Kleptografski ElGamalovi kriptosistemi na osnovi diskretnega logaritma

Jasmina Pegan

3. september 2020

Povzetek

Velik del kriptografskih funkcij deluje po principu črne škatle – uporabnik ne vidi implementacije. Young in Yung sta leta 1997 opisala mehanizem, imenovan SETUP (Secretly Embedded Trapdoor with Universal Protection), ki izkoristi prikritost morebitno zlonamerne vsebine takšne funkcije. S tem se ukvarja kleptografija – veda o varni kraji tajnih informacij.

V nalogi bomo opisali ElGamalov kriptosistem, spoznali kleptografijo in implementirali kleptografski ElGamalov kriptosistem. Spoznali bomo tudi primere implementacij, odpornih na ta tip napada. Nato bomo povzeli praktičnost grožnje in možne protiukrepe. Koda projekta je dostopna na naslovu github.com/jasminapegan/kleptographic-elgamal.

1 Uvod

V pričujočem delu predstavimo kleptografske napade na ElGamalov kriptosistem in kako se lahko ubranimo tovrstnih napadov. Predstavimo in testiramo tudi lastno implementacijo omenjenih napadov.

Kriptografske funkije uporabljamo, ko želimo varno hraniti in prenašati informacije. Zaradi obstoja raznih napadov na kriptografske sisteme pa je pri uporabi le-teh potrebna previdnost. Poglejmo si asimetrično šifrirno shemo. Denimo, da želi Ana komunicirati z Bojanom. Za to oba potrebujeta javni in zasebni ključ. Tudi če napadalec Oskar prestreže sporočila, jih brez zasebnih ključev ne more prebrati. Če pa Oskar nekako pridobi Anin zasebni ključ, lahko bere njena sporočila Bojanu. Eden izmed načinov za krajo zasebnih ključev je kleptografski napad.

Kleptografija je veda o varni in tajni kraji informacij. Izraz sta vpeljala Young in Yung v članku [4]. Bistvo kleptografskega napada je, da ima napadalec znatno prednost pri lomljenju šifre v primerjavi z uporabo grobe sile. Eden izmed mehanizmov kleptografskih napadov se imenuje SETUP (angl. *Secretly Embedded Trapdoor with Universal Protection*). Ta mehanizem temelji na načrtovani ranljivosti, ki omogoči avtorju-napadalcu možnost pridobivanja izpuščenih (angl. *leaked*) informacij, kot je žrtvin zasebni ključ.

Young in Yung sta v člankih [3] leta 1996 in [4] leta 1997 definirala kleptografske napade tipa SETUP. Predstavila sta tovrstne napade na razne kriptografske protokole, kot sta RSA in DSA. Pokazala sta tudi, da je možno nastaviti SETUP v vse kriptosisteme, ki temeljijo na problemu diskretnega logaritma (DLP). Mohamed in Elkamchouchi [11] sta leta 2010 predstavila kleptografske napade na kriptosisteme, ki temeljijo na eliptičnih krivuljah. Leta 2015 je Blomqvist [1]

spisal magistrsko nalogo, ki zajema celotno področje kleptografije in primere algoritmov za napade na razne kriptosisteme, med drugim tudi napad na ElGamalov kriptosistem, s katerim smo si pomagali pri izvedbi napadov v okviru tega dela. Na kleptografske napade odporne modele kriptografskih funkcij so razvili Russell in sod. [10] leta 2016. Modeli so splošni in ob uporabi izničijo stranski učinek kleptografske funkcije s SETUP.

V 2. poglavju ponovimo potrebne definicije in algoritme, kot je ElGamalov kriptosistem. Sledi 3. poglavje o kleptografiji na splošno ter aplicirani na ElGamalov kriptosistem. V 4. poglavju opišemo našo implementacijo SETUP mehanizma v ElGamalovem kriptosistemu, v 5. poglavju pa testiranje omenjene implementacije. V 6. poglavju opišemo na SETUP odporne algoritme. Delo zaključimo s 7. poglavjem, v katerem opišemo praktičnost kleptografskih napadov ter spoznanja o protiukrepih.

2 ElGamalov kriptosistem

Preden spoznamo kleptografske napade na ElGamalov kriptosistem, si oglejmo matematično ozadje komponent tega sistema. Bralec, ki je seznanjen z ElGamalovim kriptosistemom lahko to poglavje preskoči. ElGamalovi shemi povzamemo po Stinsonovem učbeniku [2].

Generator psevdonaključnih števil (angl. *pseudorandom number generator*, PRNG) je deterministični algoritem, ki generira navidez naključno zaporedje števil. To pomeni, da se zaporedje obnaša kot naključno glede na čim več statističnih testov naključnosti.

Za testiranje naključnosti zaporedja obstajajo testne suite, kot je `ENT` [13]. Suita `ENT` vključuje izračun entropije zaporedja bajtov, porazdelitve χ^2, povprečja zaporedja, koeficienta serijske korelacije in izračun π po metodi Monte Carlo iz zaporedja. Naključno zaporedje bajtov ima entropijo blizu 8 bitov/bajt, povprečno vrednost 127.5 in koeficient serijske relacije blizu 0. Iz zaporedja izračunan π po metodi Monte Carlo je blizu realni vrednosti te konstante. Pri izračunu porazdelitve χ^2 se izpiše tudi verjetnost, da je porazdelitev naključno večja. Čim je verjetnost med 10% in 90%, zaporedje opravi ta test naključnosti.

Šifriranje in podpisovanje po ElGamalu se odvijata nad ciklično grupo G. Običajno je to multiplikativna grupa $\mathbb{Z}_p^*$ za neko praštevilo p. Javna parametra kriptosistema sta praštevilo p in generator g multiplikativne grupe $\mathbb{Z}_p^*$. Vsaka oseba U, ki želi pošiljati šifrirana sporočila in odšifrirati prejeta sporočila, potrebuje zasebni ključ $x_U \in \mathbb{Z}_p^*$, ki je generiran z PRNG in javni ključ $y_U = g^{x_U} \mod p$. Naj bo (x_A, y_A) par ključev osebe A in (x_B, y_B) par ključev osebe B.

2.1 ElGamalova shema za šifriranje

Denimo, da želi oseba A poslati šifrirano sporočilo osebi B. Algoritem poteka v treh fazah – v prvi fazi se generira parameter, sledita šifriranje in odšifriranje. Osebi se navadno vnaprej dogovorita za vrednosti praštevilskega modula p in generatorja g, ki sta javna parametra.

1. *Generiranje ključev.* Osebi generirata zasebna ključa x_A in x_B. Javna ključa y_A in y_B izbereta tako, da ustrezata definiciji zgoraj.

2. *Šifriranje.* Oseba A želi zašifrirati sporočilo m in ga poslati osebi B. Najprej izbere naključno število $k \in \mathbb{Z}_p^*$ in izračuna $r = g^k \mod p$ ter $c = m \cdot y_B^k \mod p$. Osebi B pošlje par (r, c).

3. *Odšifriranje.* Oseba B prejme par (r, c). Sporočilo odšifrira: $m = s \cdot (r^{x_B})^{-1} \mod p$.

Pokažimo še, da odšifriranje šifriranega sporočila vrne začetno sporočilo:

$$m_2 m_1^{-b} = m g^{ab} (g^a)^{-b} = m g^{ab} g^{-ab} = m.$$

2.2 ElGamalova shema za podpisovanje

Denimo, da želi oseba A podpisati sporočilo, oseba B pa naj ima možnost podpis preveriti. Kot pri shemi za šifriranje imamo javna vnaprej izbrana parametra p in g.

1. *Generiranje ključev* poteka enako kot pri shemi za šifriranje.
2. *Podpisovanje.* Oseba A želi podpisati sporočilo m in poslati podpisano sporočilo osebi B. Najprej izbere naključno število $k \in \mathbb{Z}_{p-1}^*$ in izračuna $r = g^k \mod p$ ter $s = k^{-1}(m - x_A \cdot r) \mod p-1$. Osebi B pošlje trojico (m, r, s).
3. *Preverjanje podpisa.* Oseba B lahko preveri, da je prejeto sporočilo podpisala oseba A: $g^m = y_a^r \cdot r^s \mod p$.

V praksi se podpisuje zgoščeno vrednost sporočila $H(m)$, če je sporočilo večje od $p-1$.
Pokažimo, da preverjanje podpisa deluje pravilno:

$$g^m = g^{x_A \cdot r + k \cdot s} = (g^{x_A})^r \cdot (g^k)^s = y_A^r \cdot r^s \mod p.$$

2.3 Varnost

Varnost ElGamalovega kriptosistema temelji na problemu DLP, za katerega se predpostavlja, da je težek. NIST priporoča za Diffie-Hellmanovo izmenjavo ključev, da ima praštevilo p vsaj 2048 bitov [9]. Ker Diffie-Hellman praav tako sloni na DLP, bomo priporočilo prenesli na ElGamalov kriptosistem. Parameter k mora biti vsakič naključno izbran. Če napadalec pozna sporočilo m_1 in se pri šifriranju sporočila m_2 vrednost k ponovi, potem lahko pridobi tudi drugo sporočilo m_2. Drži namreč

$$c_1 \cdot m_1^{-1} = x_U^k = c_2 \cdot m_2^{-1} \mod p$$

in posledično

$$m_2 = c_2 \cdot c_1^{-1} \cdot m_1 \mod p.$$

Če se k ponovi v podpisni shemi, lahko napadalec pridobi zasebni ključ podpisovalca. Velja

$$s_1 \cdot k - m_1 = -x_U \cdot r = s_2 \cdot k - m_2 \mod p-1,$$

iz česar lahko napadalec izračuna k s pomočjo sledečega para ekvivalenc:

$$(s_1 - s_2) \cdot k = m_1 - m_2 \mod p-1 \quad (1)$$

$$g^k = r \mod p. \quad (2)$$

Zasebni ključ osebe U dobi tako:

$$x_U \cdot r = m_1 - k_1 \cdot k \mod p-1.$$

2.4 Praktična uporaba

ElGamalova shema za šifriranje se uporablja v novejših različicah PGP ter GNU Privacy Guard. ElGamalova shema za podpisovanje v izvorni različici se skoraj ne uporablja, ker je dolžina podpisa veliko večja kot pri ostalih algoritmih za podpisovanje. Vendar na tem algoritmu temelji DSA (angl. *Digital Signature Algorithm*), ki je poleg omenjenih PGP in GNU Privacy Guard uporabljen tudi v OpenSSL.

3 Kleptografija

Zadnja vrata (angl. *backdoor*) so prikrit način za dostop do sistema ali krajo informacij, ki se izogne običajnim varnostnim ukrepom, kot sta npr. avtentikacija in šifriranje. Primer zadnjih vrat je trojanski konj, ki omogoča napadalcu dostop do okuženega računalnika.

Kleptografija se ukvarja z zadnjimi vrati v protokolih, ki omogočajo napadalcu, da dostopa do tajnih informacij. Kovalenko in Kudin v [5] opišeta, da področje kleptografije zajema načrtovanje ter odkrivanje kleptografskih funkcij in izdelavo kriptografskih protokolov, ki so odporni na kleptografske napade.

Eden izmed načinov izpeljave kleptografskega napada je SETUP. Različica kriptografske funkcije s SETUP mora po članku Younga in Yunga [4] med drugim biti učinkovita, ne sme vsebovati napadalčevih skrivnosti in vmesnik se mora ujemati s prvotnim. Poleg tega niče ne more ločiti med izhodom funkcije in njene predelane različice in iz implementacije ni mogoče pridobiti preteklih ali prihodnjih ključev. Mehanizem SETUP omogoča napadalcu, da nezaznavno in neizsledljivo pridobi žrtvin zasebni ključ. Poleg tega lahko le napadalec, ki je ustvaril SETUP, dostopa do tako pridobljenih informacij.

3.1 Primeri

Najbolj znan primer kleptografske funkcije je obelodanil Snowden leta 2013, kot poroča Landau v članku [7]. Kleptogram je bil vložen v standardiziran generator psevdonaključnih števil Dual_EC_DRBG, ki temelji na grupi eliptične krivulje. Sicer pa je že pred Snowdenovim razkritjem generator veljal za sumljivega, saj je bil počasnejši in bolj pristranski (angl. *biased*) od ostalih generatorjev naključnih števil v uporabi, kot poročajo Bernstein, Lange in Niederhagen v članku [8].

Poseben primer kleptografskega napada je SETUP. Nekateri znani SETUP mehanizmi temeljijo na RSA, DSA, ElGamal na osnovi diskretnega logaritma in Kerberos, kot predstavita Young in Yung v članku [3]. Mohamed in Elkamchouchi v [11] predstavita tudi napade na ElGamalov kriptosistem na osnovi eliptičnih krivulj.

3.2 SETUP v ElGamalovem kriptosistemu

Pri generiranju ključev po ElGamalu je možno izvesti SETUP in pridobiti žrtvin zasebni ključ. Poleg tega obstaja tudi možnost kleptografske funkcije za podpisovanje. Sledeča algoritma smo povzeli po Blomqvistovem magistrskemu delu [1].

3.2.1 SETUP v ElGamalovem generiranju ključev

Napad na ElGamalovi shemi je izvedljiv v fazi generiranja ključev, če se parametra p in g (oziroma vsaj g) izračunata na novo, kar pa v praksi ni pogosto.

Naj bo x žrtvin zasebni ključ po ElGamalu, e ter d pa napadalčeva zasebni in javni ključ za RSA po modulu n. Parameter p je javen in fiksen. Parameter K izbere napadalec med generatorji $\mathbb{Z}_p^*$. Na sliki 1 je prikazan SETUP v algoritmu za generiranje ključev.

Napadalec uporabi javne parametre p, g, y za izračun žrtvinega zasebnega ključa x. Postopek, ki ga uporabi, je opisan na sliki 2.

1. Izbran je naključen $x \in \mathbb{Z}_p$.
2. Dobljena vrednost x je šifrirana z napadalčevim zasebnim ključem po RSA in randomiziran s funkcijo R_1 z uporabo ključa $K + i$: $x' = R_1(K + i, \gamma) \mod p$, kjer je $\gamma = x^e \mod n$. Sprva je $i = 0$, a dokler velja $x' \geq n$, se i povečuje do meje B_1.
3. Če je $x' < n$, se x' šifrira z napadalčevim javnim ključem: $x'' = (x')^e \mod n$.
4. Dobljeni x'' je šifriran z napadalčevim zasebnim ključem po RSA in randomiziran s funkcijo R_2 z uporabo ključa $K + j$: $x''' = R_2(K + j, \gamma') \mod p$, kjer je $\gamma' = (x')^e \mod n$. Sprva je $j = 0$, a dokler x' ni generator $\mathbb{Z}_p$, se povečuje do meje B_2.
5. Če je x' generator $\mathbb{Z}_p$, se generator posodobi: $g \leftarrow x'$.
6. Rezultat je četverica (g, p, x, y), kjer je $y = g^x \mod p$.

Slika 1: SETUP v ElGamalovem generiranju ključev.

1. Izračuna se $x'' = R_2^{-1}(K + j, g)$, na začetku je $j = 0$. Vrednost se odšifrira: $x' = (x'')^d \mod n$.
2. Za $i = 0$ se izračuna $x = R_1^{-1}(K + i, x')$. Če velja $g^x = y \mod p$, smo našli zasebni ključ. Sicer nadaljujemo s preiskovanjem prostora rešitev za različne vrednosti i in j.

Slika 2: Algoritem s katerim napadalec pridobi žrtvin zasebni ključ.

3.2.2 SETUP v ElGamalovi shemi za podpisovanje

Napad na ElGamalovo shemo za podpisovanje je bolj realističen, saj ne predpostavlja ponovnega generiranja ključev. Napad potrebuje vsaj dva dogodka komunikacije in ni vedno uspešen. Pri tem napadu mora napadalec uporabiti enaka parametra p in g, kot ju uporablja žrtev. Naj bo x žrtvin zasebni ključ ter X in Y napadalčeva zasebni in javni ključ po ElGamalu. Parameter K izbere napadalec tako, da generira $\mathbb{Z}_p^*$. Podpisuje se sporočilo m. Napad se izvede v dveh fazah, ki sta opisani na sliki 3.

Napadalec prestreže zaporedni sporočili (m, r, s) in (m', r', s'). Če je napad bil uspešen, lahko s postopkom na sliki 4 pridobi javni ključ žrtve.

3.3 Protiukrepi

Možnosti kleptografskega napada se lahko izognemo na več načinov. Lahko preverimo izvorno kodo, a le če je ta prosto dostopna, pa še takrat je ta način neučinkovit. Sumljiv algoritem lahko določimo s preverjanjem njegovega obnašanja, raziščemo lahko na primer statistiko rezultatov in časovno zahtevnost algoritma ter preverimo če se ugotovitve ujemajo s pričakovanim obnašanjem algoritma. Za izhode kriptografskih funkcij navadno pričakujemo, da sledijo enakomerni statistični porazdelitvi. Kleptografska funkcija je običajno počasnejša od čiste različice funkcije, a v praksi razlika velikokrat ni zaznavna.

Še ena možnost je uporaba samovarovalnih shem (angl. *self guarding schemes*), ki so neobčutljive na tovrstne napade. Te si bomo podrobneje ogledali v 6. poglavju.

Prva faza:

1. Izbran je naključen $x \in \mathbb{Z}_p$ in naključen $k \in \mathbb{Z}_{p-1}$, da je $\gcd(k, p-1) = 1$.
2. Izvede se podpisovanje kot običajno: $r = g^k \mod p$, $s = k^{-1} \cdot (m - Xr) \mod p - 1$.
3. Pošlje se trojica (m, r, s).

Druga faza:

1. Izračuna se $c = Y^k \mod p$, kjer je k enak kot v predhodnem koraku.
2. Če velja $\gcd(c, p-1) = 1$, poleg tega pa še $\gcd(c^{-1} \mod p, p-1) = 1$, je napad izvedljiv. Izračuna se $k' = c^{-1}$. Če pogoja nista zadoščena, se k' izbere kot v predhodnjem koraku.
3. Izvede se podpisovanje kot običajno: $r = g^k \mod p$, $s = k^{-1} \cdot (m - xr) \mod p - 1$.
4. Pošlje se trojica (m, r, s).

Slika 3: SETUP v ElGamalovi shemi za podpisovanje.

1. Izračuna se $c = r^X \mod p$. Če velja $\gcd(r', p-1) = 1$, je napad uspešen.
2. Rezultat je $x = (r')^{-1} \cdot (m' - s' \cdot c^{-1}) \mod p - 1$.

Slika 4: Algoritem s katerim napadalec pridobi žrtvin zasebni ključ.

4 Implementacija

Osnovno in kleptografsko različico ElGamalovega kriptosistema smo implementirali v jeziku Python s pomočjo knjižnice `pycryptodome` [12]. Pri implementaciji smo se opirali na algoritme, navedene na dodatnih straneh Blomqvistovega magistrskega dela [1].

Podatke, ki naj bi se shranili na napravi oziroma pri napadalcu, smo simulirali z ločenima datotekama tipa `json`. S tem smo zagotovili realnost scenarijev in ločenost dostopa do podatkov. Za vsak del kriptosistema smo implementirali program, ki se izvaja na žrtvini napravi ter program, s katerim napadalec pridobi zasebne podatke žrtve.

4.1 SETUP v fazi generiranja ključev

Funkcijo, ki se izvaja na žrtvini napravi, smo poimenovali `elgamal_keygen_setup`. Sprejme le parameter p, pa še to ni nujno, saj ga opcijsko generira na novo. Podatke o napadalčevem RSA ključu ter ključ K funkcija bere iz datoteke `device.json`. Med izvajanjem se posodobljena vrednost parametra K shrani nazaj v to datoteko. Večji izziv je predstavljala konstrukcija randomizacijskih funkcij s ključi R_1 in R_2. Uporabili smo AES v načinu CFB (angl. *Cypher Feedback*) z ničelnim inicializacijskim vektorjem. Tako smo ključ uporabili na običajen način in zagotovili konsistentno dolžino izhoda. Meji za parametra i in j sta nastavljeni na $B_1 = 16$, $B_2 = 512$, saj Young in Yung v [3] omenita, da ti vrednosti zadoščata za dobre rezultate. Funkcija vrne četverico parametrov in ključev (p, g, x, y).

Funkcija na napadalčevi napravi je `elgamal_keygen_recovery`. Sprejme parametre p, g in y, kjer je y žrtvin javni ključ. Podatke o napadalčevem RSA ključu ter ključ K funkcija bere iz datoteke `attacker.json`. Meji B_1 ter B_2 sta nastavljeni enako kot v prejšnji funkciji. Funkcija vrne x, ki naj bi ustrezal žrtvinemu zasebnemu ključu.

4.2 SETUP v shemi za podpisovanje

Na žrtvini napravi se izvaja funkcija `elgamal_signature_setup`. Sprejme parametre m, x, p in g, kjer je m sporočilo (oziroma njegova zgoščena vrednost) namenjeno podpisovanju in x je žrtvin zasebni ključ. Funkcija bere iz datoteke `device.json`, saj potrebuje parametra k (alternativni generator $\mathbb{Z}_p$, če se je program že kdaj izvedel) in y, napadalčev javni ključ po ElGamalu. Med izvajanjem se posodobljena vrednost parametra k shrani nazaj v to datoteko. Funkcija vrne trojico sporočil in podpisov (m, r, s).

Napadalec uporablja funkcijo `elgamal_signature_recovery`, ki sprejme parametre r, m_1, r_1 in s_1, kjer je r del predhodno prestreženega sporočila, (m_1, r_1, s_1) pa je naslednje sporočilo. Podatke o napadalčevem zasebnem ključu X in praštevilskem modulu p, ki je enak žrtvinem modulu, funkcija bere iz datoteke `attacker.json`. Če je napad bil uspešen, funkcija vrne kandidata za žrtvin zasebni ključ x.

5 Testiranje

Testirali smo pravilnost delovanja kleptografskih funkcij – zanimalo nas je, v kolikšni meri napadalec pridobi pravilno vrednost zasebnega ključa žrtve. Po definiciji mehanizma SETUP naj bi predelana funkcija bila tudi neločljiva od izvorne funkcije glede na čas izvajanja in lastnosti izhodov, zato smo ocenili tudi ti dve značilnosti funkcij. Vrednosti zasebnih ključev in podpisov naj bi izgledale naključne. Za oceno naključnosti rezultatov naših implementacij smo uporabili program `ENT` [13].

5.1 SETUP v generiranju ključev

Izvedli smo 1000 generiranj ključev in v vsakem poskusu smo uspešno pridobili žrtvin zasebni ključ. Za različne n smo izračunali sredino in standardni odklon časa izvajanja funkcije, ki izbere nov generator ter generira ključe in podobne funkcije, ki vsebuje SETUP. Pri iskanju generatorja g uporabljamo za faktorizacijo funkcijo `FactorDB` iz knjižnice `factordb`, ki je odvisna od povezave s spletom. To lahko vpliva na dobljene rezultate.

	čisti algoritem		SETUP	
n	mean	stddev	mean	stddev
1	0.165552	0.000000	0.165594	0.000000
10	0.187957	0.013119	0.198685	0.021864
100	0.192131	0.024461	0.196122	0.031935
1000	0.204958	0.036586	0.214924	0.029965

Tabela 1: Čas generiranja ključev z iskanjem novega generatorja g.

V tabeli 1 vidimo, da se povprečen čas izvajanja obeh različic algoritma ne razlikuje pretirano. Razlika je manjša od standardnega odklona, torej se po času izvajanja ne da razlikovati med algoritmoma.

Program `ENT` smo pognali na datoteki, ki vsebuje 1000 generiranih ključev. Rezultati so zbrani v tabeli 2.

test	rezultat	komentar
entropija	7.999234 bitov/bajt	zgornja meja: 8 bitov/bajt
porazdelitev χ^2	272.04	naključno večja v 22.13% primerih
aritmetična sredina	127.1504	naključno zaporedje: 127.5
Monte Carlo π	3.140392819	napaka 0.04%
koeficient serijske korelacije	0.003670	nekoreliranost: 0.0

Tabela 2: Ocena naključnosti zaporedja generiranih ključev glede na program ENT.

V tabeli 2 lahko vidimo, da je entropija našega zaporedja ključev zelo blizu zgornje meje, kar je značilno tudi za naključna zaporedja. Verjetnost, da ima naključno zaporedje višjo oceno testa χ^2 pove stopnjo nenaključnosti zaporedja. Ker je vrednost med 10% in 90%, lahko po opisu programa `ENT` sklepamo, da je zaporedje naključno. Aritmetična sredina je očitno blizu pričakovane, tudi vrednost π je ocenjena precej natančno, z napako 0.04%. Korelacijski koeficient zaporedja je blizu 0, torej je zaporedje praktično nekorelirano. Povzamemo lahko, da je naš algoritem za generiranje ključev dosegel pričakovano stopnjo naključnosti.

5.2 SETUP v funkciji za podpisovanje

Izvedli smo 1000 podpisovanj naključnih sporočil. Žrtvin zasebni ključ smo uspešno pridobili v 10% primerov. SETUP je bil uporabljen le v 18% primerov. Če gledamo uspešnost pridobitve ključa le v teh primerih, je napad uspel v 56% primerov.

Za različne n smo izračunali sredino in standardni odklon časa izvajanja običajne funkcije za podpisovanje in funkcije za podpisovanje, ki vsebuje SETUP.

	čisti algoritem		SETUP	
n	mean	stddev	mean	stddev
1	0.027885	0.000000	0.050904	0.000000
10	0.025931	0.001092	0.067827	0.011697
100	0.026003	0.001341	0.062806	0.011477
1000	0.026396	0.001237	0.068234	0.012534
10000	0.026776	0.001729	0.065076	0.011940

Tabela 3: Čas podpisovanja naključnih sporočil.

V tabeli 3 vidimo, da se povprečen čas izvajanja obeh različic algoritma precej razlikuje. Ker je različica z zadnjimi vrati počasnejša od čiste funkcije skoraj za faktor 2, lahko algoritma ločimo po času izvajanja.

Program ENT smo pognali na datoteki, ki vsebuje 10000 podpisov naključnih sporočil. Rezultati so zbrani v tabeli 4.

V tabeli 4 lahko vidimo, da je entropija našega zaporedja podpisov zelo blizu zgornje meje, kar je značilno tudi za naključna zaporedja. Verjetnost, da ima naključno zaporedje višjo oceno testa χ^2 pove stopnjo nenaključnosti zaporedja. Ker je vrednost med 10% in 90%, lahko sklepamo, da je zaporedje naključno. Aritmetična sredina je očitno blizu pričakovane, tudi vrednost π je

test	rezultat	komentar
entropija	7.999226 bitov/bajt	zgornja meja: 8 bitov/bajt
porazdelitev χ^2	274.62	naključno večja v 19.03% primerih
aritmetična sredina	127.3831	naključno zaporedje: 127.5
Monte Carlo π	3.144424132	napaka 0.09%
koeficient serijske korelacije	-0.003019	nekoreliranost: 0.0

Tabela 4: Ocena naključnosti zaporedja podpisov glede na program ENT.

ocenjena precej natančno, z napako 0.09%. Korelacijski koeficient zaporedja je blizu 0, torej je zaporedje praktično nekorelirano. Povzamemo lahko, da je naš algoritem za podpisovanje dosegel pričakovano stopnjo naključnosti.

6 Samovarujoči ElGamalov kriptosistem

V članku [10] Russell in sod. predstavijo generično konstrukcijo samovarujočih shem za šifriranje ter podpisovanje. Uporabimo jo lahko za gradnjo samovarujočega ElGamalovega kriptosistema, ki je odporen na kleptografske funkcije za šifriranje in podpisovanje. Galindo in sod. članku [6] navajajo tudi shemo BEG-KEM, ki prikrije ključe, s čimer lahko odpravimo SETUP pri izmenjavi ključev.

6.1 Samovarujoča ElGamalova shema za šifriranje

Naj bosta E in D funkciji za šifriranje ter odšifriranje po ElGamalu, ki morda vsebujeta SETUP. Shema sestoji iz štirih faz:

1. generacija ključev,
2. vzorčenje – šifrira se n naključno generiranih sporočil: $c_i = E(m_i)$,
3. šifriranje – šifrira se skrito (angl. *blinded*) sporočilo $E(m \cdot m_i)$ in množi z c_i^{-1},
4. odšifriranje – $D(E(m \cdot m_i) \cdot c_i^{-1})$.

Lahko vidimo, da shema deluje pravilno, saj je rezultat šifriranja $E(m)$:

$$\begin{aligned} E(m \cdot m_i) \cdot c_i^{-1} &= E(m \cdot m_i) \cdot E(m_i)^{-1} \\ &= E(m \cdot m_i \cdot m_i^{-1}) \\ &= E(m), \end{aligned}$$

tu smo uporabili homomorfnost ElGamalovega šifriranja in inverza števila v $\mathbb{Z}_p$.

6.2 Samovarujoča ElGamalova shema za podpisovanje

Naj bosta S in V funkciji za podpisovanje ter potrjevanje veljavnosti podpisa po ElGamalu, število λ pa velikost ključev. Shema sestoji iz štirih faz:

1. generacija ključev – generira se λ parov ključev (x_i, y_i),
2. vzorčenje – podpiše se n naključno generiranih sporočil: $s_i = S(x_i, m_i)$, vsakič z novim ključem,
3. podpisovanje:

- za $i = 1, ..., \lambda$ se izbere naključni bit b_i in izvede dve podpisovanji: $S(x_i, m_i)$ in $\sigma_i = S(x_i, m_i \oplus [m||s_i])$
- Če je $b_i = 0$, se uporabi podpisa v tem vrstnem redu, sicer se ju zamenja.
- Na vsakem koraku se preveri, ali velja $m_i = \sigma_i$. Če pride do neujemanja, se izvajanje prekine.
- Podpis sporočila m je $(m_1, s_1, \sigma_1, ..., m_\lambda, s_\lambda, \sigma_\lambda)$.

4. preverjanje podpisa – za vsak i se izračuna $m_i \oplus [m||s_i]$ in preveri veljavnost $V(x_i, m_i, s_i)$ ter $V(x_i, m_i \oplus [m||s_i], \sigma_i)$. Podpis je veljaven, če se delni podpisi v vseh korakih uspešno preverijo.

7 Zaključek

Večina standardiziranih kriptosistemov dopušča možnost kleptografskega napada. Ker se pogosto uporabljajo kot črna škatla, lahko traja leta preden se SETUP odkrije.

Videli smo primer napada na fazo generiranja ključev ter na podpisovanje sporočila v ElGamalovem kriptosistemu. Vzpostavitev napada ni preprosta, saj mora napadalec povzročiti spremembo na ciljni napravi. Ko je SETUP na mestu, je sprememba lahko nezaznavna.

Implementirali smo dva kleptografska napada na ElGamalov kriptosistem. Prvi spremeni algoritem za generiranje ključev, drugi pa algoritem za podpisovanje. Napad na generiranje ključev je vedno uspešen, napad na podpisovanje pa le v 18% primerov. SETUP v generiranju ključev ne povzroči opazne zakasnitve, medtem ko SETUP v podpisovanju podaljša izvajanje za faktor 2. Oba algoritma dosežeta pričakovano stopnjo naključnosti. SETUP v fazi generiranja ključev potrebuje možnost ponovnega izračuna generatorja g, kar v praksi ni pogosto. Za praktično rabo bi zato kljub slabši uspešnosti in daljšem času izračuna bil bolj primeren SETUP v algoritmu za podpisovanje.

Najvarnejši protiukrep je uporaba samovarujočih shem. Te so navadno računsko, časovno in prostorsko zahtevnejše od osnovnih shem, a onemogočijo tu opisane napade tipa SETUP.

Literatura

[1] F. Blomqvist. Kleptography – Overview and a new proof of concept, magistrsko delo, Aalto University, 2015.

[2] D. R. Stinson, M. Paterson. Cryptography: theory and practice. CRC press, 2018.

[3] A. Young, M. Yung. The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?. In *Advances in Cryptology—CRYPTO '96*, pp 89–103, 1996.

[4] A. Young, M. Yung. Kleptography: Using Cryptography against Cryptography. In *Advances in Cryptology—EUROCRYPT '97*, pp 62–74, 1997.

[5] B. Kovalenko, A. Kudin. Kleptography trapdoor free cryptographic protocols. *IACR Cryptology ePrint Archive 2018*, pp 989, 2018.

[6] D. Galindo, J. Großschädl, Z. Liu, S. Vivek, P. K. Vadnala. Implementation of a leakage-resilient ElGamal key encapsulation mechanism. In *Journal of Cryptographic Engineering*, Volume 6, Issue 3, pp 229–238, 2016.

[7] S. Landau. Making sense from Snowden: What's significant in the NSA surveillance revelations. In *Security & Privacy*, vol. 11, pp 54-63, 2013.

[8] D. J. Bernstein, T. Lange, R. Niederhagen. Dual EC: A standardized back door. In *The New Codebreakers*, pp 256-281. Springer, Berlin, Heidelberg, 2016.

[9] National Security Agency. Commercial National Security Algorithm (CNSA) Suite Factsheet. apps.nsa.gov. 2015. Available: `https://apps.nsa.gov/iaarchive/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/commercial-national-security-algorithm-suite-factsheet.cfm`. Accessed 1. 9. 2020.

[10] A. Russell, Q. Tang, M. Yung, H.-S. Zhou. Cliptography: Clipping the power of kleptographic attacks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp 34-64. Springer, Berlin, Heidelberg, 2016.

[11] E. Mohamed, H. Elkamchouchi. Kleptographic attacks on elliptic curve cryptosystems. In *International Journal of Computer Science and Network Security*, vol. 10(6), pp 213-215, 2010.

[12] E. Helder. PyCryptodome. Available: `https://github.com/Legrandin/pycryptodome`. Accessed 3. 8. 2020.

[13] J. Walker, ENT: A pseudorandom number sequence test program. Available: `https://www.fourmilab.ch/random/`. Accessed 3. 8. 2020.