" není nikde viditelně zajištěna ani ve formátu OpenPGP, a jak se ukázalo praktickým útokem, ani v programech PGP™. Přitom při vytváření digitálního podpisu dochází právě k využití veřejných parametrů tohoto pole (v případě programu PGP™ je Secret Key Packet uložen konkrétně v souboru secring.skr). Tyto parametry by se sice mohly číst ze záznamu veřejného klíče (souboru pubring.pkr), ale je logické, že pokud je otevřen záznam privátního klíče, budou čteny odtud. Přitom v záznamu Secret Key Packet je chráněna hodnota privátního podepisovacího klíče, ale chybou je, že zde není nijak chráněna hodnota veřejných parametrů ani veřejného klíče. Konkrétně se v případě DSA jedná o hodnoty p, q, g, y, z nichž využijeme ke konkrétnímu útoku pouze p, g.
Hlavní myšlenka útoku na DSA spočívá v následujících krocích. Útočník:

1. si připraví speciální čísla (konstanty) PGPrime a PGGenerator
2. získá strukturu Secret Key Packet daného uživatele a ve struktuře "Public Key Packet" uvnitř ní na místo hodnot p, g zde uložených podstrčí hodnoty p´= PGPrime a g´= PGGenerator
3. zachytí první nešifrovanou zprávu nebo soubor, který uživatel podepsal s takto podvrženými parametry a uchová si její podpis
4. na základě získané zprávy a jejího podpisu vypočte privátní klíč uživatele (hodnotu x)
5. vrátí hodnoty p, g do původního stavu

Postup útoku na algoritmus DSA si nyní popíšeme podrobně a konkrétně tak, jak jsme ho provedli s využitím programu PGP™ 7.0.3. Příklady a postup jsou uvedeny pro 1024 bitové DSA. V dalším textu budeme podvržené a na základě nich vypočítané hodnoty označovat vždy čárkou.

1.krok
Zvolili jsme prvočíslo p´ (= PGPrime, konstanta) tak, aby

1. p´ mělo 159 bitů a byla tak určitě splněna podmínka p´ < q.
2. při zápisu p´ ve tvaru $p' = t*2^s + 1$ bylo 2^s co největší číslo a t malé prvočíslo.

Konkrétně jsme vybrali s = 151 a t = 167, tj. p´ má binární tvar 10100111000....(150 nul)....0001 a hexadecimálně je zapsáno jako 0x5380 0000 0000 0000 0000 0000 0000 0000 0000 0001.

Dále jsme zvolili číslo g´ (=PGGenerator, konstanta) tak, aby

1. $1 < g´ < p´ - 1$.
2. g´ bylo generátorem multiplikativní grupy $\mathbf{Z}_{p´}^{*}$.

Konkrétně jsme volili g´ = 0x31AC8529 1FF814E6 25E4B88C 8C5047A7 DB2F0E45 a ověřili jsme, že $(g´)^{(p´ - 1)/2} \bmod p´ \neq 1$ a $(g´)^{(p´ - 1)/t} \bmod p´ \neq 1$.

2.krok
Nyní jsme získali soubor secring.skr a v jeho záznamu Secret Key Packet jsme vyměnili hodnoty (p,g) za hodnoty (p´, g´). Dále jsme upravili délky těchto čísel ve formátu MPI a zkrátili celkovou délku Secret Key Packet (hodnoty na počátku záznamu) tak, aby odpovídala kratším falešným hodnotám p´ a g´. Situaci ilustruje obrázek 1 a 2.

3.krok
S takto podvrženými hodnotami jsme vyčkali, až uživatel podepíše nějaký nám známý soubor (zprávu m) a získali jsme jeho podpis - hodnoty (r´, s´). Nyní označme k nám neznámou hodnotu náhodně voleného čísla, které uživatelův program zvolil při tomto podpisu (viz popis DSA výše).

4.krok
V tomto kroku jsme vypočítali hodnotu privátního klíče x daného uživatele na základě hodnot p´, g´, m, r´ a s´. Z definice hodnoty podpisu (r´, s´) totiž vyplývá, že

(1) $r´ = (g´)^{k} \bmod p´ \bmod q$, což vzhledem k volbě $p´ < q$ dává

(1a) $r´ = (g´)^{k} \bmod p´$

a

(2) $s´ = \{[k^{-1} \bmod q] * [h(m) + \mathbf{x}*r´] \} \bmod q$, tedy

(2a) $\mathbf{x} = \{ [s´* k - h(m)] * [(r´)^{-1} \bmod q] \} \bmod q$.

Klíčovým bodem nyní je, že neznámé náhodně volené číslo k umíme díky volbě PGPrime a PGGenerator vypočítat. Prvočíslo p´= PGPrime bylo zvoleno tak, aby rovnice (1a), tj. úloha diskrétního logaritmu v $Z_{p´}^{*}$ byla snadno řešitelná. Konkrétní postup určení diskrétního logaritmu v této speciální grupě je uveden samostatně v dodatku 1. Na základě tohoto postupu jsme pak z rovnice (1a) určili hodnotu k a z rovnice (2a) dopočítali hodnotu **x**.

Správnost **x** jsme zkontrolovali podle vztahu $y = g^{\mathbf{x}} \bmod p$ s originálními hodnotami y, g, p. Hodnota x je tedy vypočtena a její platnost je ověřena proti hodnotě veřejného klíče.

5.krok
Uživateli jsme vrátili jeho původní soubor secring.skr.

1.1 Praktická realizace útoku

Útok byl aplikován na nejnovější verzi programu PGP™ v. 7.0.3 pro Windows 95/98/NT/2000. Z povahy útoku a využitých formátů dat (Public Key Packet verze 3 i 4, Secret Key Packet verze 3 a 4) vyplývá, že by tento útok měl být úspěšný i na jiných platformách. Postup útoku byl zvolen přesně podle výše uvedeného popisu. Úprava parametrů byla provedena v poli Secret Key Packet v souboru secring.skr, kde je uložen privátní podpisový klíč, jak můžeme vidět na obrázku 1 a 2. Na prvním jsou vyznačeny původní hodnoty a na druhém hodnoty podvržené.

95**01D7**043A8D29DF11**0400**F2E02A396A14E137085DA859B3569AF4027EA379682F46780920B
72127C88787DDC1BFF9FDB59E564B741FD5FC98856679F1C041CB71895CB6975E7FE6E15A6D
4B70514560E11A25637F3FBA35E89E5F1FA272A2707F4865EA106EE402973D4969A276DA491
1005B968B81548621CEBBB5771A35C5A785F7F480E47277D2BAB500A0FF04303152BD2A9AD9
63E063A3FE34A8A5534F3F**0400**CD8580F20AA821A6D2FF5255DFD02E4F4C8D8DA3731517476
BEE096F7B104B01B6CE1C4DE586BAEA30D82B50DCB3F0D20B0F0D07D8384C09F12CBF079887
CEB696E822D753A48584F2BC573C84E8490AB310FDBCC40EAEBCD05973B3F2A1A479FFE0E4B
63026E066B6E936F1B2B7F1C91C65CBA0F27B4C0D22254BBC852DEDE10400AC756BB6EB8231
3A0FE91F47A36D1425D89FB124CD0ACBA082E8B2C2B048BE92C5CE7A5FAA5AF317DCC086150
B98AB504C0DA6BF1D87FAB73C8F8D0FC821BD8902CA6927338CF0D682E7C9E3E8D89A3D00D5
3224C301E6C932ADA7562FA15E9027E105F803043D4CBC08807A8FB71FEF9B27EE6A0722C4B
F601D032CC59F6FE4FF09030213CF38106B7BCA3F603F59437C3860B98DA3A1A3F02A4D2754
075B494CAC156E38E1282705FB0BBD68940A1653457E161AB00187B428566C617374696D696
C204B6C696D612C20445353203C762E6B6C696D6140646563726F732E637A3EB00303FFFF89
005A04101102001A05023A8D29DF050901E13380040B09080702190105 1B03000000000A091
09B89D5F084A0EAD7A35C009F5B643D5D2C37F4B2CFC9F399873B747CB3FBFE6800A0A8FE0A
2498E332586F6BED3BB88F278B0C5CF079B001679D014E043A8D2A78100400C904D0246F862
2352D6A60F67F1B4AAA4E94562BB00595A67DDB853AEF3F421CEE2D5FCD5AF18180872FE502
960093815901046096792 74CC92770C6DEBCF391A39B92270D71E7C5EBEC66B3A3BF1BDF521
7E9F609F5D011B9D648A930998C61CD462F3BAAAFD916FDBFFFAA01FCBD2E42F1BC5C406BB0
763B3D4830240841390002020 3FE2B39B802893DE670D745D2AE4DF802BDA707E829B7B0FF7
438FFB88EAB76189AA90A143FB11C1DCC5149046C913AB114D9775563BA0103E65C951D6AC9
199D52818BD2B8A8BF07A6E9F8C242811FF9522EE168207F1EC5D49B441C63D473F7C83D89C
3B6F43A3D80B1B38F7195DE45A55807207159A70CA883493532CD4D8802FF09030221D3B8EC
1276C0D3601B745E982D01201DA87DB47FD3B9C8CDD8F6BC857F56B6F4370AB8A94C7A4E528
D209A80B365A416AF80E1198BAC1AE4175A0F90B00187890052041811020012050 23A8D2A78
050901E13380051B0C000000000A09109B89D5F084A0EAD7CD04009F7056BA18F5907E36E4E
D9A79B4160AE8C6338D98009B043443B6665E860719768B49382DEC95FD7F96BBB00167

Obrázek.1: soubor secring.skr hexadecimálně, vyznačeny jsou původní hodnoty p,g, jejich délky a celková délka záznamu

95**00FF**043A8D29DF11**009F**5380000000000000000000000000000000000000100A0FF0430315
2BD2A9AD963E063A3FE34A8A5534F3F**009E**31AC85291FF814E625E4B88C8C5047A7DB2F0E45
0400AC756BB6EB82313A0FE91F47A36D1425D89FB124CD0ACBA082E8B2C2B048BE92C5CE7A5
FAA5AF317DCC086150B98AB504C0DA6BF1D87FAB73C8F8D0FC821BD8902CA6927338CF0D682
E7C9E3E8D89A3D00D53224C301E6C932ADA7562FA15E9027E105F803043D4CBC08807A8FB71
FEF9B27EE6A0722C4BF601D032CC59F6FE4FF09030213CF38106B7BCA3F603F59437C3860B9
8DA3A1A3F02A4D2754075B494CAC156E38E1282705FB0BBD68940A1653457E161AB00187B42
8566C617374696D696C204B6C696D612C20445353203C762E6B6C696D6140646563726F732E
637A3EB00303FFFF89005A04101102001A05023A8D29DF050901E13380040B0908070219010
51B03000000000A09109B89D5F084A0EAD7A35C009F5B643D5D2C37F4B2CFC9F399873B747C
B3FBFE6800A0A8FE0A2498E332586F6BED3BB88F278B0C5CF079B001679D014E043A8D2A781
00400C904D0246F8622352D6A60F67F1B4AAA4E94562BB00595A67DDB853AEF3F421CEE2D5F
CD5AF18180872FE5029600938159010460967927 4CC92770C6DEBCF391A39B92270D71E7C5E
BEC66B3A3BF1BDF5217E9F609F5D011B9D648A930998C61CD462F3BAAAFD916FDBFFFAA01FC
BD2E42F1BC5C406BB0763B3D4830240841390002020 3FE2B39B802893DE670D745D2AE4DF80
2BDA707E829B7B0FF7438FFB88EAB76189AA90A143FB11C1DCC5149046C913AB114D9775563
BA0103E65C951D6AC9199D52818BD2B8A8BF07A6E9F8C242811FF9522EE168207F1EC5D49B4
41C63D473F7C83D89C3B6F43A3D80B1B38F7195DE45A55807207159A70CA883493532CD4D88
02FF09030221D3B8EC1276C0D3601B745E982D01201DA87DB47FD3B9C8CDD8F6BC857F56B6F
4370AB8A94C7A4E528D209A80B365A416AF80E1198BAC1AE4175A0F90B00187890052041811
02001205023A8D2A78050901E13380051B0C000000000A09109B89D5F084A0EAD7CD04009F7
056BA18F5907E36E4ED9A79B4160AE8C6338D98009B043443B6665E860719768B49382DEC95
FD7F96BBB00167

Obrázek.2: soubor secring.skr po narušení, vyznačeny jsou změněné hodnoty p´,g´, jejich délky a celková délka záznamu

5 Útok na podpisový algoritmus RSA v OpenPGP

5.1 Stručný popis RSA

Zde si krátce připomeneme definici algoritmu RSA (viz např. [6]) a zavedeme označení proměnných. Označme n modul RSA a nechť n = p*q, kde p, q jsou prvočísla. Označme e veřejný exponent a d privátní exponent. Dále označme pInv = p^{-1} mod q. Veřejným klíčem označujeme dvojici (n,e). Pro potřeby formátu OpenPGP se za privátní klíč považuje čtveřice (d, p, q, pInv). Podpis zprávy m je vytvořen jako hodnota s = m^d mod n, přičemž se předpokládá, že zpráva m už je určitým způsobem předem zformátována. Podpis s zprávy m je platný, pokud platí m = s^e mod n.

5.2 Popis útoku na podpisový klíč RSA

Předesíláme, že následující úvahy platí pro podpisový klíč algoritmu RSA a formát OpenPGP, nikoli pro programy PGP™. Programy PGP™ mají totiž zabudovány kontrolní mechanismy na integritu tohoto klíče před jeho použitím k podpisu. V tomto případě tedy útočíme pouze na formát OpenPGP.
Jako v případě DSA, i zde je privátní klíč uložen ve struktuře Secret Key Packet. V současné době se používají verze 3 a verze 4 tohoto formátu, které se liší v tom, jakým způsobem jsou šifrována privátní data. Obě verze Secret Key Packet obsahují na začátku údaje o veřejném klíči ve struktuře Public Key Packet a za nimi následují data privátního klíče. Struktura Public Key Packet má také dvě verze formátu (3 a 4), ale ty se liší jen jedním časovým údajem. Proto si uvedeme jen obsah novější verze 4 (viz tabulka 2).

Struktura Public Key Packet obsahuje:

1. 1 oktet číslo verze
2. 4 oktetové číslo, označující čas, kdy byl klíč vytvořen
3. 1 oktet identifikující asymetrický algoritmus (zde RSA), patřící k tomuto klíči
4. Řadu celých čísel ve tvaru MPI, obsahující veřejný klíč. Zde je to :

- číslo n (modul RSA) ve tvaru MPI,
- číslo e (veřejný exponent RSA) ve tvaru MPI.

Za Public Key Packet následují další údaje Secret Key Packet, přičemž data privátního klíče jsou již šifrována, jak ukazuje tab.2. U RSA se jedná se o tyto údaje:

- privátní exponent d
- prvočíslo p
- prvočíslo q (p < q)
- pInv (= p^{-1} mod q)
- dva oktety kontrolního součtu checksum.

<table>
<tr><td rowspan="6">Public Key Packet</td><td>1 oktet označující číslo verze</td><td rowspan="10" colspan="2">Oblast veřejně přístupných dat</td></tr>
<tr><td>4 oktety označující čas, kdy byl klíč vytvořen</td></tr>
<tr><td>(ve verzi 3 Public Key Packet navíc 2 oktety, označující počet dnů platnosti klíče)</td></tr>
<tr><td>1 oktet označující algoritmus RSA</td></tr>
<tr><td>číslo n (modul RSA) ve formátu MPI</td></tr>
<tr><td>číslo e (veřejný exponent RSA) ve formátu MPI</td></tr>
<tr><td colspan="2">1 oktet (string-to-key usage)</td></tr>
<tr><td colspan="2">[Volitelné] V případě, že string-to-key usage je 0xFF, je zde 1 oktet, identifikující symetrický šifrovací algoritmus pro ochranu privátního klíče (viz tab. 1).</td></tr>
<tr><td colspan="2">[Volitelné] V případě, že string-to-key usage je 0xFF, je zde "string-to-key specifier" (viz tab. 1).</td></tr>
<tr><td colspan="2">[Volitelné] Jestliže privátní klíč je šifrován, je zde uložen inicializační vektor (IV). Jsou to náhodná data v délce bloku použité blokové šifry (8 oktetů pro 64bitové blokové šifry, 16 oktetů pro algoritmus AES).</td></tr>
<tr><td>Algoritmicky závislá čísla ve formátu MPI. Obsah pro RSA (modul 1024 bitů):</td><td>verze 3</td><td>verze 4</td></tr>
<tr><td>2 oktety, prefix čísla d</td><td>otevřeně</td><td rowspan="10">šifrováno</td></tr>
<tr><td>128 oktetů, číslo d</td><td>šifrováno</td></tr>
<tr><td>2 oktety, prefix čísla p</td><td>otevřeně</td></tr>
<tr><td>64 oktetů, číslo p</td><td>šifrováno</td></tr>
<tr><td>2 oktety, prefix čísla q</td><td>otevřeně</td></tr>
<tr><td>64 oktetů, číslo p</td><td>šifrováno</td></tr>
<tr><td>2 oktety, prefix čísla pInv</td><td>otevřeně</td></tr>
<tr><td>64 oktetů, číslo pInv</td><td>šifrováno</td></tr>
<tr><td>2 oktety (H_{Sum} , L_{Sum}) checksum, součet předchozích položek (čísel MPI) v otevřeném tvaru modulo 65536</td><td>otevřeně</td></tr>
</table>

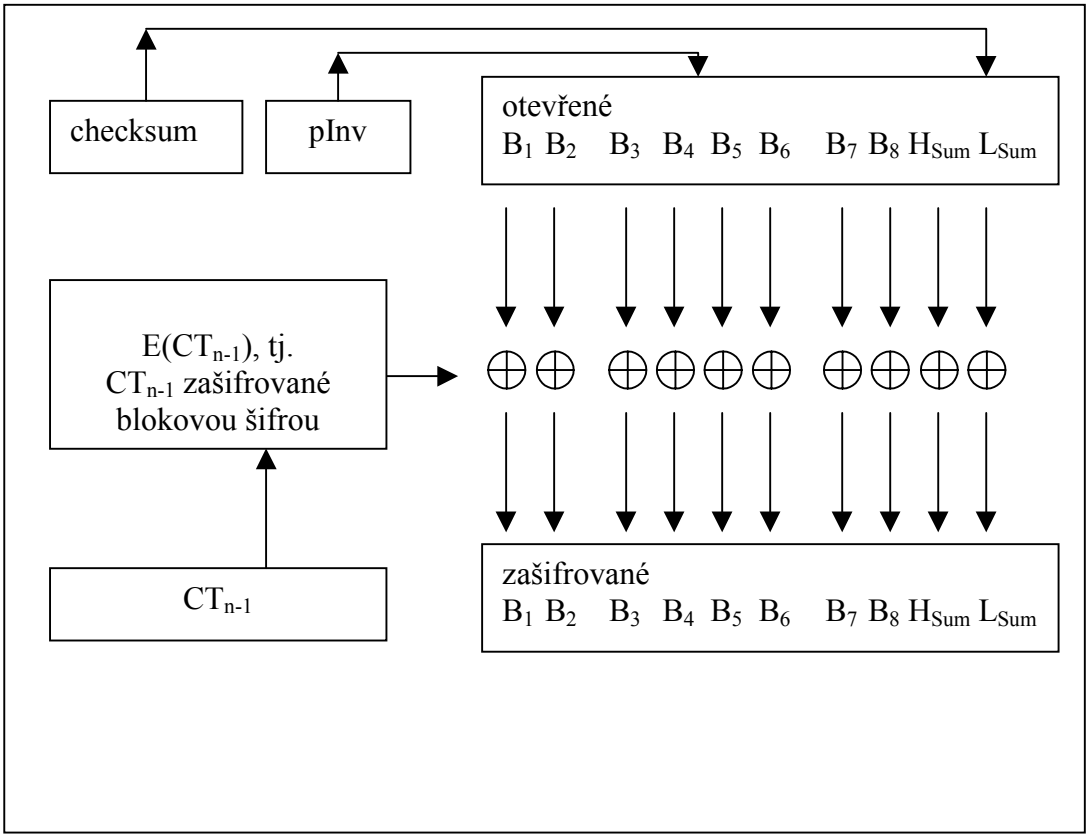
Tabulka 2: Obsah struktury Secret Key Packet verze 3 a 4 pro algoritmus RSA (modul 1024 bitů)

V tabulce 2 je uvedeno, jak se liší šifrování privátních údajů ve verzi 3 a verzi 4 Secret Key Packet. Oba formáty budeme zkoumat zvlášť. Společným prvkem obou dvou formátů je výpočet kontrolního součtu checksum jako prostého aritmetického součtu jednotlivých bajtů privátních dat modulo 65536: checksum = $(d_1 + d_2 + ...+ d_n)$ mod 65536. U obou verzí je k šifrování použita uživatelem zvolená symetrická bloková šifra v tzv. specifickém (PGP^{TM}) modu CFB.
Zvláštností verze 3 formátu Secret key Packet je, že nejsou šifrovány prefixy čísel MPI, tvořících privátní klíč, ani checksum. Navíc na počátku každého MPI je stav CFB resynchronizován tak, že nový blok začíná až od nové hodnoty MPI.
U formátu Secret Key Packet verze 4 jsou šifrována všechna privátní MPI včetně prefixů, kontrolního součtu a bez resynchronizace.

1.3 Útok na verzi 3 formátu Secret Key Packet (RSA)

Útok, který je možno vést na verzi 3 formátu spočívá v tom, že je možné měnit délku jednotlivých MPI, tj. prefixy MPI, protože nejsou šifrovány a není šifrován ani checksum.

Například je možné snížit prefix MPI čísla pInv o jedničku a současně snížit i kontrolní součet o jedničku. Platnost kontrolního součtu záznamu Secret Key Packet (s novým pInv´) zůstane zachována (sčítají se všechny oktety bez ohledu na to, zda je v nich 1 nebo 8 platných bitů), ale hodnota privátní informace pInv se změní, neboť toto číslo se zkrátí o 1 bit. Obdobně lze měnit hodnoty ostatních privátních čísel d, p, q. Jakmile útočník získá zprávu a její podpis, který bude vytvořen pomocí privátního klíče (d, p, q, pInv´), umožní mu to vypočítat celý privátní klíč RSA. Podrobný postup je uveden v dodatku 2. V tomto smyslu je situace zcela analogická útoku proti DSA.



Obrázek 1: Způsob šifrování posledního bloku dat privátního klíče v Secret Key Packet, obsahujícího privátní hodnotu pInv.

1.4 Útok na verzi 4 formátu Secret Key Packet (RSA)

Při útoku na verzi 4 formátu Secret Key Packet nelze předešlý postup využít přímo, protože prefixy i checksum jsou šifrovány, ale lze je modifikovat. Modifikace spočívá v tom, že se využije přímo vlastnosti CFB při šifrování posledního bloku otevřených dat. Pokud například použijeme blokovou šifru s délkou bloku 16 oktetů (tj. v případě AES) a modul RSA 1024 bitů, bude v posledním neúplném bloku šifrového textu osm posledních oktetů čísla pInv (označme je B_1, B_2,..., B_8) a dva oktety kontrolního součtu checksum (označme je H_{Sum}, L_{Sum}). Tato otevřená data budou zašifrována prostou operací XOR s klíčovým materiálem (zašifrovaný předchozí šifrový blok). Pokud v posledním bloku šifrového textu provedeme změnu typu "XOR CONST"
, projeví se to přesně jako "XOR CONST" po odšifrování v otevřeném textu. Cílem je provést změnu v pInv (přesněji v posledních osmi oktetech čísla pInv) a současně v checksum (v oktetu L_{Sum}) na pInv´ a checksum´ tak, aby kontrola integrity po odšifrování souhlasila.

Výsledkem opět bude narušená hodnota pInv, přičemž formát Secret Key Packet bude mít checksum v pořádku. Využití tohoto podvrhnutého pInv´ je stejné jako v předchozím případě. Získáme podpis nějaké zprávy s tímto podvrženým klíčem a odtud vypočítáme hodnotu privátního klíče RSA. Nyní si ukážeme, jak je možné změnit jednotlivé bity oktetů B_1 až B_8 a L_{Sum} tak, aby kontrola integrity privátního klíče po jejich změně souhlasila. Změny jednotlivých bitů uvedených oktetů budeme provádět na šifrovém textu a jak jsme už uvedli, vzhledem k modu CFB se tyto změny budou promítat stejným způsobem do odpovídajících bitů oktetů otevřeného textu. Nyní předpokládejme, že v otevřeném tvaru některého oktetu B_i z množiny $\{B_1, B_2,..., B_8\}$ je nastaven některý j-tý bit (kde j může být 0 až 7) *stejně* jako j-tý bit v oktetu L_{Sum}. Potom stačí v šifrovaném oktetu B_i a současně v šifrovaném oktetu L_{Sum} tento j-tý bit změnit a kontrola integrity bude souhlasit. Pokud by totiž tento bit byl 1, oktet B_i se změní na oktet B_i - 2**j a oktet L_{Sum} se změní na L_{Sum} - 2**j. Nový kontrolní součet bude proto platný! Podobně v případě, že j-tý bit oktetu B_i a L_{Sum} byl 0, oktet B_i se změní na B_i + 2**j a oktet L_{Sum} na L_{Sum} + 2**j. Nový kontrolní součet bude opět platný! Protože nevíme, zda j-tý bit vybraného oktetu B_i je a nebo není stejný jako j-tý bit L_{Sum}, budeme to zkoušet. Můžeme měnit celkem 64 bitů a pravděpodobnost, že neuspějeme, je velmi nízká (2^{-64}). Poznamenejme, že o úspěchu uvedené změny se dozvíme poté, až uživatel zkusí tímto podvrženým klíčem podepsat nějakou zprávu. Nicméně by v průměru dva pokusy měly stačit. Další metodou je měnit j-tý bit současně vždy ve dvou libovolných oktetech z množiny $\{B_1,B_2,..., B_8\}$, zatímco L_{Sum} ponecháme nezměněno. Tentokrát čekáme na situaci, kdy tyto bity budou v otevřeném textu různé. Jejich současnou změnou se jejich vliv v kontrolním součtu anuluje. Podobně můžeme provádět variace se čtveřicí nebo osmicí j-tých bitů. Výsledkem této změny je narušení pInv se stejnými důsledky jako v předchozích případech, tj. zjištění privátního klíče RSA.

6 Útok na privátní klíče po jejich exportu

Dále je nutné poznamenat, že kromě privátní klíčenky secring.skr lze stejným způsobem útočit i na privátní klíč, který je exportovaný do souboru typu "ASCII Key File" a přenášený potom prostřednictvím sítě nebo na disketě. Tento soubor má kromě přídavného kódování stejný obsah jako soubor secring.skr, a proto na něj lze uplatnit stejný útok, jako na soubor secring.skr. To znamená, že přenos privátního klíče prostřednictvím tohoto souboru sítí nebo na disketě není bezpečný.

7 Protiopatření

7.1 Základní přechodná protiopatření

Hlavní příčinou právě prezentovaných útoků je nedostatečná kontrola integrity veřejných i privátních dat v souboru, obsahujícím privátní klíč uživatele. Jako logické protiopatření odtud vyplývá nutnost zavedení lepší kontroly integrity uložených záznamů. Zdůrazňujeme, že tato kontrola musí zajišťovat i integritu veřejných hodnot, které nemusí být nutně šifrovány.

Požadavek na zavedení kvalitní kontroly integrity nemusí být snadné realizovat v krátkém časovém horizontu. Do doby, než dojde k úpravě formátu OpenPGP [1] pro záznamy privátních klíčů (Secret Key Packet), je možné v programech PGP™ a dalších, které implementují formát OpenPGP, přechodně využít alespoň následující kontrolní testy. Ty jsou navrženy tak, aby klíče pro algoritmy DSA a RSA, které splní níže uvedené vztahy, neumožňovaly provedení námi popsaného útoku. Předpokládá se, že tento test bude prováděn jako dodatečná kontrola integrity po přečtení příslušných parametrů ze souboru s privátním

klíčem. K operaci vlastního podpisu přitom smí být použit jen takový klíč, jehož hodnoty tímto testem projdou. Zdůrazňujeme, že uvedený test nemá za úkol nahradit chybějící kontrolu integrity souboru s privátním klíčem, ale má sloužit pouze jako dočasné opatření, které brání zde uvedenému útoku.

7.2 Přechodný test pro DSA

Navrhujeme tento přechodný test pro DSA. Měly by se ověřit následující vztahy:

1. p, q, g, x, y > 0
2. p je liché, q je liché
3. $2^{159} < q < 2^{160}$
4. 1 < g < p
5. 1 < y < p
6. x < q
7. q | (p-1)
8. $g^q \bmod p = 1$
9. $g^x \bmod p = y$

Poznamenejme ještě, že zatímco do takového druhu testů, jaký jsme si právě uvedli, se například u RSA vkládá poměrně velká důvěra, v případě DSA musíme být velmi opatrní. Narozdíl od RSA je zde totiž pouze jedna hodnota (privátní klíč x), kterou útočník nezná. Ostatní parametry jsou pro něho známé a může je libovolně měnit. To je důvod, proč tento test považujeme pouze za dočasné řešení, které musí být co nejdříve nahrazeno jiným druhem kontroly integrity diskutovaných záznamů, viz dále.

1.3 Přechodný test pro RSA

Navrhujeme tento přechodný test pro RSA. Měly by se ověřit následující vztahy:

1. e*d mod (p - 1) = 1
2. e*d mod (q - 1) = 1
3. pInv * p (mod q) = 1
4. n (ze záznamu veřejného klíče) = p*q
5. e ∈ E, kde E je množina možných hodnot, plánovaných pro e, tj. pro PGP™ například {17, 65537,...}

Poznamenejme, že v programu PGP™ jsou kontroly 1 až 4 a některé další implementovány. Ve formátu OpenPGP však tyto kontroly uvažovány nejsou, což je zásadní chyba.

1.4 Další náměty pro formát OpenPGP

Zde uvádíme některé další náměty, které nás napadly po prvním seznámení s formátem OpenPGP (záznam Secret Key Packet) a s programem PGP™. Tyto náměty by rozhodně přispěly ke zvýšení bezpečnosti formátu i programu PGP™. Je však třeba je chápat spíše jako ideová doporučení. Před realizací konkrétních úprav je třeba tyto úpravy podrobit alespoň základní samostatné analýze. Analýza celého formátu OpenPGP je však mnohem obtížnější a jde za rámec tohoto příspěvku.
Navrhovaná opatření jsou:

1. Modus šifrování CFB nahradit modem CBC
   - ztíží popsaný útok na poslední blok šifrovaných privátních dat
2. Kontrolní součet checksum (suma bajtů modulo 65536) nahradit HMAC, založeným na SHA-1 nebo na jiné bezpečné hašovací funkci (například SHA-256, 384, 512 a pod.)
   - znesnadní útoky na chráněná data a současně na zabezpečovací kód

3. Nový kontrolní součet (HMAC):
   a) ukládat v délce minimálně 160 bitů,
   - znesnadní útoky využívající narozeninový paradox,
   b) vypočítávat ho ze všech dat záznamu Secret Key Packet (nejen z privátních, ale i z veřejných),
      - znesnadní integritní útoky na veřejné i privátní části klíče v Secret Key Packet
   c) klíč, použitý v HMAC, derivovat z passphrase jiným způsobem, než klíč pro symetrickou šifru
   - znesnadní útok na HMAC
   d) šifrovat výsledný HMAC společně s privátními daty symetrickou šifrou podobně jako je to v případě checksum ve verzi 4 formátu Secret Key Packet
      - znesnadní integritní útoky na veřejné i privátní části klíče v Secret Key Packet
4. Pro podpisové schéma RSA používat formát typu EMSA-PSS, který je uveden v [6]
      - znemožní řadu útoků, včetně útoku z dodatku 2.

8 Důsledky

Předvedené typy útoků mají značný dopad na bezpečnost programů, využívajících formát OpenPGP (například samotný program PGP™). Kdokoliv, kdo dokáže popsaným způsobem změnit soubor s privátním klíčem, je schopen na základě jediného chybného podpisu získat hodnotu privátního klíče u algoritmů DSA a RSA. K této změně přitom zdaleka nemusí dojít jen na pracovní stanici napadeného uživatele. Citlivým místem systému jsou také soubory s exportovanými privátními klíči, které uživatel používá k přenosu svých privátních klíčů mezi různými stanicemi. Fakt, že privátní klíč je v těchto souborech uložen v zašifrovaném tvaru, může vzbuzovat falešný pocit bezpečí. Dostane-li se však k takové disketě při její přepravě útočník, je bezpečnost uživatelova privátního klíče vážně ohrožena.

Jiný, v praxi velmi efektivní scénář pro použití popsaného útoku, je možné použít v případě, kdy je soubor s privátním klíčem uložen na sdíleném zařízení. Zde může být útočníkem například správce serveru, který na určitou dobu uživateli podstrčí upravenou verzi tohoto souboru, počká, až jej uživatel použije k podpisu (dobu lze poměrně přesně určit monitorováním síťové aktivity uživatelovy stanice) a poté vrátí zpět jeho původní obsah. Z vygenerovaného podpisu potom získá hodnotu privátního klíče. Při dostatečné kontrole nad celým systémem může navíc tento útočník zahladit stopy po útoku tím, že do systému místo zprávy vybavené neplatným podpisem vyšle zprávu, jejíž podpis je platný. To může snadno udělat, neboť zná jak samotnou zprávu, tak i příslušný privátní klíč.

Samotného uživatele programů na bázi OpenPGP staví existence popsaného útoku do nelehké situace v okamžiku, kdy zjistí, že byla vygenerována chybná hodnota podpisu. V takové situaci může být právem na rozpacích, zda se jedná o důsledek záměrného útoku nebo „jen“ o technické selhání. Prakticky je zřejmé, že každý soubor s neplatným podpisem si zasluhuje stejnou pozornost, jako by se jednalo o soubor obsahující privátní klíč v otevřeném tvaru! To zahrnuje zejména odpovídající péči věnovanou jeho neobnovitelnému odstranění z příslušné stanice nebo dokonce serveru.

9 Závěr

Zde popsané útoky, vedoucí k odhalení nejcitlivějších informací systému (privátních podepisovacích klíčů algoritmů RSA a DSA), velmi názorným způsobem poukazují na důležitý aspekt ochrany privátních klíčů a veřejných parametrů asymetrických algoritmů v bezpečnostních systémech. Poznamenejme ještě, že to byla právě výzkumná práce týkající se obecných problémů a principů v této oblasti, při které jsme se podívali, "jak to dělá program PGP™", aniž bychom se jím chtěli primárně zabývat.
Námi prováděná analýza vycházela z obecné dokumentace OpenPGP [1]. Odhalili jsme v ní závažné nedostatky, které mohou vyústit ve snadnou zranitelnost aplikací vytvořených podle ní. Praktickým příkladem je program PGP™, který sice v případě RSA vykazuje díky dodatečným ochranám nad rámec OpenPGP odolnost vůči útokům na RSA, avšak je snadno zranitelný pomocí útoku na podpisový algoritmus DSA.

Připomínáme, že ačkoliv jsme se zde s ohledem na OpenPGP omezili na algoritmy RSA a DSA, lze za předpokladu nedostatečné ochrany privátních klíčů a veřejných parametrů očekávat obdobnou zranitelnost i u dalších asymetrických kryptosystémů včetně systémů na bázi eliptických křivek. Rovněž formát OpenPGP a potažmo s ním program PGP™ nebude patrně jediným případem, kdy díky nesprávné ochraně zmíněných parametrů může dojít k napadení daného systému. Celý tento dokument tak chce důrazně apelovat na pozornost při návrhu způsobu zacházení s uvedenými hodnotami a jejich uložení v příslušném systému.

Poděkování

Touto cestou bychom rádi poděkovali Ing. Pavlu Rydlovi za technickou spolupráci na praktické realizaci zde popsaného útoku.

Literatura

[1] RFC 2440: *OpenPGP Message Format*, J. Callas, Network Associates, L. Donnerhacke, IN-Root-CA Individual Network e.V., H. Finney, Network Associates, R. Thayer, EIS Corporation, November 1998

[2] Menezes A.J., Oorschot P.C., Vanstone S.A.: *Handbook of Applied Cryptography*, CRC Press, 1997

[3] Lenstra, A. K.: *Memo on RSA signature generation in the presence of faults*, manuscript, Sept. 28, 1996. Available from the author.

[4] Rosa, T.: *Future Cryptography: Standards Are Not Enough*, zasláno do konference CATE 2001, 2001.

[5] Rosen, K. H., Michels, J. G., Gross, J. L., Grossman, J. W., Shier, D. R.: *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, 2000.

[6] PKCS#1 verze 2.1: RSA Cryptography Standard, RSA Laboratories, Draft 1, September 17, 1999

Dodatky

Dodatek 1: Získání hodnoty privátního klíče pomocí změny veřejných parametrů DSA

Předpoklad P1: ***Způsob výpočtu podpisu DSA***
Mějme dány parametry DSA (p, q, g, y, x), kde p, q jsou prvočísla, $g \in Z_p^*$, ord(g) = q, $y = g^x \bmod p$, $0 < x < q$, x je privátní klíč signatáře. Výpočet podpisu pro zprávu s hašovacím kódem h je pak následující:

1. *náhodně zvolíme k, $0 < k < q$*
2. *vypočteme $r = (g^k \bmod p) \bmod q$*
3. *vypočteme $s = k^{-1}(h + xr) \bmod q$, kde $k*k^{-1} \equiv 1 \pmod q$*
4. *podpisem (výstupem podepisovacího algoritmu) je dvojice (r, s)*

Dále popsaný útok předpokládá, že útočník změní parametry DSA (p, q, g, y, x) na (p', q, g', y, x), kde p' je 159 bitové prvočíslo, $2^{158} < p' < 2^{159}$ a g' je generátor grupy $Z_{p'}^*$ tak, že umí pro všechna $r \in Z_{p'}^*$ snadno určit hodnotu w, $0 \le w < (p-1)$, takovou, že $(g')^w \equiv r \pmod p$, neboli $w = \log_{(g')} r$. Umí tedy snadno řešit úlohu diskrétního logaritmu v $Z_{p'}^*$.

V následujících dvou krocích ukážeme, že na základě znalosti hodnoty hašovacího kódu h a podpisu (r, s), který byl pořízen algoritmem DSA s podvrženými parametry (p', q, g', y, x), lze pak snadno určit hodnotu privátního klíče x.

Krok 1: určení množiny K
V tomto kroku budeme pracovat s hodnotou r. Z rovnice *(P1.2)* vyplývá, že $r = (g')^k \bmod p' \bmod q = (g')^k \bmod p'$, neboť $p' < q$.

Dle výše uvedeného předpokladu umí útočník pro libovolné r snadno určit hodnotu $w = \log_{(g')} r$, takže pro neznámou hodnotu k máme: $k = w + b(p'-1)$, kde $b \in Z$, $b \ge 0$. Z první *(P1.1)* máme pro k ještě podmínku $0 < k < q$. Protože p' je 159 bitové a q 160 bitové, dostáváme pro k množinu přípustných hodnot $K = \{ w + b(p'-1) : b(p'-1) < q-w, 0 \le b \le 3 \}$. Dostali jsme tak množinu o nejvýše čtyřech možných hodnotách k_i, mezi nimiž s jistotou leží hledané k.

Krok 2: určení hodnoty x
Nyní budeme postupně vybírat $k_i \in K$ a z *(P1.3)* určovat hodnoty x_i jako $x_i = r^{-1}(k_i*s - h) \bmod q$, kde $r*r^{-1} \equiv 1 \pmod q$. Poznamenejme, že $\gcd(r,q) = 1$, takže hodnota r^{-1} existuje a je jednoznačná. Takto obdržíme množinu hodnot $X = \{x_i : k_i \in K\}$, v níž leží i hledaná hodnota privátního klíče x.

Nyní zbývá z množiny X vybrat hledanou hodnotu x. To lze snadno provést pomocí vztahu $y = (g^{xi} \bmod p)$, kde p a g jsou původní veřejné parametry DSA a y je veřejný klíč.Vyzkoušením nejvýše čtyř různých hodnot x_i tak odstraníme zbývající neurčitost zanesenou nízkou hodnotou p' a získáme hledanou hodnotu x. Poznamenejme, že prvek g má v grupě Z_p^* řád q. Odtud plyne, že existuje pouze jedna hodnota $0 < x_i < q$, pro kterou platí, že $y = (g^{xi} \bmod p)$. Určení hodnoty x uvedeným postupem je proto jednoznačné.

Poznámka. *Z předchozího popisu je dobře patrný hlavní princip celého útoku. Ten spočívá v takové modifikaci veřejných parametrů DSA, díky které se při výpočtu podpisu vytvářejí velmi slabé instance problému diskrétního logaritmu. Místo řešení tohoto problému v multiplikativní cyklické podgrupě grupy Z_p^* o 160 bitovém řádu postačí tento problém řešit pouze v cyklické grupě $Z_{p'}^*$, kde p' je 159 bitové prvočíslo s vhodně volenou strukturou.*

Složitost takových instancí problému diskrétního logaritmu je z kryptologického hlediska považována za naprosto nedostatečnou. Pro praktickou realizaci popsaného útoku byla navíc nalezena univerzálně použitelná grupa $Z_{p'}^*$ se speciální strukturou, která umožňuje velmi rychlé řešení vznikajících instancí zmíněného problému i na běžném kancelářském PC, viz popis algoritmu A1 dále.

Algoritmus A1: Výpočet hodnoty $w = \log_g r$ pro speciální druh Z_p^*.

V následující části popíšeme efektivní algoritmus pro výpočet hodnoty diskrétního logaritmu, který je možné použít pro multiplikativní grupy Z_p^* s určitou speciální strukturou (p je zde rovno podvržené hodnotě p'). Předpokládá se, že univerzálně vybraná grupa o této struktuře bude použita pro praktickou realizaci výše popsaného druhu útoku vůči DSA. Strukturu zmíněné grupy si uvedeme ve formě následujícího předpokladu.

Předpoklad P2. *Mějme multiplikativní grupu Z_p^*, kde p je prvočíslo ve tvaru $p = t*2^s + 1$ a t je prvočíslo. Dále buď generátorem Z_p^*. Následující postup ukazuje způsob výpočtu hodnoty w, který je efektivní pro malá t (pro praktickou realizaci předchozího útoku byla nalezena univerzálně použitelná grupa s parametry* ***t = 167, s = 151****).*

Předtím, než se pustíme do popisu jednotlivých kroků vlastního algoritmu, uvedeme několik užitečných formalizmů, které se budou později hodit při výkladu jednotlivých operací.

Definice D1. [viz 5, str. 277] *Buďte p a k celá kladná čísla. Potom číslo b s vlastností gcd(b, p) = 1 nazveme zbytkem k-té mocniny modulo p právě tehdy, když kongruence $x^k \equiv b \pmod p$ má řešení pro nějaké $x \in Z$.* (*V případě k=2 používáme často výraz kvadratický zbytek modulo p.)*

Fakt F1. [viz 5, str. 279] *Buďte p prvočíslo, k celé kladné číslo a b celé číslo takové, že gcd(b, p) = 1. Potom b je zbytek k-té mocniny modulo p právě tehdy, když $b^{(p-1)/d} \equiv 1 \pmod p$, kde d = gcd(k, p-1).*

Lemma L1. *Buďte p prvočíslo a g generátor grupy Z_p^*. Potom hodnota y, $y = g^w \bmod p$, je zbytek k-té mocniny modulo p, kde k|(p-1), právě tehdy, když k|w.*

Důkaz. Je-li y zbytek k-té mocniny modulo p, pak podle faktu F1 $y^{(p-1)/k} \equiv 1 \pmod p$. Z předpokladu $y = g^w \bmod p$ potom dostáváme, že $(g^w)^{(p-1)/k} \equiv 1 \pmod p$. Protože g je generátor grupy Z_p^*, musí platit, že $w*(p-1)/k \equiv 0 \pmod{(p-1)}$. Odtud pak přímo dostáváme, že k|w.

Důkaz implikace v obráceném směru je snadný. Nechť k|w, tj. w=k*b, kde b je celé kladné číslo. Potom $y = g^w \bmod p = (g^b)^k \bmod p$ a přímo z definice D1 plyne, že y je zbytek k-té mocniny modulo p. °

Dále popíšeme postupně tři kroky algoritmu, podle kterých probíhá výpočet hledaného diskrétního logaritmu $w = \log_{(g)}r$. Ve své podstatě se jedná o modifikovanou verzi Pohling-Hellmanova algoritmu (viz [2]), který by na uvedeném druhu multiplikativní grupy byl rovněž velmi efektivní. Ve snaze využít co nejvíce konkrétní strukturu použité grupy jsme se však rozhodli použít následující postup.

Krok 1: určení hodnoty $^{s}w = w \bmod 2^{s}$

Nechť $w = w_n*2^{n-1} + w_{n-1}*2^{n-2} + ...+ w_1$, kde n je počet bitů binárního rozvoje w a $w_i \in \{0,1\}$, pro $1 \leq i \leq n$.

Zabývejme se nyní určením bitu w_1. Je-li tento bit nulový, potom platí $w = 2*b$, pro nějaké celé číslo b. Pro hodnotu $r = g^w \bmod p$ odtud dostáváme, že $r \equiv (g^b)^2 \pmod p$, takže r je kvadratický zbytek modulo p. Pokud je naopak hodnota bitu w_1 jednička, potom je hodnota w lichá a podle lemmatu L1 není r v tomto případě kvadratickým zbytkem modulo p. Na základě tohoto rozboru a faktu F1 můžeme pro w_1 formulovat následující:

- $r^{(p-1)/2} \equiv 1 \pmod p \Rightarrow w_1 = 0$
- $r^{(p-1)/2} \not\equiv 1 \pmod p \Rightarrow w_1 = 1$

Pokračujme nyní určením w_2. Nejprve na základě znalosti w_1 upravíme r na $r_2 = (r*g^{-w1}) \bmod p$. Touto úpravou jsme získali hodnotu $r_2 = g^{w'} \bmod p$, kde $w' = w_n*2^{n-1} + w_{n-1}*2^{n-2} + ...+ w_2*2$. Pokud nyní platí, že $w_2 = 0$, potom pro hodnotu r_2 dostáváme, že $r_2 \equiv (g^b)^4 \pmod p$, kde b je celé číslo. Čili hodnota r_2 je v tomto případě zbytkem 4-té mocniny modulo p.

Pokud ovšem platí, že $w_2 = 1$, potom w' není dělitelné čtyřmi a hodnota r_2 není dle lemmatu L1 zbytkem 4-té mocniny modulo p. Takto můžeme pro w_2 odvodit následující:

- $r_2^{(p-1)/4} \equiv 1 \pmod p \Rightarrow w_2 = 0$
- $r_2^{(p-1)/4} \not\equiv 1 \pmod p \Rightarrow w_2 = 1$

Po určení w_2 opět upravíme r_2 na r_3 jako $r_3 = (r_2*g^{-2*w2} \bmod p)$ a pokračujeme v určování hodnot w_i tak dlouho, dokud platí $2^i \mid (p-1)$.

Protože $(p-1)/2^s = t$, kde t je liché prvočíslo, můžeme takto určit hodnoty w_i pro $1 \leq i \leq s$. Získáme tak binární zápis hodnoty $^{s}w = w_s2^{s-1} + w_{s-1}2^{s-2} + ...+ w_1$, kde $^{s}w = w \bmod 2^s$. Toto byla hodnota, kterou jsme v tomto kroku hledali. Celý postup je vidět na obrázku 3.

Postup výpočtu ${}^s w = \log_g r \bmod 2^s$.

1. Předpokládejme:
   a. binární zápis ${}^s w$ jako ${}^s w = w_s w_{s-1} \ldots w_1$
   b. Z_p^*, kde p je prvočíslo, $p = t*2^s + 1$
2. $i = 1$; $f = g^{p-2} \bmod p$; $v = p-1$
3. $v = v/2$; $y = r^v \bmod p$
4. **if** $(y = 1)$ **then** $w_i = 0$ **else** $w_i = 1$; $r = r*f \bmod p$
5. $f = f^2 \bmod p$
6. $i = i+1$
7. **if** $(i \leq s)$ **then goto** 3
8. **return** ${}^s w = w_s w_{s-1} \ldots w_1$

Obr. 3: Krok 1 algoritmu A1.

Krok 2: určení hodnoty ${}^t w = w \bmod t$

Je snadné dokázat, že pro celé číslo j takové, že $r^{(p-1)/t} \equiv (g^{(p-1)/t})^j \pmod p$, platí, že ${}^t w \equiv j \pmod t$. Protože $j \leq t-1$, tak přímo platí, že ${}^t w = j$. Hodnotu ${}^t w$ v tomto kroku proto nalezneme tak, že budeme postupně zkoušet čísla j, $0 \leq j \leq t-1$, dokud nenajdeme číslo j splňující kongruenci $r^{(p-1)/t} \equiv (g^{(p-1)/t})^j \pmod p$. Takové číslo j pak bude hledanou hodnotou ${}^t w$.

Krok 3: určení hodnoty $w = \log_g r$.

V předchozích krocích jsme obdrželi soustavu následujících kongruencí:

- $w \equiv {}^s w \pmod{2^s}$
- $w \equiv {}^t w \pmod t$

Přitom platí, že $\gcd(t, 2^s) = 1$, takže podle Čínské věty o zbytku (dále CRT – Chinese Remainder Theorem) existuje jednoznačná hodnota $0 \leq w < t*2^s$, která splňuje obě kongruence. Protože hodnota $t*2^s$ je rovněž řádem grupy Z_p^* pro $p = t*2^s+1$, je hodnota w zároveň hledaným diskrétním logaritmem hodnoty r. Dále uvádíme přímo postup vedoucí k určení w:

1. vypočteme $\gamma \equiv (2^s)^{-1} \pmod t$, tato hodnota existuje a je jednoznačná, protože $\gcd(t, 2^s) = 1$
2. vypočtěme $v = ({}^t w - {}^s w)*\gamma \bmod t$
3. $w = {}^s w + v*2^s$

Důkaz (správnosti předchozího postupu):
Pro faktor 2^s je z výrazu pro w přímo vidět, že $w \equiv {}^s w \pmod{2^s}$. Pro faktor t dostáváme, že $w \equiv {}^s w + ({}^t w - {}^s w)(2^s)^{-1}*2^s \pmod t$, takže $w \equiv {}^t w \pmod t$. Navíc $w = {}^s w + v*2^s < 2^s + (t-1)*2^s = t*2^s$. Tím jsme ověřili, že uvedený postup skutečně odpovídá aplikaci CRT na výše uvedenou soustavu kongruencí. °

Výsledky experimentu

Popsaný postup v krocích 1 až 3 byl realizován na různých konfiguracích kancelářských PC. Tabulka 3 ukazuje průměrné doby výpočtu pro náhodně volené hodnoty r. Vidíme, že celý výpočet trvá řádově stovky milisekund.

Konfigurace	Doba výpočtu jednoho diskrétního logaritmu v milisekundách
Pentium III/ 500MHz 128 MB RAM Windows NT 4.0 SP 6a	96
Celeron 400 MHz 128 MB RAM Windows NT 4.0 SP 6a	113
Pentium II 400 MHz 128 MB RAM Windows 2000 Advanced Server	116
Pentium II 300 MHz 128 MB RAM Windows NT 4.0 SP 6a	150
Pentium 166 MHz 96 MB RAM Windows NT 4.0 SP 6a	535
Pentium 75 MHz 46 MB RAM Windows NT 4.0 SP 4	1020

Tabulka 3: Doba výpočtu hodnoty $w = \log_g r$ pro speciální druh Z_p^*.

Dodatek 2: Útok na privátní klíč RSA

V tomto dodatku si stručně nastíníme, jak lze získat hodnotu privátního klíče RSA z hodnoty chybného podpisu, který byl vypočten za použití narušeného privátního klíče. Tento útok vychází z rozboru formátu OpenPGP. Na testovaném programu PGP™ nebyl tento útok přímo aplikovatelný, protože program PGP™ provádí nad rámec definice OpenPGP dodatečnou kontrolu integrity privátního klíče. V případě aplikací, realizovaných přesně podle OpenPGP, však takový útok hrozí a je stejně efektivní, jako výše prezentovaný útok na DSA.

Podle OpenPGP je privátní klíč RSA tvořen následující šesticí hodnot (n, p, q, pInv, e, d), kde p, q jsou prvočísla, $n = p*q$ je veřejný modul, $p*pInv \equiv 1 \pmod q$, e je veřejný exponent a d je privátní exponent, tj. $e*d \equiv 1 \pmod{\text{lcm}(q-1,p-1)}$. Na základě této struktury lze předpokládat, že podepisovací transformace RSA je pro konkrétní hodnotu zformátované zprávy m počítána podle následujícího postupu:

1. $s_1 = m^d \bmod p$
2. $s_2 = m^d \bmod q$
3. $h = pInv*(s_2 - s_1) \bmod q$
4. $s = s_1 + p*h$
5. s je výsledek podepisovací transformace, neboť lze odvodit, že pro hodnotu s vypočítanou podle tohoto postupu platí $s = m^d \bmod n$

Tento postup odpovídá aplikaci Čínské věty o zbytku a umožňuje efektivní výpočet hodnoty podpisové transformace. Jak bylo poprvé ukázáno v [3], je použití této techniky poměrně náchylné k útokům, využívajícím chyby při výpočtu podpisu. Tyto chyby lze přitom zanést nejen ovlivňováním napadeného zařízení během výpočtu podpisu, ale například i narušením

určitých hodnot tvořících privátní klíč. Detailnější pozornost je této problematice věnována v [4]. Zde se zaměříme přímo na jeden konkrétní typ útoku, který přichází v úvahu u OpenPGP.

Předpokládejme, že útočník naruší parametry RSA (n, p, q, pInv, e, d) tak, že místo pInv použije pInv'$\in$ Z, pInv' $\not\equiv$ pInv (mod q). Poznamenejme, že náhodná změna pInv tuto podmínku s velkou pravděpodobností splní. Ostatní parametry zůstanou beze změny.

Mějme nyní dvojici hodnot (m, s'), kde hodnota s' byla získána jako výsledek výše popsané podepisovací transformace při použití narušené hodnoty pInv´.
Platí tedy

1. $s_1 = m^d \bmod p$
2. $s_2 = m^d \bmod q$
3. $h' = pInv'*(s_2 - s_1) \bmod q$
4. $s' = s_1 + p*h'$
5. s´ je výsledek podepisovací transformace

S ohledem na faktor p pro tuto hodnotu dle rovnice z bodu 4 platí, že $s' \equiv m^d \pmod p$. Pro faktor q však s velkou pravděpodobností (blízkou $1 - q^{-1}$) platí, že $s' \not\equiv m^d \pmod q$. Dvojice čísel (m, y), kde $y = (s')^e \bmod n$, potom splňuje následující podmínky: $m \equiv y \pmod p$, $m \not\equiv y \pmod q$. Odtud plyne, že $p|(m-y)$, avšak zároveň $q \nmid (m-y)$. Proto pro faktor p platí, že $p = \gcd((m-y), n)$.

Uvedeným postupem jsme na základě jediného chybného podpisu získali hodnotu faktoru p, ze které potom již snadno určíme zbývající tajné hodnoty privátního klíče.

Poznamenejme, že uvedený postup předpokládá, že útočník zná hodnotu zformátované zprávy m, která přímo vstupuje do podepisovací transformace RSA. Toto nemusí být splněno pro všechny typy formátů, avšak v OpenPGP je doporučován formát RFC 2313 (alias PKCS#1, verze 1.5), kde podmínka útoku splněna je.

Stejně jako v případě útoku na DSA doporučujeme zavést do formátu OpenPGP silnější kontrolu integrity dat v souborech privátních klíčů. Přímo v programu PGP™ 7.0.3 oprava není nutná, neboť zde jsou použity dodatečné kontroly algebraických vztahů mezi hodnotami privátního klíče, které pokusy o útok tohoto typu maří.