SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

SUSTAV ZA STVARANJE SIGURNOSNIH KOPIJA DOKUMENATA

Završni rad

Borna Jelić

Osijek, 2017.

Sadržaj:

1. UVOD

U današnjem načinu života, dokumentiranje i arhiviranje u elektronskom obliku je primarni način produkcije i čuvanja dokumenata u svim sferama ljudskog života (poslovnoj, edukacijskoj, sigurnosnoj, administrativnoj, itd.). S druge strane, sve češće se pojavljuju elektronske prijetnje tim zapisima u kontekstu ili prijetnje njihovim uništenjem ili iznuđivanjem novčanih ili drugih protuusluga u zamjenu za pristup zapisu. Iz potrebe zaštite elektronskih zapisa, razvijena su rješenja koja najčešće pokušavaju te zapise zaštititi stvaranjem sigurnosnih kopija posljednjih verzija originalne datoteke. Međutim, takav način ne osigurava potpunu zaštitu jer je baziran na procesu sinkronizacije datoteka, čime bi se zaražena datoteka sinkronizirala s kopijom te poništila svrhu stvaranja sigurnosne kopije.

U drugome poglavlju rada analizirani su enkripcijski kripto virusi i neenkripcijski *ransomware,* neučinkovitost postojećih antivirusnih programa te bitcoin kriptovaluta i njezina uloga u virusnom napadu. U sljedećem poglavlju detaljno su opisani najčešće korišteni sustavi za kreiranje sigurnosnih kopija datoteka te njihove prednosti i nedostaci. Uočeno je kako proces *hashinga* može biti iskorišten kao postupak kontrole „netaknutosti“ datoteke te je upotrijebljen u izradi aplikacije, a sama izrada je opisana u četvrtom poglavlju.

1.1. Zadatak završnog rada

Zadatak završnog rada je kreirati aplikaciju za stvaranje sigurnosnih kopija datoteka koja će neutralizirati nedostatak postojećih alata (sinkronizacija datoteka). Kroz teorijsku podlogu, detaljno je razrađen način funkcioniranja aplikacije, kao i procedure koje koristi u svome radu (vremenska dinamika, usporedba putem heširanja).

2. KRIPTO VIRUSI

Kripto virusi, podgrupa *ransomwarea*, su vrsta zlonamjernog softvera koja korisniku blokira pristup računalu i/ili podacima te od njega zahtijevaju plaćanje otkupnine (engl. *ransom*). Otkupnina i službeni razlog zašto bi je žrtva morala platiti ovisi o tipu virusa, no najčešće se radi o kripto virusnom iznuđivanju, odnosno verzijama koje korisnika obavještavaju da je to jedini način na koji će moći dešifrirati zaključane podatke. Dodatne aktivnosti kripto virusa mogu uključivati krađu korisnikovih osjetljivih informacija, uništavanje legalnog softvera i dr. Ne postoji jamstvo da će uplaćivanje novčane svote ili bilo koja druga zahtijevana radnja omogućiti pristup šifriranim datotekama, odnosno računalu. [1]

Mogu napasti računala s bilo kojim operacijskim sustavom (Windows, Mac OS X, Linux) i mobilne uređaje (Android, iOS, Windows Phone). Postoje više verzija kripto virusa, no svima je zajedničko da korisniku sprječavaju normalno korištenje računala te traže od korisnika određenu radnju (uplatu) prije korištenja.

Kripto virusi na računala mogu dospjeti na više načina. Najčešće su automatski skinuti prilikom posjeta zlonamjernim ili hakiranim web stranicama. Također, otvaranjem privitaka unutar e-mail poruka ili samim otvaranjem e-mail poruka od ljudi koje korisnik ne poznaje ili ne očekuje takvu vrstu poruke. U novije vrijeme, moguće se zaraziti ovakvim virusima i klikom na sumnjivu ili lošu poveznicu preko Facebooka, Twittera ili bilo kojeg drugog društvenog medija.

Način rada kripto virusa događa se u tri faze [2]:

1) Napadač generira par ključeva šifriranja te ugrađuje odgovarajući javni ključ u sami virus.
2) Virus slučajno generira simetrični ključ i šifrira žrtvine podatke. Javni ključ se koristi za šifriranje simetričnog ključa. Ova radnja rezultira malim asimetrično šifriranim tekstom i simetrično šifriranim tekstom podataka, čime je onemogućeno vraćanje originala. Ovo je poznato kao hibridna enkripcija. Žrtva tada šalje asimetrični šifrirani tekst i novac napadaču.
3) Napadač dešifrira šifrirani tekst svojim privatnim ključem i šalje simetrični ključ žrtvi. Žrtva tada simetričnim ključem dešifrira podatke i time napad kripto virusom završava.

Ovakvi virusi češće ciljano napadaju poduzeća i velike kompanije s ciljem ulaska na unutarnju mrežu poduzeća i iznuđivanja veće svote novca. Općenito, napadač ima određen popis ekstenzija dokumenata ili lokacija direktorija koje će kripto virus napasti. Kripto virusi se dijele na dvije velike skupine: *lockscreen ransomware* i *encryption ransomware*. [3]

2.1. „LockScreen“ *ransomware*

Ransomware virus „*LockScreen*“, poznatiji po verziji policijski virus ili „MUP virus“, originalno „FBI virus“, nije virus već obična izvršna datoteka koja se instalira na računalo korištenjem propusta u Javi i programiran je da bi ometao korisnika u radu te ga naveo da plati navodnu kaznu. Program po pokretanju blokira pristup radnoj površini i na ekranu prikaže, često polupismen tekst koji upozorava korisnika na protuzakonitu radnju. Ovakva vrsta kripto virusa ne šifrira datoteke, već kada inficira računalo pokušava na računalu pronaći ilegalne datoteke, kao što je pornografski sadržaj ili nelicencirane datoteke. U slučaju pronalaska, žrtva je informirana o njihovom pronalasku te se moli da plati „kaznu“ ne bi li izbjegla odlazak u zatvor (slika 2.1.). [4]

Sl. 2.1. MUP kripto virus [4]

Posebna vrsta ovakvog kripto virusa je program koji zaključava web preglednik. Ova verzija ne inficira računalo, već se oslanja na JavaScript program koji blokira preglednik i uzrokuje prikazivanje poruke upozorenja korisniku, koje su vrlo slične kripto virusima koji ne šifriraju datoteke (slika 2.2.). [1][4]

Ovakva vrsta virusa je manje štetna od enkripcijskog kripto virusa i korisnik ga se može lakše riješiti stvaranjem sigurnosne kopije bitnih datoteka i instaliranjem operacijskog sustava ponovno.

Sl. 2.2. Primjer blokiranog web preglednika [5]

2.2. Enkripcijski kripto virusi

Enkripcijski kripto virusi su virusi koji inficiraju računalo i šifriraju korisnikove datoteke. Ovakva verzija *ransomware* programa se najčešće šire uz pomoć različitih verzija virusa „Trojanski konj“. Kada inficira računalo, virus pronalazi datoteke koje se najčešće koriste ili imaju ekstenzije koje se nalaze na njegovoj ciljanoj listi. Šifrirane datoteke najčešće uključuju fotografije, video snimke, glazbene datoteke, poslovne i slične podatke koji se smatraju važnim korisniku. Kao i „LockScreen“ kripto virus, enkripcijski kripto virus počet će prikazivati veliko upozorenje gdje se navodi jedini način dešifriranja podataka – plaćanje otkupnine. To je često i istina, jer većina takvih virusa obriše *„shadow“*[1] kopije datoteka da bi se spriječilo vraćanje. Najpoznatiji primjeri su: *Wana Decrypt0r 2.0* (slika 2.3.) *i CryptoLocker* (slika 2.4.). Najnoviji

[1] „Shadow copy“ je tehnologija ugrađena kao servis u Windows OS koja omogućava automatsko ili ručno stvaranje sigurnosnih kopija datoteka putem praćenja promjena nad datotekama i bilježenjem tih promjena. https://www.howtogeek.com/129188/htg-explains-what-are-shadow-copies-and-how-can-i-use-them-to-copy-or-backup-locked-files/

masovni napad dogodio se 14. svibnja 2017. za koji je zaslužan *WannaCry*, verzija već spomenutog *Wana Decrypt0r 2.0*.

Sl. 2.3. Wana Decrypt0r 2.0 kripto virus [6]

Sl. 2.4. CryptoLocker kripto virus [7]

2.3. Antivirusni programi i kripto virusi

Antivirusni programi se vrlo teško nose s detekcijom kripto virusa, zbog toga što stopostotna zaštita od ovakvog tipa virusa ne postoji. Također, navedeno je da postoje verzije ovakvog virusa koje zapravo nisu virus, već obični programi tipa *Notepad*. Moguće je podijeliti ovaj problem u tri kategorije:

1) Baza potpisa virusa – antivirusna industrija se uglavnom temelji na uzorcima i kreiranju obrane protiv konkretnog *malwarea*, dolazak do uzoraka kod ovakvih virusa je dugotrajan posao jer se kriptiranje dogodi izuzetno brzo i korisnici ne stignu reagirati, a nakon obavljene radnje virus izbriše sve tragove
2) Heuristika – kripto virusi nemaju karakteristike ponašanja *malwarea*. U principu je ista stvar kada korisnik dobije e-mail s linkom na kojega klikne, preuzme program s Interneta i pokrene ga, itd. Kada bi antivirusni programi prepoznavali ovakve radnje, koje su praktički svakodnevne, nijedna legitimna aplikacija ne bi mogla biti pokrenuta bez upozorenja na opasnost od virusa
3) Brojne varijante – na crnom tržištu je moguće kupiti izvorni kod i prilagoditi ga svojim potrebama

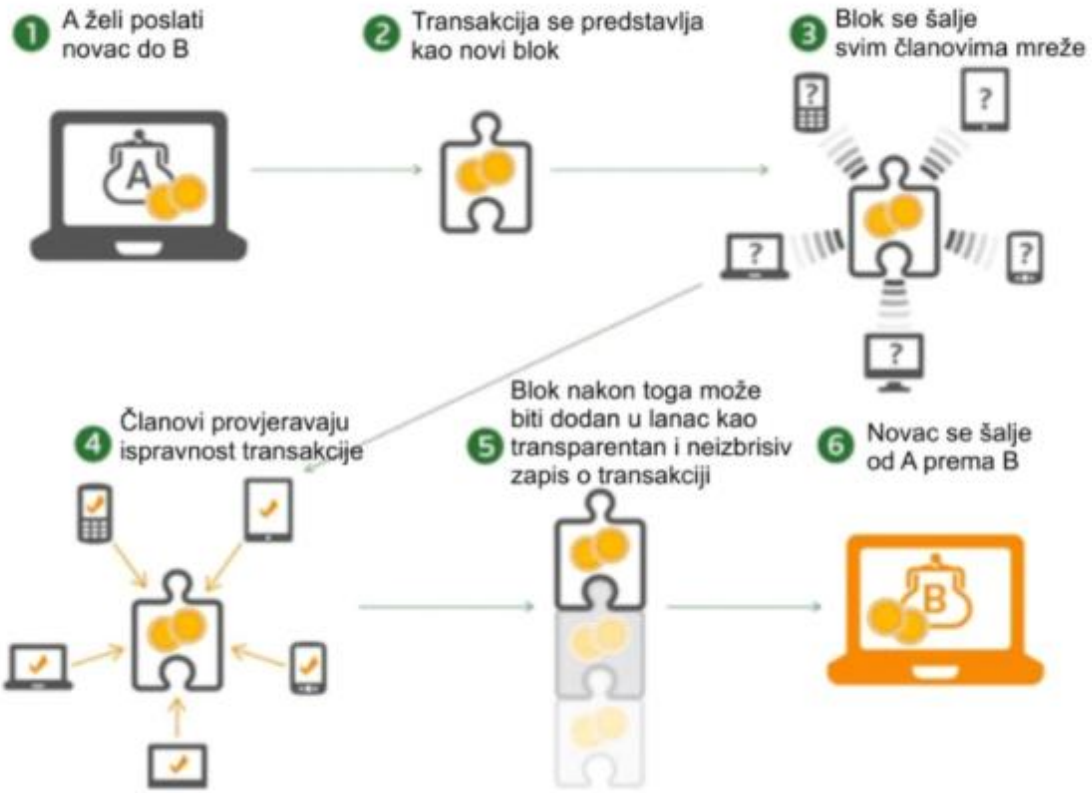
2.4. Bitcoin

Bitcoin je produkt ideje Satoshi Nakamotoa[2], a predstavlja digitalni novac, stvoren i čuvan elektronički, tj. nije printan niti kontroliran od strane bilo koga. [7] Proizvodi se pomoću računala, koristeći softver koji rješava matematičke probleme. Bitcoin je prvi primjer ovakve valute nazvane kriptovaluta. Koristi se za kupovinu u elektroničkom obliku, čime se ne razlikuje ni od jedne druge valute. No, ono što pravi najveću razliku od ostalih valuta jest što je decentraliziran, odnosno nijedna institucija ne kontrolira bitcoin mrežu. [8]

Karakteristike bitcoina su jednostavnost otvaranja bitcoin računa, zanemarivi transakcijski troškovi, transparentnost transakcija i nepovratan prijenos bitcoina. Također, po nekima i najvažnija karakteristika je anonimnost korisnika, koja je djelomična zbog transparentnosti transakcija. Korisnik može posjedovati više bitcoin adresa te one nisu povezane s imenom, adresom ili drugim osobnim podatcima. Međutim, podaci o svakoj transakciji koja se ikad dogodila unutar bitcoin mreže pohranjuje se u tzv. *Blockchain* (slika 2.5.). Ako posjedujete adresu,

[2] „Satoshi Nakamoto“, navodno rođen 5. travnja 1975. u Japanu. Ime Satoshi Nakamoto predstavlja alias pravoga tvorca bitcoina, koji je još uvijek anoniman.

svatko može vidjeti koliko je bitcoina pohranjeno na toj adresi. [8] Korisnik ima svoj generirani privatni ključ „*wallet*“ koji generira javni ključ. Prilikom transakcije, taj javni ključ se veže s transakcijom, čime se može pratiti stanje transakcije. [9] Time je omogućeno praćenje i bilježenje svake transakcije na bitcoin mreži u javnu glavnu računovodstvenu knjigu (engl. *ledger*), tj. *Blockchain* kojoj svatko može pristupiti i provjeriti transakciju.



Sl. 2.5. Bitcoin procedura i tijek novca [10]

2.5. Bitcoin i kripto virusi

Zbog anonimnosti korisnika koji su uključeni u transakciju bitcoina te nepovratnosti novca nakon izvršenja transakcije, bitcoin je najpopularnija valuta i način plaćanja putem kojeg napadači traže otkupninu. Bitcoin je time postala valuta na lošem glasu, jer joj se predbacuje da je valuta koja može prikriti zločin. Prije pojave bitcoina, napadači su tražili uplatu preko PayPala, ali zbog povećanja njegove sigurnosti i KYC (engl. *Know Your Customer*[3]) metode prešli su na kriptovalutu bitcoin.

[3] KYC metoda PayPal sustava periodično pregledava PayPal korisničke račune i postavlja određena ograničenja trošenja i primanja novca, čime pokušava spriječiti pranje novca i ostale kriminalne radnje. [11]

2.6. Hashing

Hashing je proces generiranja *hash* vrijednosti podataka. *Hash* vrijednost (ili jednostavno *hash*), također poznat pod imenom *message digest*, je broj generiran iz dijela teksta (engl. s*tring of text*), odnosno podatka. *Hash* je uvijek jednake veličine neovisno o veličini odabranog *stringa* te je generiran preko matematičke formule. Iako je poznat *hashing* algoritam, iz *hash* vrijednosti nije moguće dobiti ulazni podatak, jer su informacije izgubljene. Najznačajnija karakteristika procesa heširanja je da se vrlo teško, skoro nemoguće može dobiti jednaki *hash* obradom 2 različita *stringa*. Javni ključevi uobičajeno koriste puno kompleksnije algoritme i vrlo velike *hash* vrijednosti za šifriranje, uključujući 40-bitne i čak 128-bitne vrijednosti. 128-bitni broj ima 2^{128} različitih kombinacija, što bi značilo da je vjerojatnost pronalaska određene vrijednosti bila jednaka pronalasku određenog zrna pijeska u pustinji Sahara, čime je osigurana „potpuna“ jednoznačnost *hasha*.

Hash igra značajnu ulogu u sigurnosnim sustavima gdje se koristi radi osiguranja poruka od falsificiranja i sprečavanja utjecaja treće strane u komunikacijskom kanalu. *Hashing* se upotrebljava najčešće zbog osiguravanja poruke naknadnog mijenjanja.

Datoteka može biti korumpirana na više načina: greška prilikom slanja, greška prilikom korištenja aplikacije, neispravna pohrana ili zbog neispravnog medija na koji se pohranjuje, itd. Prilikom slanja datoteke, pošiljatelj generira *hash* poruke, šifrira ga i šalje zajedno sa porukom. Primatelj dešifrira i poruku i *hash* te generira novi *hash* zaprimljene poruke. Usporedbom dva *hasha* donosi se zaključak je li poruka originalna ili je promijenjena tijekom slanja. Zbog prirode *hash* funkcija, može doći do *hash* kolizije, odnosno događaja gdje postoje dvije identične *hash* vrijednosti dobivene kao rezultat različitih ulaza *hash* funkcije. Vjerojatnost pojave kolizije se može smanjiti tako da se duljina ulaznog *stringa* poveća. *Hash* kolizija može rezultirati lažnim pozitivnim rezultatima usporedbe *hash* vrijednosti, ali je vjerojatnost pojave kolizije zanemariva u usporedbi s vjerojatnošću slučajne korupcije datoteka. Rukovanje kolizijom se obavlja na jedan od dva načina [14]:

- Ulančavanjem – svaka ćelija tablice *hash* vrijednosti pokazuje na povezani popis podataka koji imaju jednaku *hash* vrijednost. Ovaj način je jednostavan, ali zahtjeva memorije izvan tablice

- Otvoreno adresiranje – svi elementi su pohranjeni u samoj tablici *hash* vrijednosti. Svaki unos sadrži ili *hash* ili NIL[4] vrijednost. Prilikom pretrage određenog elementa, vrši se pretraga svakog elementa tablice dok se ne nađe traženi element ili ne prođe kroz cijelu tablicu (slučaj kada ne postoji taj element)

Hashing se također koristi i kao metoda za brzo pretraživanje baze podataka. Kreira se indeks nad podacima baze, tj. tablica *hash* vrijednosti, tako što se algoritam heširanja primjeni na svaki unos u bazi. Prilikom pretrage određenog unosa, dovoljno je samo ponovno primijeniti algoritam koji direktno daje *hash* vrijednost tog unosa, a time se pronađe cijeli traženi unos unutar baze.

Hashing ima veliku ulogu u tržištu Bitcoina. Princip rada Bitcoin *hash* algoritma je sljedeći [15]:

1) Blok podataka je objavljen javno te prije nego ga poslužitelj može prihvatiti radi pregleda, mora pronaći podatak koji kada se doda bloku će rezultirati *hash* vrijednosti s određenim brojem nula.
2) Pošto je računanje obrnute *hash* funkcije kompleksno, jedini način da se ovo odradi je dodavanjem pokusnih podataka bloku podataka te računanje *hash* vrijednosti svaki puta dok se ne dobije rezultat s traženim brojem nula. Ovo zahtjeva puno vremena te je velika vjerojatnost da će samo jedan poslužitelj naići na rješenje, čime se izbjegava problem gdje više poslužitelja (često Bitcoin rudari) istovremeno pregledava isti blok podataka.
3) Kada je rezultat pronađen i blok podataka proglašen ispravnim, transakcijski podatak je dodan bloku tako da se podaci unutar bloka ne mogu više mijenjati od tada pa nadalje, čime se osigurava ispravnost podataka i sigurnost od utjecaja treće strane.

Najčešći *hash* algoritmi su SHA-1 (160-bitni *hash*), SHA-2 (najčešće 256-bitni i 512-bitni *hash*) i MD5 (128-bitni *hash*). MD5 je osmislio Ronald Rivest 1991. čime zamjenjuje prethodni MD4, dok je SHA-1 i SHA-2 osmislila Agencija za nacionalnu sigurnost SAD-a 1995. odnosno 2001. Zbog savjetovanja analitičara 2005. i 2010. da SHA-1 neće biti dovoljno siguran za buduću upotrebu, Microsoft, Google, Apple i Mozilla su objavili da će prestati koristiti SHA-1 do kraja 2017.

```
MD5("The quick brown fox jumps over the lazy dog") =
9e107d9d372bb6826bd81d3542a419d6
```

Sl. 2.6. Primjer MD5 *hash* vrijednosti za zadani ulaz

[4] NIL vrijednost je unos u tablicu *hash* vrijednosti kada ne postoji *hash* vrijednost jednaka indeksu unutar tablice za određeni unos.

3. TRENUTNI SUSTAVI ZA KREIRANJE SIGURNOSNIH KOPIJA DATOTEKA

Otkako je Interneta, korisnici računala su podložni hakerskom napadu i razaranju podataka. Korisnici su počeli raditi duplikate svojih podataka ne bi li im barem jedna kopija ostala netaknuta. Izumom eksternog tvrdog diska, korisnici su mogli stvoriti sigurnosnu kopiju svojih podataka *off-grid*, čime su podaci, nakon izvršenja kopiranja, sigurni jer nemaju doticaja s Internet mrežom. No, veličina i količina podataka raste velikom brzinom, pogotovo logistički i računovodstveni podaci poduzeća. Kao i interna memorija, eksterna memorija je ograničena i preskupa za stalnu nadogradnju. Počelo se pribjegavati Internetu. Razvojem mrežnih tehnologija, pogotovo tehnologija u oblaku (engl. Cloud[5]) - točnije modela sigurnosne kopije kao usluge (engl. Backup as a Service – BaaS) i modela softvera kao usluge (engl. Software as a Service – SaaS), sustavi za kreiranje sigurnosnih kopija datoteka prešli su na Internet način pohrane. Prelaskom na Internet, postavlja se pitanje koliko su ti podaci zapravo sigurni. U ljudskoj prirodi je da su ljudi sigurniji i više vjeruju u nešto što je fizički ispred njih, poput eksterne memorije, nego u nešto virtualno. Također, postavlja se pitanje privatnosti na Internetu. Poslužitelji su vlasnici računalne opreme, a time i podataka koji se nalaze na njoj. Razina privatnosti koju poslužitelji moraju poštovati regulira se korisničkim ugovorom o korištenju usluge, a često i zakonskim putem.

Ipak, korisnici vjeruju dugogodišnjim proizvođačima pouzdanih aplikacija, poput Microsofta i Google-a te se pojavljuju novi pouzdani igrači na velikom informatičkom tržištu – Dropbox Inc. i Mega Group. Korištenje pojedinih servisa nije besplatno, no većina poslužitelja prilikom kreiranja korisničkog računa korisniku dodijeli određenu količinu resursa besplatno. Daljnje resurse je potrebno zakupiti, odnosno plaćati mjesečno naknadu za njihovo korištenje.

Microsoft OneDrive i Google Drive ne nude samo jedan model usluge, već nude na korištenje i pojedine vlastite aplikacije poput Excel Online ili Word Online (Microsoft OneDrive), odnosno Google Hangouts ili Google Sheets (Google Drive) te mnoge druge usluge partnerskih tvrtki. Ovdje će se opisati način rada kreiranja sigurnosnih kopija datoteka tri najkorištenija servisa: Google Drive, Microsoft OneDrive i Dropbox.

[5] Cloud tehnologija je grana računarstva u kojoj vrijedi princip zakupa, dijeljenja i korištenja tuđih resursa, infrastrukture i aplikacija preko Interneta.

3.1. Google Drive

Google je u travnju 2012. pokrenuo servis Google Drive da bi ponudio alternativni sustav dijeljenja dokumenata konkurentnome Microsoft 365. Njihova originalna platforma za dijeljenje datoteka, Google Docs nastala je u rujnu 2007. Dijelovi Google Docs platforme, pogotovo sustav pohrane datoteka, su migrirani na novi Google Drive servis. [12]

Google Drive je *cloud* rješenje za pohranu datoteka i dijeljenje dokumenata. Cilj mu je poslužiti korisniku kao mobilni osobni repozitorij datoteka. Korisnike privlači svojom jednostavnošću korištenja, mogućnosti višekorisničkog načina rada na jednom dokumentu istovremeno i mogućnosti pokretanja s bilo kojeg uređaja spojenog na Internet i koji podržava Google Drive aplikaciju. Također ima mogućnost otvaranja različitih formata datoteka koji nisu podržani na korisnikovom računalu.

Google Drive ima mogućnost kreiranja sigurnosnih kopija datoteka tako da su datoteke spremljene na računalu i datoteke spremljene na Internetu sinkronizirane. Prilikom instalacije na računalo kreira se poseban direktorij koji služi kao dvosmjerni cjevovod. Datoteke i direktoriji koji su u tom direktoriju će se sinkronizirati s Google Drive-om, ali i sve datoteke i direktoriji koji su već na Google Drive-u, a ne na računalu, će se sinkronizirati nazad na računalo.

3.2. Microsoft OneDrive

Microsoft OneDrive je *cloud* usluga Microsofta koja omogućuje korisnicima pohranu datoteka kao i osobnih podataka poput Windows postavki na *cloud server*. Sam servis je pokrenut u kolovozu 2007. pod imenom *Windows Live Folders*, odnosno *SkyDrive*.

OneDrive servis dolazi besplatno s Microsoft korisničkim računom i korisnicima dodjeljuje 5 GB slobodnog prostora za pohranu podataka besplatno te pojednostavljenu verziju Office programskog paketa. Servis automatski stvara direktorij na računalu u koji korisnik može spremati datoteke i direktorije koje želi duplicirati u svrhu sigurnosne kopije. Kao i Google Drive, OneDrive se ne može nazvati pravim sustavom za kreiranje sigurnosnih kopija, jer se također radi o procesu sinkronizacije datoteka između dva računala (lokalno i *server*).

3.3. Dropbox

Dropbox je servis pohrane podataka koji korisnicima omogućuje *cloud* pohranu datoteka. Stvoren je 2007. godine kao startup tvrtka u San Franciscu, CA, SAD od strane dvojice MIT studenata.

Dropbox koristi drugačiji način rada od prethodna dva servisa. Dropbox također kreira direktorij, *My Dropbox,* na korisničkom računalu u koji korisnik stavlja datoteke koje želi duplicirati. No, datoteke su duplicirane više puta, odnosno original se nalazi na lokalnom računalu, a jedna sinkronizirana kopija na korisničkom računu na mreži te kopija prethodne kopije. Ako korisnik poveže više uređaja na isti korisnički račun, kopije se nalaze i na drugim uređajima. Dropbox snima povijest svih obrisanih i prethodnih verzija datoteka tijekom 30 dana. Sve datoteke pohranjene na mreži su šifrirane od strane Dropboxa i pohranjene u više podatkovnih centara. [13]

3.4. Usporedba servisa

Usporedbom tri najkorištenija servisa kreiranja sigurnosnih kopija datoteka može se zaključiti da rade na više-manje istom principu. Prilikom prvog pokretanja, stvara se direktorij koji je povezan s korisničkim računom tog servisa, te se jednosmjerno (ili u slučaju Google Drive servisa dvosmjerno) sinkroniziraju datoteke i direktoriji. Također, servisi omogućuju korisniku visoku razinu mobilnosti podataka.

Sinkronizacijom ne postižemo efekt stvaranja odvojene sigurnosne kopije, odnosno *backup*. Sinkronizacijom se ista verzija datoteke nalazi na računalu i *serveru*, odnosno sve promjene učinjene nad datotekom na računalu bit će odražene i na svim spojenim uređajima. Dakle, ako korisnik obriše datoteku na jednom uređaju, obrisat će ju sa svih. Pod sigurnosnom kopijom misli se na mogućnost vraćanja određene verzije ili cjelovite datoteke ako ju korisnik obriše.

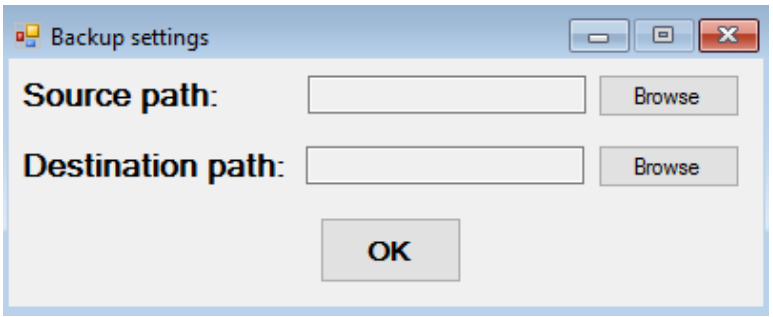
Idealni sustav *backupa* trebao bi moći stvarati kopije datoteka u svim verzijama, dakle jedna originalna verzija te više verzija datoteke nakon svake spremljene izmjene. Time bi se korisniku omogućilo vraćanje datoteke u različitim stadijima, ovisno o njegovoj potrebi, čime bi se smanjila količina izgubljenih podataka. Svakom izmjenom datoteci se, između ostalih parametara, mijenja i *hash.* Taj slučajno generirani kod bi bio ono po čemu bi se verzije razlikovale, te usporedbom kodova se može vidjeti je li došlo do promjena nad datotekom od prethodnog korištenja. Uz *backup,* idealni sustav treba imati vlastiti način povratka datoteka čime bi se mogle vlastito šifrirati i dešifrirati kopije.

Za razliku od OneDrive-a i Google Drive-a, Dropbox prati povijest svih obrisanih i prethodnih verzija datoteka te se time približava idealnom servisu *backupa*.

4. PROGRAMSKO RJEŠENJE

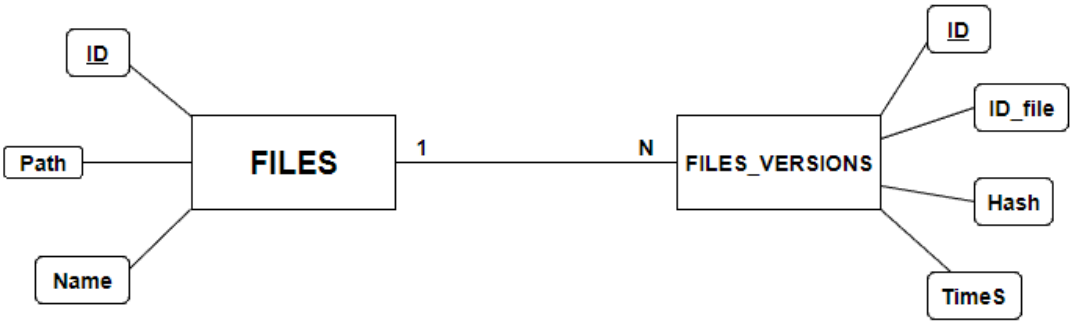
Trenutno najkorišteniji sustavi za stvaranje sigurnosnih kopija dokumenata stvaraju kopije na *cloudu.* No, dovode se u pitanje korisnici koji žele svoje kopije dokumenata pohranjivati na fizički mediji. Tržište koje zahtjeva nekakvo programsko rješenje za ovaj problem je pretežno na Windows platformi. Zato je odabrana tehnologija u kojoj se izrađuje aplikacija dio .NET platforme, odnosno C#. Izradit će se konzolna aplikacija za stvaranje sigurnosnih kopija i program s grafičkim sučeljem za vraćanje datoteka. Pokretanje konzolne aplikacije može se izvoditi autonomno preko *Windows Task Schedulera* ili ručno, dok se program za vraćanje pokreće ručno.

Programsko rješenje rađeno je u *Microsoft Visual Studio 2017* kao *Windows application* projekt, čime je omogućeno korištenje i konzole i *Windows* formi za grafičko sučelje. Korištena je lokalna *SQLite* baza podataka. Prosljeđivanjem argumenata programu prilikom pokretanja pokreće se određeni način rada, tj. *backup* ili *restore*. Prilikom prvog pokretanja, program moli korisnika da se unese izvorišni i odredišni direktorij, stvara se baza podataka gdje će se unositi podaci o kopiranim datotekama (slika 4.1.) te se odvija „nulti“ *backup*. Baza podataka sastoji se od dvije tablice, *Files* i *Files_versions,* međusobnog odnosa 1:N (slika 4.2.).



Sl. 4.1. Postavljanje izvorišnog i odredišnog direktorija

Tablica *Files* sadrži podatke o originalnoj putanji datoteke, odnosno mjesto na disku gdje se nalazi te njezino ime, dok tablica *Files_versions* sadrži unose pojedinih verzija svakog kopiranog dokumenta, tj. verzije svakog unosa u tablici *Files*, odnosno *hash* vrijednost datoteke i vremensku oznaku kada je *backup* određene verzije nastupio. Aplikacija sadrži i vlastiti instalacijski program koji pri instalaciji stvara 2 ikonice na radnoj površini, jednu za *restore* i jednu za ručno pokretanje *backupa*.



Sl. 4.2. E-R dijagram baze podataka

4.1. Aplikacija za stvaranje sigurnosnih kopija dokumenata

Jedna od glavnih značajki autonomnog stvaranja sigurnosnih kopija dokumenata je pozadinski način rada, odnosno korisnik ne smije biti opterećen ili ometan u svome radu dok se proces kopiranja odvija. Zato je prilikom pokretanja procesa kopiranja skriven konzolni prozor te se time postiže neki stupanj prikrivenosti. Uspješnost procesa može se vidjeti u tekstualnom zapisu koji program stvori za pojedinačni *backup*.

Program svoj rad započinje čitanjem parametara izvorišnog i odredišnog direktorija za *backup* iz konfiguracijskog dokumenta te ih šalje u funkciju *CopyAllFiles* gdje se odvija glavna zadaća programa (slika 4.3.).

```
//Start
string sourceDirectory = ConfigurationManager.AppSettings["sourceDirectory"];
string destinationDirectory = ConfigurationManager.AppSettings["destinationDirectory"];

CopyAllFiles(sourceDirectory, destinationDirectory);
```

Sl. 4.3. Početak izvršavanja programa

Unutar funkcije *CopyAllFiles* provjerava se postojanje izvorišne datoteke. Ako ne postoji, program stvara tekstualni zapis greške na koju je naišao i prekida izvršavanje. U suprotnome, funkcija prolazi kroz stablo datoteka izvorišnog direktorija i stvara listu datoteka te provjerava postojanje odredišne lokacije i stvara ju u slučaju nepostojanja. Otvara se konekcija na bazu podataka te se u tablici *Files* provjerava postoji li unos originalne datoteke pomoću funkcije *GetFilesFromDB* (slika 4.4. i 4.5.). *GetFilesFromDB* funkcija uspostavlja konekciju s bazom podataka te pokušava za određenu datoteku pronaći postoji li zapis putem objekta klase *SQLiteDataReader*. Ako je pročitan unos, funkcija završava s radom i šalje povratnu informaciju u glavni program. U slučaju pronađene datoteke, stvara se zapis u tekstualnu datoteku.

```
bool filesFlag = GetFilesFromDB(sourceDirectory, file.Name);

if (!filesFlag)
{
    conn.Open();
    using (SQLiteCommand cmd = new SQLiteCommand(conn))
    {
        // SQL Files Entry
        string insertFiles = "INSERT INTO Files (Path, Name) VALUES ('" + sourceDirectory + "','" + file.Name + "')";
        cmd.CommandText = insertFiles;
        cmd.ExecuteNonQuery();
    }
    conn.Close();
}
else
{
    CreateTextFileLog("The original " + file.Name + " is already backed up!");
}
```

Sl. 4.4. Provjera unosa originala u tablici *Files* i unos u tablicu

```
// File in DB check
private static bool GetFilesFromDB(string source, string fileName)
{
    string searchFiles = "SELECT ID FROM Files WHERE Path='" + source + "' AND Name='" + fileName + "'";
    using (SQLiteConnection conn = new SQLiteConnection(ConfigurationManager.ConnectionStrings["db"].ConnectionString))
    {
        if (conn.State != ConnectionState.Open)
        {
            conn.Open();
        }
        using (SQLiteCommand cmd = new SQLiteCommand(searchFiles, conn))
        {
            using (SQLiteDataReader rdr = cmd.ExecuteReader())
            {
                while (rdr.Read())
                {
                    conn.Close();
                    return true;
                }
            }
        }
        conn.Close();
    }
    return false;
}
```

Sl. 4.5. *GetFilesFromDB* funkcija

Nakon provjere za originalnu datoteku, provjerava se postoji li unos za trenutnu verziju datoteke putem funkcije *GetVersionsFromDB* koja radi na istom principu kao *GetFilesFromDB* samo što je *SQL* upit drugačije formuliran. Pomoću funkcije *GetMD5HashFromFile* (slika 4.6.) dohvaća se *hash* vrijednost datoteke u *byte* obliku, pretvara u *string* te uspoređuje sa svakim unosom *hash* atributa unutar *Files_versions* tablice (slika 4.7.). Vrijednost *hash* atributa je unikatna te se zbog toga može koristiti za diferencijaciju različitih verzija datoteke, odnosno za provjeru je li datoteka unesena u tablicu temeljem usporedbe *hash* vrijednosti trenutne datoteke i svih unosa u tablici.

```
// MD5 hash
private static string GetMD5HashFromFile(string fileName)
{
    FileStream file = new FileStream(fileName, FileMode.Open);
    MD5 md5 = new MD5CryptoServiceProvider();
    byte[] returnValue = md5.ComputeHash(file);
    file.Close();

    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < returnValue.Length; i++)
    {
        sb.Append(returnValue[i].ToString("x2"));
    }
    return sb.ToString();
}
```

Sl. 4.6. *GetMD5HashFromFile* funkcija

```
bool hashFlag = GetVersionsFromDB(hash);

if (!hashFlag)
{
    conn.Open();
    using (SQLiteCommand cmd = new SQLiteCommand(conn))
    {
        // SQL Files_versions entry
        string getFileId = "SELECT ID FROM Files WHERE Name='" + file.Name + "'";
        cmd.CommandText = getFileId;
        cmd.ExecuteNonQuery();
        Object returnValueId = cmd.ExecuteScalar();

        string insertVersions = "INSERT INTO Files_versions (ID_file, Hash, TimeS) VALUES ('"
            + returnValueId + "','" + hash + "','" + DateTime.Now.ToString("yyyy-MM-dd") + "')";
        cmd.CommandText = insertVersions;
        cmd.ExecuteNonQuery();
    }
    conn.Close();

    string tempFileName = file.Name + "_" + hash;
    string tempDestinationPath = Path.Combine(destinationDirectory, tempFileName);
    file.CopyTo(tempDestinationPath, true);
}
else
{
    CreateTextFileLog("File " + file.Name + ", version: " + hash + " is already backed up!\r\n");
}
```

Sl. 4.7. Provjera unosa u *Files_versions* i kopiranje datoteke

Ako ne postoji unos trenutne verzije datoteke, sustav tek tada vrši kopiranje datoteke iz izvorišnog direktorija u odredišni, prilikom čega stvara novi unos u tablicu *Files_versions*. U suprotnome, stvara se zapis u tekstualnoj datoteci da je trenutna verzija već kopirana.

Opisani procesi provjera i kopiranja događaju se i za sve poddirektorije unutar izvorišnog tako što se za svaki poddirektorij funkcija *CopyAllFiles* poziva rekurzivno.

Kopirane datoteke u svome imenu imaju dodanu cijelu *hash* vrijednost, čime se stvara vizualna razlika u imenima višestrukih kopija datoteka. Nakon što funkcija *CopyAllFiles* završi s radom, sustav kopirane datoteke na odredištu komprimira u jednu komprimiranu datoteku te joj u

ime dodaje datumsku oznaku. Stvara se tekstualna datoteka s rezultatima pojedinih koraka u kopiranju i rezultatom cjelokupnog *backupa*, čime je proces *backupa* završen.

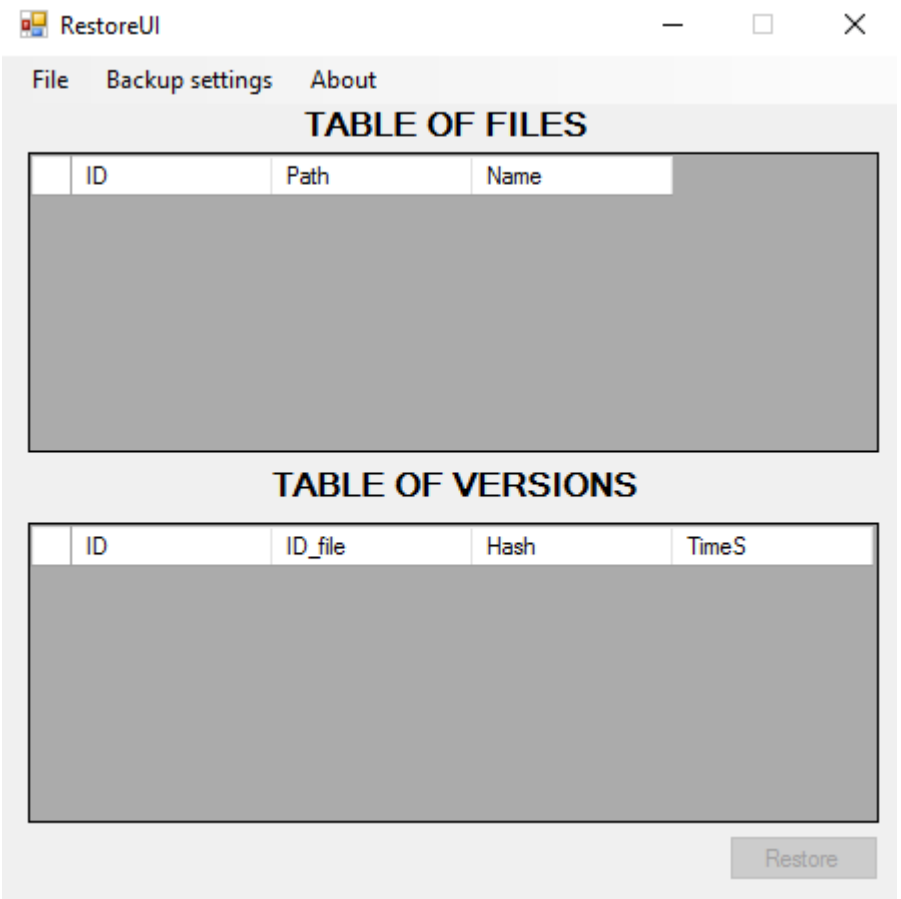
4.2. Program za vraćanje datoteka

Prilikom klika na ikonicu programa za vraćanje datoteka, operacijski sustav dobiva parametar „-ui“ čime je naznačeno da se izvršava dio programa s grafičkim sučeljem, odnosno dio za vraćanje datoteka (slika 4.8.).

```
if (args.Length > 0)
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new RestoreUI());
}
```

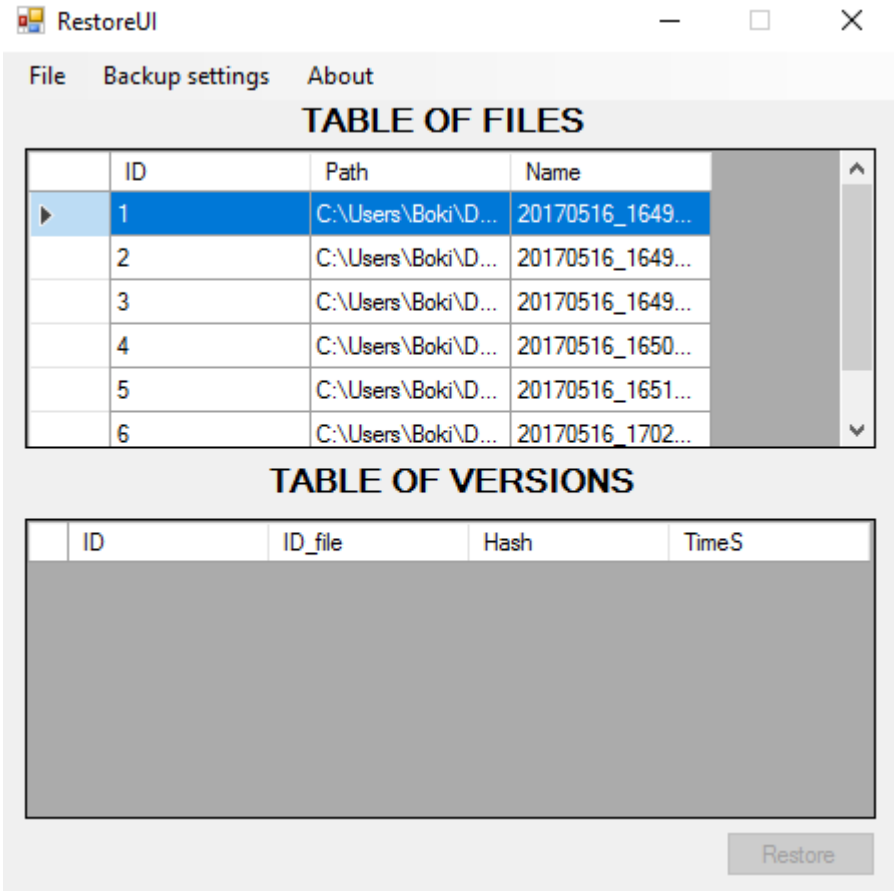
Sl. 4.8. Pokretanje programa za vraćanje datoteka

Početno grafičko sučelje programa (slika 4.9.) prikazuje okvire praznih tablica *Files* i *Files_versions* te traku izbornika.



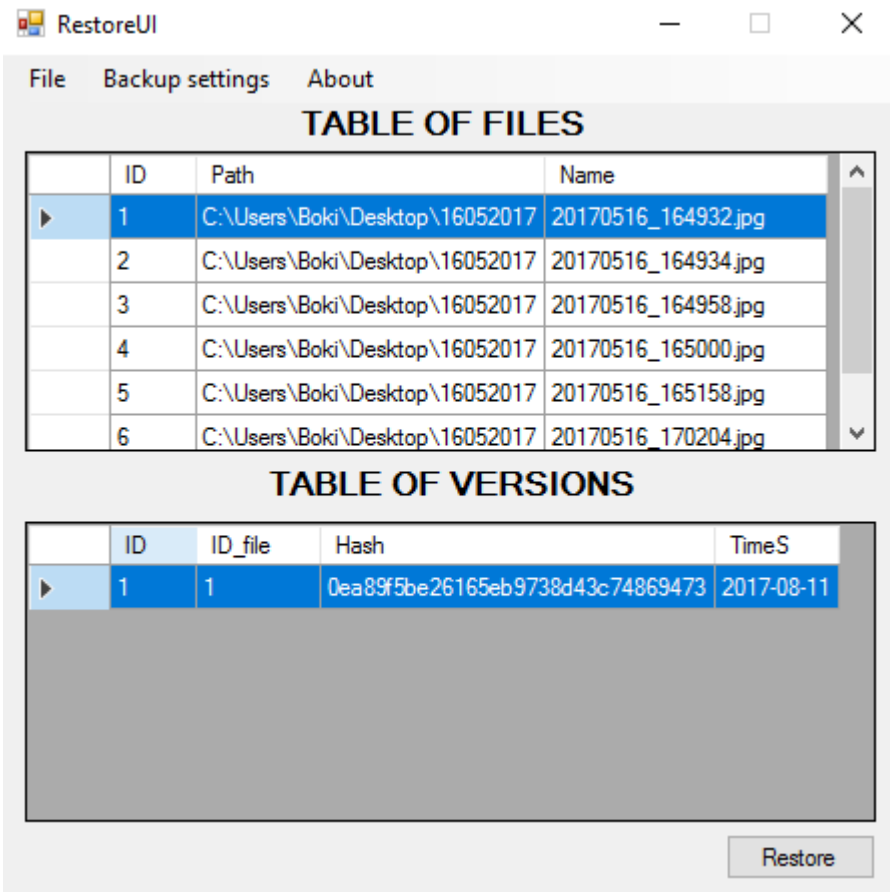
Sl. 4.9. Sučelje programa

Tipka *Restore* je onemogućena zato što nije odabran nijedan dokument i njegova verzija za vraćanje. Nakon pokretanja programa, prvo je potrebno u padajućem izborniku *File* povezati se na bazu podataka klikom na *Connect* prilikom kojeg se program poveže s bazom podataka te se prva tablica popunjava sa svim unosima u tablici *Files* (slika 4.10.).



Sl. 4.10. Popunjena tablica *Files*

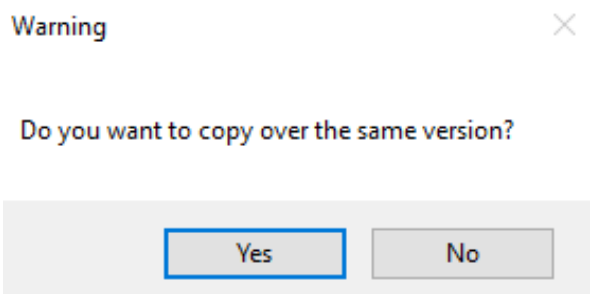
Tipku *Restore* je još uvijek nemoguće kliknuti. Time je onemogućena korisnikova pogreška u kojoj nije odabrana niti jedna verzija datoteke, jer je nemoguće znati koju verziju korisnik želi vratiti. Dvostrukim klikom na bilo koji redak u tablici datoteka, popunjava se tablica verzija samo datoteke koja je prethodno odabrana, odnosno filtriraju se svi unosi u tablici verzija tako da se prikazuju unosi sa stranim ključem koji je jednak primarnom ključu odabranog unosa u tablici *Files* (slika 4.11.).



Sl. 4.11. Prikaz verzija odabrane originalne datoteke

Unutar tablice verzija, prvi red je uvijek odabran da bi se izbjegla još jedna korisnikova pogreška gdje korisnik zaboravi odabrati verziju koju želi vratiti te se tipka *Restore* aktivira. Nakon što je odabrana verzija datoteke koju korisnik želi vratiti, klikom na tipku *Restore* program raščlanjuje podatke iz obje tablice, tako da si složi kompletni *path* do originala spajanjem vrijednosti atributa *Path* i *Name* iz prve tablice, te dohvaća *hash* vrijednost i vremensku oznaku iz druge tablice. Ti podaci služe programu da stvori put do odredišnog direktorija (prvenstveno izvorišni direktorij) i da se mogu uspoređivati *hash* vrijednosti u imenu kopija, kao i vrijednost trenutne datoteke na tom mjestu. Time dolazi do tri moguća načina vraćanja datoteka:

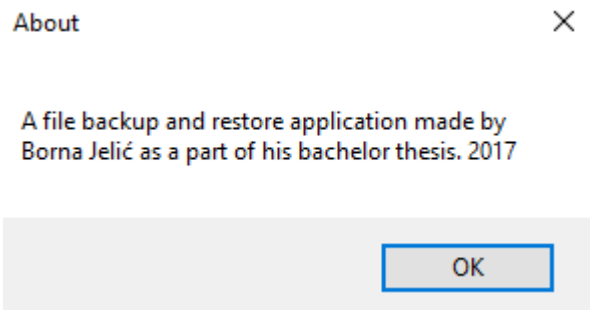
- Ako je datoteka koju korisnik želi vratiti jednaka po *hash* vrijednosti onoj koja se nalazi na tom mjestu, korisnik dobiva dodatni upit želi li nastaviti s kopiranjem (slika 4.12.)
- Ako je drugačija verzija nego trenutna na tom mjestu, vrši se direktno kopiranje preko stare verzije
- Ako ne postoji datoteka na odredištu, vrši se kopiranje odabrane verzije na odredište identično načinu stvaranja kopije prilikom *backupa*



Sl. 4.12. Dodatan upit prilikom vraćanje identične verzije

Ako korisnik ne želi prekopirati preko identične verzije, može ponovno izabrati verziju nekog dokumenta ili odabrati drugi dokument jer program ponovno poprima sučelje prikazano slikom 4.11. Nakon što korisnik povrati dokument, ima mogućnosti vraćati i druge dokumente ili druge verzije već vraćenog dokumenta. Baza podataka i dalje pamti unos vraćene verzije, neovisno o tome što je vraćena, čime je omogućeno ponovno vraćanje u budućnosti.

Na traci izbornika se također nalazi *Backup settings* koji omogućuje ponovno postavljanje izvorišnog i odredišnog direktorija. Program sprema novo postavljene podatke u konfiguracijski dokument te će ponovnim pokretanjem *backupa* program tek koristiti zadane podatke. Također se nalazi *About* u kojem je sažeto opisana namjena programa i naveden autor (slika 4.13.).



Sl. 4.13. Prozor *About*

Iz aplikacije je moguće izaći na dva načina: klikom na tipku za zatvaranje u gornjem desnom kutu ili klikom na opciju *Exit* u padajućem izborniku *File*. Izlaženjem iz aplikacije se zatvara konekcija na bazu, spremaju postavke izvorišta i odredišta *backupa* i zatvara se grafičko sučelje.

5. ZAKLJUČAK

U prvom dijelu završnog rada teorijski je obrađen način rada kripto virusa i opisan trenutni problem s antivirusnim programima te tako predstavljen problem koji svakodnevno prijeti korisnicima računala. Kripto virusi se prebrzo šire i djeluju za pravovremenu reakciju antivirusnog programa. Također, svakim danom ti virusi „mutiraju“, pa je nemoguće održavati bazu potpisa virusa potpunom.

Usporedbom trenutnih sustava za stvaranje sigurnosnih kopija dokumenata zaključeno je da se oni ne mogu nositi sa zaraženim datotekama, jer rade na principu sinkronizacije datoteka čime se virus prenese i na kopiju, a ne klasičnog *backupa* gdje uobičajeno, postoji trenutna i prethodna verzija. Time dolazi do potrebe za izradom sustava za stvaranje sigurnosnih kopija dokumenata koji će raditi po željenim principima i svojstvima klasičnog *backupa*.

Analizom trenutnih metoda zaštite datoteka u procesu komunikacije, postupak heširanja datoteka ima veliku ulogu. Funkcija heširanja dodjeljuje jedinstvenu matematičku vrijednost, tzv. *hash* po kojoj možemo razlikovati različite verzije iste datoteke. Upravo *hashing* predstavlja temelj oko kojeg je izrađena aplikacija za stvaranje sigurnosnih kopija dokumenata.

Aplikacija se sastoji od dva dijela: programa za stvaranje sigurnosnih kopija dokumenata i programa za vraćanje verzija datoteka. Program za stvaranje sigurnosnih kopija ima početni direktorij u kojemu počinje svoj prolazak kroz datotečno stablo, tražeći datoteke i stvarajući njihovu kopiju na odredištu, ako je potrebno. Program se pokreće u vrijeme određeno postavkama unutar *Windows Task Scheduler* alata te se nakon kopiranja potrebnih datoteka gasi. Program za vraćanje datoteka ima određenu apsolutnu putanju do direktorija u datotečnom sustavu gdje se nalaze sigurnosne kopije dokumenata. Program u svome grafičkom sučelju prikazuje sadržaj baze podataka, gdje korisnik odabire datoteku i njezinu verziju koju želi vratiti. Također, u programu se mogu postaviti izvorišni i odredišni direktorij *backupa*. Preko zaražene datoteke se vraća prijašnja, nezaražena verzija čime je virus uklonjen. Korisnik tada može obrisati zaraženu kopiju datoteke i unos u bazi podataka.

Postoji više smjerova u kojima se može poboljšati rad aplikacije. Dio aplikacije zadužen za *backup* može se razviti kao *Windows* servis, tada korisnik ne mora namještati pokretanje programa unutar *Task Scheduler* alata. Može se dodati funkcionalnost da sustav nakon svakog n-tog *backupa* obriše onaj najstariji čime se održava zauzeće memorije na disku. Programu za vraćanje datoteka je moguće dodati postavke vezane za način vraćanja datoteka, poput ručnog odabira mjesta vraćanja pojedine datoteke, različitog od mjesta odakle je prvotno stvorena sigurnosna kopija.

LITERATURA

[1] Jake Doevan, „*Ransomware programi*“, http://virusi.hr/ransomware-programi/, ožujak 2016., pristup ostvaren 23. svibnja 2017.

[2] A. Young, M. Yung, „*Cryptovirology: extortion-based security threats and countermeasures*“, IEEE Security and Privacy Conference 1996, IEEE Xplore, str. 129-141, 0-8186-7417-2, Oakland, CA, USA, 1996., pristup ostvaren 22. svibnja 2017.

[3] „*Ransomware*“, https://www.microsoft.com/en-us/security/portal/mmpc/shared/ransomware.aspx, Microsoft Corp., 2015., pristup ostvaren 23. svibnja 2017.

[4] „*Kako ukloniti LockScreen virus*“, http://www.nod32.com.hr/tabid/2341/faq/605/, kolovoz 2016., pristup ostvaren 23. svibnja 2017.

[5] Lucia Danes, „*Browser has been blocked*“, http://www.2-spyware.com/remove-your-browser-has-been-blocked.html, kolovoz 2015., pristup ostvaren 23. svibnja 2017.

[6] „*Your personal files are encrypted*“, https://malwaretips.com/blogs/remove-your-personal-files-are-encrypted-virus/, Malwaretips, 23. svibanj 2017.

[7] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, svibanj 2009., pristup ostvaren 23. svibnja 2017.

[8] „*Što je bitcoin?*“, http://crobitcoin.com/bitcoin, pristup ostvaren 23. svibnja 2017.

[9] Surbhi Gloria Singh, „*Ransomware attack: Why do hackers want payments through bitcoins?*“, http://www.business-standard.com/article/economy-policy/ransomware-attack-why-do-hackers-want-payments-through-bitcoins-117051700271_1.html, Business Standard, svibanj 2017., pristup ostvaren 23. svibnja 2017.

[10] I. Horvat, „Što je Bitcoin i Blockchain“, https://www.ucionica.net/internet/sto-su-bitcoin-blockchain-i-kriptovaluta-4199/, svibanj 2017., pristup ostvaren 17. kolovoza 2017.

[11] „*Know Your Customer*“, https://www.paypal.com/us/webapps/mpp/public-policy/issues/anti-money-laundering-and-know-your-customer, PayPal Service, 2017.

[12] Cristopher Alghini, „*Difference between Google Docs and Google Drive*“, http://www.coolheadtech.com/blog/heres-the-difference-between-google-docs-and-google-drive, ožujak 2013., pristup ostvaren 25. svibnja 2017.

[13] „*Does Dropbox keep backups of my files?*“, https://www.dropbox.com/help/security/file-backups, Dropbox Inc., 2017.

[14] „*Hashing*“, http://www.geeksforgeeks.org/hashing-set-1-introduction/, pristup ostvaren 26. svibnja 2017.

[15] Mike James, „*Hashing – the Greatest Idea in programming*“, http://www.i-programmer.info/babbages-bag/479-hashing.html, 2013., pristup ostvaren 26. svibnja 2017.

SAŽETAK

Naslov: Sustav za stvaranje sigurnosnih kopija dokumenata

Cilj ovog završnog rada je teorijski opisati način rada neenkripcijskog i enkripcijskih *ransomwarea*, značenje Bitcoin kriptovalute u virusnom napadu te objasniti postupak heširanja u razvijanju programske zaštite datoteka od kripto virusa. Usporedbom trenutno najkorištenijih sustava za stvaranje sigurnosnih kopija dokumenata zaključeno je da rade na drugačijem principu od klasičnog *backupa*. Potrebno je stvoriti sustav za stvaranje sigurnosnih kopija dokumenata u kojemu je moguće stvoriti sigurnosnu kopiju dokumenata određenih ekstenzija i omogućiti njihovo vraćanje. Usporedbom *hash* vrijednosti datoteke može se vidjeti promjena učinjena nad datotekom. Sustav provjerava *hash* vrijednost određenog dokumenta i uspoređuje s vrijednostima u bazi podataka. Ako je rezultat pretrage negativan, stvara kopiju datoteke i novi unos u bazu. Sustav time omogućuje vraćanje različitih verzija istog dokumenta, čime se izbjegava gubitak podataka. Aplikacija je izrađena u C# programskom jeziku.

Ključne riječi: kripto virusi, enkripcija, Bitcoin, hashing, sigurnosna kopija, C#

ABSTRACT

Title: File backup and restoration system

The aim of this bachelor thesis was to theoretically describe the non-encrypting ransomware and encrypting ransomware, usage of Bitcoin cryptocurrency in a viral attack and to explain the process of hashing within the file protection system. By comparing the most frequently used file backup systems, it was concluded that they do not operate like a classic backup system should. It is necessary to create a file backup and restoration system in which it is possible to create copies of files with specific file extensions and to enable their restoration. By comparing hash values of a file, a change within the file can be detected. The system checks the hash value of the selected document and compares them with the values in its database. If the result of the search is negative, the system creates a copy of the selected file and a new entry in the database. With that, the system enables restoration of different versions of the file, avoiding data loss. The application was made in C# programming language.

Key words: ransomware, encryption, Bitcoin, hashing, backup, C#

ŽIVOTOPIS

Borna Jelić rođen je 25. ožujka 1996. u Osijeku. U Osijeku završava osnovnu školu „Mladost“ te 2010. upisuje Prirodoslovno-matematičku gimnaziju. 2014. ostvaruje direktan upis na Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, smjer računarstvo. Dobitnik je nagrade za uspješnost u studiranju za 2016./2017. akademsku godinu.