Formalna verifikacija post-kvantnog protokola pomoću alata Tamarin

Futivić, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 615

FORMALNA VERIFIKACIJA POST-KVANTNOG PROTOKOLA POMOĆU ALATA TAMARIN

Ivan Futivić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 615

FORMALNA VERIFIKACIJA POST-KVANTNOG PROTOKOLA POMOĆU ALATA TAMARIN

Ivan Futivić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 615

Pristupnik:	**Ivan Futivić (0036522493)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Ante Đerek

Zadatak: **Formalna verifikacija post-kvantnog protokola pomoću alata Tamarin**

Opis zadatka:

Tradicionalni kriptografski algoritmi i protokoli su ranjivi na napade kvantnim računalima, a cilj post-kvantne kriptografije je razvoj algoritma i protokola koji su otporni na takve napade. U sklopu ovog diplomskog rada potrebno je istražiti post-kvantne inačice često korištenih kriptografskih protokola poput Signal protokola, te istražiti pokušaje formalne verifikacije takvih protokola. Na temelju istraživanja, potrebno je provesti formalno modeliranje i analizu odabranog post-kvantnog protokola koristeći alat Tamarin. Pritom, formalna analiza uključuje definiranje sigurnosnih svojstava kao što su povjerljivost, integritet, autentičnost poruka i unaprijedna povjerljivost. Radu je potrebno priložiti izvorni kod razvijenih i korištenih programa, citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se svojem mentoru Anti Đereku na pomoći koju mi je pružio i savjetima tijekom pisanja ovog rada. Zahvaljujem se svojoj djevojci koja me motivirala na svakom koraku i bila mi društvo tijekom pisanja rada. Naposljetku, zahvaljujem svojim roditeljima koji su mi bili podrška od početka do kraja studiranja.

Sadržaj

1. Uvod

Sigurnost modernih komunikacijskih protokola temelji se na kriptografskim algoritmima i protokolima koji su otporni na trenutne tehnološke prijetnje. Međutim, s razvojem kvantnih računala, tradicionalni kriptografski protokoli postaju ranjivi. Ova prijetnja motivira razvoj post-kvantne kriptografije koja ima cilj osigurati otpornost na napade kvantnim računalima. Tradicionalni kriptografski protokoli, poput RSA i ECC, oslanjaju se na matematičke probleme koje je teško riješiti klasičnim računalima, ali koji postaju trivijalni za kvantna računala. Post-kvantna kriptografija razvija nove algoritme i protokole koji su dizajnirani da budu sigurni i protiv kvantnih napada.

U ovom diplomskom radu istražuje se PQXDH protokol, post-kvantna inačica X3DH protokola korištenog u aplikaciji Signal i mnogim drugim aplikacijama za razmjenu poruka. X3DH protokol koristi kriptografiju eliptičkih krivulja za osiguranje povjerljivosti i autentičnosti, oslanjajući se na sigurnost tradicionalnih kriptografskih metoda. S druge strane, PQXDH protokol uvodi post-kvantne komponente kako bi osigurao otpornost na napade kvantnim računalima. Konkretno, PQXDH koristi post-kvantni KEM (Key Encapsulation Mechanism) kako bi osigurao povjerljivost, dok i dalje koristi tradicionalne metode za osiguranje autentičnosti. Ova kombinacija omogućuje PQXDH protokolu da zadrži sigurnosna svojstva X3DH protokola uz dodatnu otpornost na kvantne napade. Fokus rada je formalno modeliranje i verifikacija PQXDH protokola, s ciljem da se analizira njegova otpornost na napade kvantnim računalima te da se potvrde ključna sigurnosna svojstva kao što su povjerljivost, integritet, autentičnost i unaprijedna povjerljivost. Kroz detaljnu analizu koristeći alat za formalnu verifikaciju Tamarin, bit će prikazano kako se protokol nosi s potencijalnim prijetnjama i koje su modifikacije moguće kako bi se dodatno unaprijedila njegova sigurnost.

Formalna verifikacija je metoda kojom se matematički dokazuju sigurnosna svojstva

kriptografskih protokola. Korištenjem alata poput Tamarin, moguće je rigorozno analizirati složene protokole i otkriti potencijalne ranjivosti koje bi mogle proći neopaženo drugim metodama testiranja. Rezultati formalne verifikacije u ovom radu pokazali su da modificirani PQXDH protokol zadovoljava sva očekivana sigurnosna svojstva.

Prvi dio rada opisuje X3DH protokol i njegova sigurnosna svojstva, pružajući temelj za razumijevanje originalnog protokola. Drugi dio posvećen je PQXDH protokolu, objašnjavajući njegove post-kvantne modifikacije u odnosu na X3DH protokol. Treći dio analizira postojeću sigurnosnu analizu i formalnu verifikaciju PQXDH protokola, pružajući uvid u pronađene potencijalne ranjivosti. Četvrti dio opisuje izrađeni Tamarin model, definirana ispitana sigurnosna svojstva i opisuje modifikacije modeliranog protokola u odnosu na specifikacije. Konačno, kraj rada opisuje rezultate analize te pokazuje važnost određenih dijelova protokola u ostvarivanju njegovih ključnih sigurnosnih svojstava.

2. X3DH protokol

2.1. Uvod

X3DH (Extended Triple Diffie-Hellman) je protokol za uspostavljanje zajedničkog tajnog ključa između dvije strane koristeći javne ključeve za autentifikaciju. Osmišljen je za asinkrono okruženja, te omogućava da jedna strana (Alice) uspostavi zajednički ključ s drugom stranom (Bob) koja u tom trenutku ne mora biti prisutna. Bob objavljuje svoje ključeve na poslužitelju, a Alice koristi te ključeve za izračunavanje zajedničkog tajnog ključa. X3DH osigurava unaprijednu povjerljivost i kriptografsko poricanje koristeći Diffie-Hellman protokol zajedno s privremenim i dugotrajnim ključevima [1].

2.2. Opis protokola

2.2.1. Kriptografska notacija

- Konkatenacija nizova bajtova X i Y označava se kao $X||Y$.
- $DH(PK1, PK2)$ predstavlja niz bajtova koji je zajednička tajna dobivena iz Diffie-Hellman protokola ključevima $PK1$ i $PK2$, gdje je prva vrijednost privatni ključ koji odgovara javnom ključu $PK1$, a druga vrijednost je javni ključ $PK2$.
- $\text{Sig}(PK, M)$ predstavlja potpis niza bajtova M koji se verificira javnim ključem PK, a kreira se potpisivanjem M s pripadajućim privatnim ključem od PK.
- $KDF(M)$ predstavlja izlaz iz HKDF (hash-based key derivation function) za ulazni niz bajtova M.

2.2.2. Ključevi

Naziv	Oznaka	Opis
Alicein identifikacijski ključ	IK_A	Dugotrajni ključ koji se koristi za identifikaciju Alice.
Alicein privremeni ključ	EK_A	Kratkoročni ključ koji Alice generira u jednoj instanci X3DH protokola.
Bobov identifikacijski ključ	IK_B	Dugotrajni ključ Boba koji se koristi za identifikaciju.
Bobov potpisani predključ	SPK_B	Dugotrajni ključ koji Bob periodično mijenja.
Bobov jednokratni predključ	OPK_B	Jednokratni ključ koji Bob koristi u jednoj instanci X3DH protokola.

Tablica 2.1. Ključevi korišteni u X3DH protokolu

2.2.3. Uloge

X3DH protokol uključuje tri glavne uloge [1]:

- **Alice**: Klijent koji inicira komunikaciju. Alice koristi Bobove javne ključeve za izračunavanje zajedničkog tajnog ključa. U procesu generira privremeni ključ (EK_A) i koristi ga za Diffie-Hellman izračune s Bobovim ključevima.
- **Bob**: Klijent koji može biti *offline* i unaprijed objavljuje svoje ključeve na poslužitelju. Bobovi ključevi uključuju identifikacijski ključ (IK_B), potpisani predključ (SPK_B) i jednokratni predključ (OPK_B).
- **Poslužitelj**: Djeluje kao posrednik koji pohranjuje Bobove ključeve i omogućuje Alice pristup tim ključevima za uspostavljanje sigurnog komunikacijskog kanala.

2.2.4. Pregled protokola

X3DH protokol ima tri faze [1]:

1. Bob objavljuje svoj identifikacijski ključ i predključeve na poslužitelju.
2. Alice preuzima "paket predključeva" s poslužitelja i koristi ga za slanje inicijalne

poruke Bobu.

3. Bob prima i obrađuje Aliceinu inicijalnu poruku.

Objavljivanje ključeva

Bob objavljuje niz javnih ključeva na poslužitelju:

- Identifikacijski ključ (IK_B)
- Dugotrajni predključ (SPK_B) i njegov potpis ($\text{Sig}(IK_B, SPK_B)$)
- Skup jednokratnih predključeva ($OPK_B^1, OPK_B^2, OPK_B^3, ...$)

Bob treba samo jednom prenijeti svoj identifikacijski ključ na poslužitelja. Međutim, može prenijeti nove jednokratne predključeve u bilo kojem trenutku (npr. kad poslužitelj obavijesti Boba da je ponestalo jednokratnih predključeva). Također, Bob povremeno prenosi novi dugotrajni predključ i njegov potpis (npr. jednom tjedno ili mjesečno), zamjenjujući prethodni. Na kraju, Bob treba izbrisati te privatni ključeve kako bi osigurao unaprijednu povjerljivost (privatni predključevi se brišu kako Bob prima poruke koje koriste te ključeve).

Slanje inicijalne poruke

Alice kontaktira poslužitelja i preuzima "paket predključeva". Ako su svi Bobovi jednokratni predključevi već iskorišteni, paket neće sadržavati jednokratni predključ. Alice verificira potpis dugotrajnog predključa i prekida protokol ako verifikacija ne uspije. Zatim generira privremeni ključ EK_A. Ako paket ne sadrži jednokratni predključ, Alice računa:

$$
\begin{aligned}
DH1 &= DH(IK_A, SPK_B) \\
DH2 &= DH(EK_A, IK_B) \\
DH3 &= DH(EK_A, SPK_B) \\
SK &= KDF(DH1||DH2||DH3)
\end{aligned}
$$

Ako paket sadrži jednokratni predključ, izračun uključuje dodatni DH parametar:

$$DH4 = DH(EK_A, OPK_B)$$
$$SK = KDF(DH1||DH2||DH3||DH4)$$

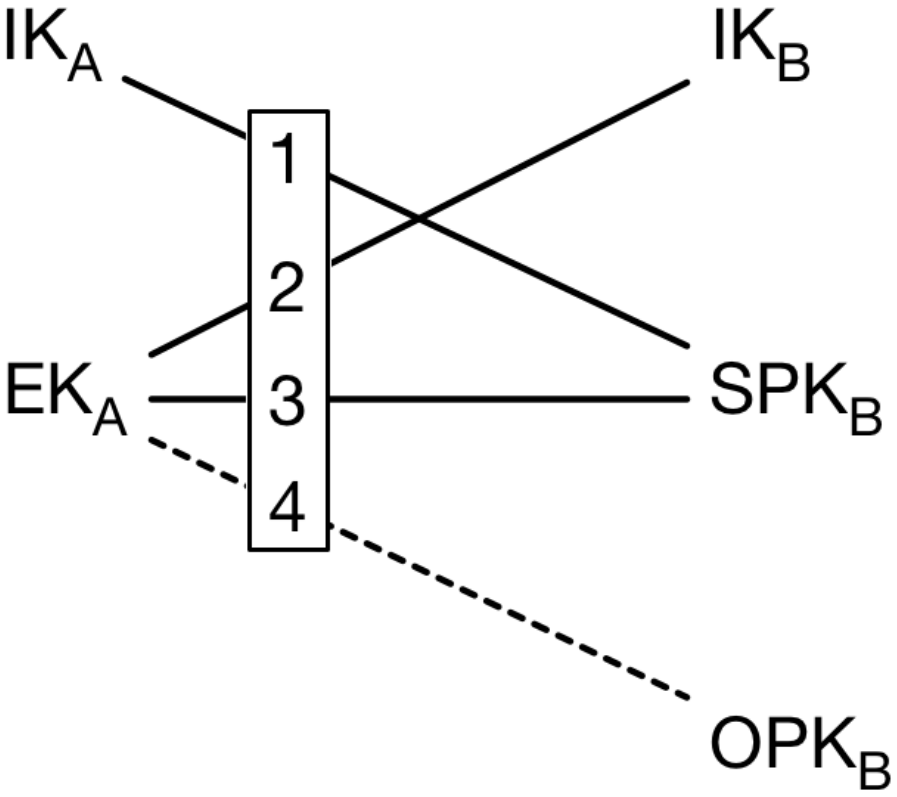
Nakon izračuna SK, Alice izračunava niz bajtova "povezanih podataka" AD koji sadrži identifikacijske informacije za obje strane:

$$AD = IK_A||IK_B$$

Alice zatim šalje Bobu početnu poruku koja sadrži:

- Alicein identifikacijski ključ IK_A
- Alicein privremeni ključ EK_A
- Početni tekst enkriptiran AEAD (authenticated encryption with associated data) shemom koristeći ključ SK i AD kao povezane podatke.

Nakon slanja inicijalne poruke, Alice može nastaviti koristiti SK ili ključeve izvedene iz SK za komunikaciju s Bobom.



Slika 2.1. Generiranje $DH1, \ldots, DH4$ u X3DH protokolu [1]

Primanje inicijalne poruke

Nakon primanja inicijalne poruke od Alice, Bob uzima ključeve iz poruke i koristeći njih, ponavlja Diffie-Hellman i KDF izračune iz prethodnog odjeljka kako bi izveo ključ SK. Bob zatim konstruira niz bajtova AD, kao što je opisano u prethodnom odjeljku, te pokušava dekriptirati početni tekst koristeći ključ SK i podatke AD. Ako dekripcija početnog teksta ne uspije, Bob prekida protokol. Ako se početni tekst uspješno dekriptira, protokol je završen i uspješno je uspostavljen zajednički ključ SK.

2.3. Sigurnosna svojstva protokola

X3DH protokol nudi sljedeća sigurnosna svojstva [1]:

1. **Uzajamna autentikacija (engl.*Mutual authentication*)**: uzajamna autentikacija između Alice i Boba postignuta je njihovim identifikacijskim ključevima tijekom izračuna vrijednosti $DH1$ i $DH2$ kao što je opisano u poglavlju 2.2.4.

2. **Unaprijedna povjerljivost (engl.*Forward secrecy*)**: zajamčuje da kompromitacija dugoročnih identifikacijskih ključeva ne ugrožava sigurnost prethodnih komunikacija. Postignuta je izračunom vrijednosti $DH3$ i $DH4$ koje koriste privremene predključeve.

3. **Poricanje (engl.*Deniability*)**: protokol ne pruža ni Alice ni Bobu kriptografski dokaz o sadržaju njihove komunikacije ili činjenici da su komunicirali, što znači da obje strane mogu negirati da su imale komunikaciju.

3. PQXDH protokol

3.1. Uvod

PQXDH (Post-Quantum Extended Diffie-Hellman) nadogradnja je X3DH protokola koja uključuje post-kvantne kriptografske tehnike kako bi osigurala otpornost na kvantne napade. PQXDH koristi post-kvantne mehanizme za enkapsulaciju ključeva (PQKEM), kao što je Crystals-Kyber[2], za dodatnu sigurnost. Glavna razlika između PQXDH i X3DH protokola je dodatni korak za generiranje zajedničke tajne pomoću PQKEM. PQXDH pruža post-kvantnu unaprijednu povjerljivost i oblik kriptografskog poricanja, ali se i dalje oslanja na težinu problema diskretnog logaritma za međusobnu autentikaciju [3].

3.2. Opis protokola

3.2.1. Kriptografska notacija

Notaciju iz 2.2.1. proširujemo s pripadnim funkcijama iz PQKEM:

- $(CT, SS) =$ PQKEM-ENC(PK) predstavlja dva niza bajtova (šifrat i zajedničku tajnu) koji nastaju kao rezultat enkapsulacije javnog ključa PK.
- $SS =$ PQKEM-DEC(PK, CT) predstavlja zajedničku tajnu SS koja se dobije dekapsulacijom šifrata CT koristeći privatni ključ koji odgovara javnom ključu PK korištenom tijekom enkapsulacije.

3.2.2. Ključevi

Ključevima iz 2.2.2. dodajemo ključeve za post-kvantnu enkapsulaciju.

Naziv	Oznaka	Opis
Bobov potpisani *last-resort* postkvantni predključ	PQSPK_B	Dugotrajni ključ koji Bob povremeno mijenja i potpisuje svojim identifikacijskim ključem.
Bobov potpisani jednokratni postkvantni predključ	PQOPK_B	Jednokratni ključ koji se koristi u jednoj instanci PQXDH protokola i potpisuje se Bobovim identifikacijskim ključem.

Tablica 3.1. Ključevi korišteni u PQXDH protokolu

3.2.3. Pregled protokola

Koraci protokola ostaju isti kao što je opisano u 2.2.4., ali se koriste novi PQKEM ključevi i vrše se dodatni izračuni tijekom slanja i primanja inicijalne poruke [3].

Objavljivanje ključeva

Bob objavljuje niz javnih ključeva na poslužitelju:

- Bobov identifikacijski ključ IK_B
- Bobov predključ SPK_B i njegov potpis ($\text{Sig}(IK_B, SPK_B)$)
- Bobov *last-resort* PQKEM predključ $PQSPK_B$ i njegov potpis ($\text{Sig}(IK_B, PQSPK_B)$)
- Skup Bobovih jednokratnih predključeva ($OPK_B^1, OPK_B^2, OPK_B^3, ...$)
- Skup Bobovih jednokratnih PQKEM predključeva ($PQOPK_B^1, PQOPK_B^2, PQOPK_B^3, ...$) i njihovi potpisi ($\text{Sig}(IK_B, PQOPK_B^1), \text{Sig}(IK_B, PQOPK_B^2), \text{Sig}(IK_B, PQOPK_B^3), ...$)

Slanje inicijalne poruke

Alice kontaktira poslužitelja i preuzima "paket predključeva" koji sadrži sljedeće vrijednosti:

- Bobov identifikacijski ključ IK_B
- Dugotrajni predključ (SPK_B) i njegov potpis ($\text{Sig}(IK_B, SPK_B)$)

- Jedan od Bobovih potpisanih jednokratnih PQKEM predključeva ($PQOPK_B$) ili potpisani *last-resort* PQKEM predključ ($PQSPK_B$), ako ne postoji nijedan neiskorišteni jednokratni predključ
- Bobov jednokratni predključ (OPK_B)

Alice verificira potpise predključeva. Ako bilo koja provjera potpisa ne uspije, Alice prekida protokol. Ako sve provjere potpisa uspiju, Alice generira privremeni ključ EK_A. Alice dodatno generira zajedničku tajnu enkapsulacijom ključa $PQOPK_B$:

$$(CT, SS) = \text{PQKEM-ENC}(PQOPK_B)$$

Ako se u paketu ne nalazi jednokratni PQKEM predključ, Alice koristi *last-resort* PQKEM predključ ($PQSPK_B$):

$$(CT, SS) = \text{PQKEM-ENC}(PQSPK_B)$$

Ako paket ne sadrži jednokratni predključ, Alice računa:

$$\begin{aligned} DH1 &= DH(IK_A, SPK_B) \\ DH2 &= DH(EK_A, IK_B) \\ DH3 &= DH(EK_A, SPK_B) \\ SK &= KDF(DH1||DH2||DH3||SS) \end{aligned}$$

Ako paket sadrži jednokratni predključ, izračun uključuje dodatni DH parametar:

$$\begin{aligned} DH4 &= DH(EK_A, OPK_B) \\ SK &= KDF(DH1||DH2||DH3||DH4||SS) \end{aligned}$$

Alice tada izračunava "povezane podatke" AD koji sadrže identifikacijske informacije za obje strane:

$$AD = IK_A||IK_B$$

Alice zatim šalje Bobu početnu poruku koja sadrži:

- Alicein identifikacijski ključ IK_A
- Alicein privremeni ključ EK_A
- PQKEM šifrat CT
- Početni kriptirani tekst enkriptiran AEAD shemom koristeći AD kao povezane podatke i enkripcijski ključ SK

Primanje inicijalne poruke

Nakon primanja inicijalne poruke od Alice, Bob uzima ključeve iz poruke i koristeći njih, ponavlja Diffie-Hellman, PQKEM i KDF izračune iz prethodnog koraka kako bi izveo ključ SK. Bob zatim konstruira niz bajtova AD, kao što je opisano u prethodnom odjeljku, te pokušava dekriptirati početni tekst koristeći ključ SK i podatke AD.

4. Pregled postojeće sigurnosne analize PQXDH protokola pomoću alata ProVerif i CryptoVerif

Istraživači Karthikeyan Bhargavan (Cryspen), Charlie Jacomme (Inria), Franziskus Kiefer (Cryspen) i Rolfe Schmidt (Signal) modelirali su PQXDH protokol pomoću alata za formalnu verifikaciju ProVerif i CryptoVerif. Kroz ovu analizu identificirano je nekoliko potencijalnih problema [4]:

- **Napad konfuzije javnih ključeva**: Otkrivena je mogućnost napada u kojem bi došlo do zabune između javnih ključeva za Diffie-Hellman i PQKEM, što bi moglo ugroziti sigurnost PQXDH. Analizom je utvrđeno da je trenutna implementacija Signala otporna na ovu vrstu napada jer veličine javnih ključeva ne odgovaraju i jer su ključevi označeni bajtom koji identificira algoritam.

- **Slaba post-kvantna unaprijedna povjerljivost**: Post-kvantna unaprijedna povjerljivost PQXDH protokola oslanja se na to da klijent koji šalje inicijalnu poruku koristi potpisani jednokratni PQKEM predključ, dok klijent koji odgovara na poruku briše odgovarajući privatni ključ. Međutim, budući da su i *PQOPK* i potpisani *last-resort* predključ *PQSPK* potpisani istim identifikacijskim ključem i imaju isti format, klijent koji šalje poruku ne može znati koristi li se *last-resort* predključ ili jednokratni predključ. Kao posljedica toga, PQXDH nudi slabiju unaprijednu povjerljivost, budući da klijent koji prima inicijalnu poruku ne može znati hoće li privatni ključ biti odmah izbrisan.

- **Napad konfuzije KDF ulaza**: Postoji zabrinutost da se format KDF ulaza između X3DH i PQXDH može zamijeniti. Na primjer, X3DH može koristiti 3 ili 4 DH za-

jedničke tajne spojene kao ulaz u KDF funkciju, dok PQXDH spaja ili 3 Diffie-Hellman i 1 KEM zajedničku tajnu, ili 4 Diffie-Hellman i 1 KEM zajedničku tajnu. Unutar PQXDH protokola, info polje sprječava zabune, ali može postojati napad između PQXDH ($3DH + SS$) i X3DH ($4DH$) protokola ovisno o sadržaju info polja u X3DH, budući da X3DH specifikacija ne definira sadržaj info polja.

- **Napad ponovnog enkapsuliranja ključa**: Čak i uz IND-CCA (INDistinguishability under Chosen Ciphertext Attack) sigurne KEM-ove, otkriveno je da napadač može kompromitirati sve ostale post-kvantne predključeve ako je jedan predključ kompromitiran. Analizom je utvrđeno da ovaj napad nije moguć kada se koristi KEM poput Crystals-Kyber koji ima svojstvo da je zajednička tajna (engl.*shared secret*) snažno povezana s javnim ključem.

Na temelju pronađenih ranjivosti ažurirana je specifikacija protokola, a nova revizija protokola ponovno je modeliran te je dokazano da ne sadrži iste, prethodno spomenute, ranjivosti. Potencijalne ranjivosti pronađene u ProVerif i CryptoVerif modelima neće biti analizirane u naknadnom Tamarin modelu jer:

- **Popravljene su u najnovijoj reviziji protokola**: Najnovija verzija specifikacije PQXDH protokola uključuje popravke i izmjene koje adresiraju sve ranjivosti otkrivene tijekom analize pomoću alata ProVerif i CryptoVerif.

- **Ne mogu se sve analizirati zbog ograničenja alata Tamarin**: Tamarin prover, iako vrlo moćan alat za formalnu verifikaciju, ima određena ograničenja koja onemogućuju analizu svih sigurnosnih aspekata PQXDH protokola. Na primjer, Tamarin ne podržava definiranje veličine ključeva, što je ključno za analizu nekih specifičnih napada poput onih koji se oslanjaju na konfuziju između različitih vrsta ključeva.

- **Tamarin model će se uglavnom fokusirati na utvrđivanje ključnih sigurnosnih svojstava PQXDH protokola**: Glavni cilj Tamarin modela bit će potvrđivanje i dokazivanje ključnih sigurnosnih svojstava PQXDH protokola, kao što su unaprijedna povjerljivost, autentikacija i integritet.

5. Sigurnosna analiza PQXDH protokola pomoću alata Tamarin

5.1. Tamarin Prover

Tamarin prover napredni je alat za formalnu verifikaciju sigurnosnih protokola koji validira sigurnosna svojstva protiv aktivnog mrežnog protivnika koji može čitati, presretati, modificirati, ponavljati i slati bilo koju poruku. Tamarin podržava niz ugrađenih jednadžbi za modeliranje različitih kriptografskih primitiva. Korisnici također mogu definirati vlastite jednadžbe. Tamarin koristi pravila (engl.*rules*) kako bi modelirao ponašanje sudionika u protokolu.

Pravila (engl.*Rules*) se sastoje od premise i zaključka, koji se sastoje od (potencijalno trajnih) činjenica. Da bi se pravilo primijenilo, činjenice u njegovoj premisi moraju biti pronađene u trenutnom stanju (engl.*state*). Kada se pravilo primijeni, sve linearne (netrajne) činjenice iz premise uklanjaju se iz stanja i zamjenjuju činjenicama iz zaključka pravila. Tamarin prover koristi leme (engl.*lemmas*) i akcijske činjenice (engl.*action facts*) za dokazivanje sigurnosnih svojstava kriptografskih protokola.

Leme (engl.*Lemmas*) su formalni izrazi koji specificiraju sigurnosna svojstva koje protokol treba zadovoljavati. One su definirane kao logičke formule koje Tamarin pokušava dokazati ili opovrgnuti. Leme mogu biti, na primjer, iskazi o povjerljivosti (da određena tajna nije dostupna protivniku) ili autentikaciji (da određena poruka dolazi od legitimnog pošiljatelja).

Akcijske činjenice (engl.*Action facts*) koriste se za označavanje specifičnih događaja unutar protokola. Svaka akcijska činjenica predstavlja neku značajnu akciju, poput slanja ili primanja poruke, stvaranja ključa ili uspješne autentikacije. Svaka akcijska činjenica je povezana s određenim pravilom u Tamarinu. Kada se pravilo primijeni, gene-

riraju se odgovarajuće akcijske činjenice koje ne postaju dio stanja, ali se zapisuju u trag (engl.*trace*). Ove činjenice mogu biti uvjeti za leme, što omogućava složeno modeliranje i analizu protokola.

Korištenje lemi i akcijskih činjenica omogućuje korisnicima da precizno specificiraju sigurnosna svojstva i ponašanje protokola. Tamarin zatim koristi ove specifikacije kako bi provjerio može li napadač ugroziti definirana sigurnosna svojstva ili ne. Ako lema nije zadovoljena, Tamarin generira napadačke scenarije koji pomažu u identifikaciji i razumijevanju potencijalnih ranjivosti protokola. Kombinacija pravila, lemi i akcijskih činjenica čini Tamarin moćnim alatom za formalnu verifikaciju sigurnosnih protokola, omogućujući dubinsku analizu i dokazivanje sigurnosnih svojstava u prisutnosti različitih napadačkih scenarija [5][6].

5.2. Opis modela

Model za analizu PQXDH protokola koristi posebna pravila za korisnike Bob i Alice, simulirajući njihove uloge i interakcije unutar protokola. Zbog sporosti verifikacije protokola, umjesto Diffie-Hellman izračuna koji su prisutni u specifikaciji, model koristi samo PQKEM izračune za generiranje potrebnih ulaza u KDF funkciju. Ovo uključuje sljedeće korake:

$$CT1, SS1 = \text{PQKEM-ENC}(IK_B)$$
$$CT2, SS2 = \text{PQKEM-ENC}(OPK_B)$$
$$CT3, SS3 = \text{PQKEM-ENC}(PQOPK_B)$$
$$SK = KDF(SS1||SS2||SS3)$$

Korištenjem isključivo KEM izračuna gubimo svojstvo međusobne autentikacije koje je bilo prisutno tijekom Diffie-Hellman izračuna. Ako koristimo KEM samo klijent koji šalje poruku može potvrditi identitet primatelja, dok primatelj ne može potvrditi identitet klijenta koji šalje inicijalnu poruku. Ova slabost nije kritična u kontekstu post-kvantne sigurnosti jer tradicionalni Diffie-Hellman protokol, koji je korišten u specifikaciji, ne pruža dovoljnu zaštitu protiv post-kvantnih napadača.

Unatoč tome, model je modificiran kako bi očuvao svojstvo međusobne autentikacije.

To svojstvo je osigurano potpisivanjem inicijalne poruke identifikacijskim ključem korisnika Alice. U takvom scenariju, Bob je implicitno autenticiran jer samo on može svojim privatnim ključevima napraviti uspješnu KEM dekapsulaciju i dobiti iste zajedničke tajne koje su dobivene tijekom enkapsulacije. Alice je eksplicitno autenticirana validacijom njezinog potpisa nad inicijalnom porukom. Ovime postižemo djelomično post-kvantnu međusobnu autentikaciju u kojoj samo jedan korisnik (primatelj poruke) može biti autenticiran post-kvantnom metodom. Ako umjesto algoritma digitalnog potpisivanja baziranog na eliptičkim krivuljama koristimo neki od post-kvantnih algoritama, kao što je Dilithium[7], svojstvo međusobne autentikacije postaje u potpunosti post-kvantno.

Model tijekom objavljivanja predključeva kreira *"last-resort"* PQKEM predključ, kao što je definirano u specifikaciji, ali taj predključ se nikada ne koristi u izračunima jer je uvijek prisutan jednokratni PQKEM predključ. Također, model simulira protokol do trenutka kad klijenti uspostave zajedničku tajnu i ne simulira naknadnu razmjenu poruka. Ovo omogućuje detaljnu analizu inicijalne faze uspostave sigurnosnog protokola, što je kritično za osiguravanje sigurne komunikacije.

Objavljivanje i dohvaćanje javnih identifikacijskih ključeva i predključeva se vrši pomoću činjenica, ali prijenos inicijalne poruke (slanje i primanje) se vrši samo preko mrežne komunikacije (`In()` i `Out()`). Ovime pretpostavljamo da postoji sigurni komunikacijski kanal pomoću kojeg klijenti mogu objaviti i dohvatiti sve potrebne ključeve. Dodatno, model omogućuje da se klijenti samo jednom inicijaliziraju i kreiraju identifikacijske ključeve, ali klijenti mogu bilo kada kreirati nove predključeve kako bi uspostavili novi zajednički ključ.

5.2.1. Funkcije i jednadžbe

U modelu su definirane sljedeće funkcije i jednadžbe:

- `functions: kem_encapsulate/2, kem_decapsulate/2, kdf/1`
- `equations: kem_decapsulate(kem_encapsulate(shared_secret, pk(secret_key)), secret_key) = shared_secret`

Tamarin ne dozvoljava funkcije koje vraćaju više od jedne vrijednosti [6]. Zbog toga se KEM ne može u potpunosti simulirati na način kako je definirano u teoriji, gdje KEM

vraća dvije vrijednosti: šifrat i zajedničku tajnu. Kako bismo zaobišli ovo ograničenje, KEM funkcije (`kem_encapsulate` i `kem_decapsulate`) su napravljene tako da primaju dva argumenta. Prvi argument kod enkapsulacije je obavezno nasumično generirana zajednička tajna, dok je drugi argument javni ključ. Enkapsulacija također vraća samo jednu vrijednosti, kriptirani tekst. Funkcija za dekapsulaciju kao prvi argument prima kriptirani tekst, a kao drugi privatni ključ. Dekapsulacija vraća istu zajedničku tajnu koja je korištena tijekom enkapsulacije što možemo vidjeti definirano u jednadžbi. Na ovaj način, iako Tamarin ima ograničenja kod definiranja funkcija koje vraćaju više vrijednosti, moguće je simulirati ponašanje KEM-a i sačuvati njegova sigurnosna svojstva.

Iako je moguće koristiti ugrađenu funkciju za izračun kriptografskog sažetka (engl.*hashing*) koristimo vlastitu funkciju `kdf` koja omogućuje derivaciju ključeva iz kombinacije zajedničkih tajni, osiguravajući kriptografski jake ključeve za daljnju uporabu u protokolu.

5.2.2. Restrikcije

U Tamarinu, restrikcije (engl.*restrictions*) su moćan mehanizam koji omogućuje precizno kontroliranje ponašanja modela i sigurnosnih svojstava koja se analiziraju. Restrikcije se koriste za definiranje uvjeta koji moraju biti ispunjeni u svakom validnom tragu (engl.*trace*) protokola. Ako bilo koja putanja modela krši definiranu restrikciju, Tamarin će tu putanju zanemariti [6]. U modelu definiramo tri restrikcije:

- **Restrikcija jednakosti** (`Equality`): Ova restrikcija provjerava jesu li dva izraza jednaka. Služi za verifikaciju potpisa nad ključevima i zanemaruje sve putanje gdje su potpisi neuspješno validirani. Definira se kao:

$$\forall x, y, \#i \ (Eq(x, y)@\#i \ \Rightarrow \ x = y) \tag{5.1}$$

- **Restrikcija inicijalizacije klijenta Alice** (`OnlyOnceInitAlice`): Ova restrikcija osigurava da se Alice inicijalizira samo jednom tijekom izvršavanja protokola. Ključna je za sprečavanje višestruke inicijalizacije Alice, što bi moglo dovesti do

nekonzistentnih stanja. Definira se kao:

$$\forall \#i, \#j \ (OnlyOnceInitAlice()@\#i \wedge OnlyOnceInitAlice()@\#j \ \Rightarrow \ \#i = \#j) \tag{5.2}$$

- **Restrikcija inicijalizacije klijenta Boba** (`OnlyOnceInitBob`): Slična prethodnoj restrikciji, ova osigurava da se Bob inicijalizira samo jednom tijekom izvršavanja protokola. Definira se kao:

$$\forall \#i, \#j \ (OnlyOnceInitBob()@\#i \wedge OnlyOnceInitBob()@\#j \ \Rightarrow \ \#i = \#j) \tag{5.3}$$

5.2.3. Pravila (engl.*Rules*)

Pravila za generiranje i dohvat ključeva

U ovom modelu definirana su specifična pravila koja upravljaju generiranjem i dohvatom kriptografskih ključeva za sudionike Bob i Alice. Ova pravila osiguravaju ispravno inicijaliziranje ključeva i njihovu razmjenu, čime se omogućuje sigurna komunikacija unutar protokola. Sljedeća pravila opisuju postupke generiranja identifikacijskih ključeva, dohvaćanja javnih ključeva, te generiranja predključeva:

- **Pravilo inicijalizacije Boba** (`init_bob`): Ovo pravilo inicijalizira Bobov identifikacijski ključ. Sadrži akcijsku činjenicu `OnlyOnceInitBob()` koja uz restrikciju 5.3 osigurava da Bob može kreirati samo jedan identifikacijski ključ.

```
rule init_bob:
  [ Fr(~curve_identity_key) ]
  --[InitBob(), OnlyOnceInitBob()]->
  [ !KeysBob(~curve_identity_key) ]
```

- **Pravilo dohvaćanja javnog ključa Boba** (`get_public_key_bob`): Ovo pravilo dohvaća javni identifikacijski ključ Boba i šalje ga u mrežu.

```
rule get_public_key_bob:
  let
    curve_identity_key_public = pk(curve_identity_key)
```

```
  in
  [ !KeysBob(curve_identity_key) ]
  --[]->
  [ !PublicKeyBob(curve_identity_key_public),
    Out(curve_identity_key_public) ]
```

- **Pravilo generiranja predključeva Boba** (`generate_prekeys_bob`): Ovo pravilo generira Diffie-Hellman i PQKEM predključeve za Boba. U zaključku se nalaze trajna činjenica `PrekeysBob` koja će se koristiti u više naknadnih pravila i ekvivalentna linearna činjenica `PublishablePrekeysBob` koja se koristi samo u pravilu za objavljivanje predključeva.

```
rule generate_prekeys_bob:
  [ Fr(~curve_prekey), Fr(~pqkem_prekey),
    !KeysBob(curve_identity_key) ]
  --[GeneratePrekeysBob()]->
  [ !PrekeysBob(~curve_prekey, ~pqkem_prekey),
    PublishablePrekeysBob(~curve_prekey, ~pqkem_prekey) ]
```

- **Pravilo inicijalizacije Alice** (`init_alice`): Ovo pravilo inicijalizira Alicein identifikacijski ključ. Sadrži akcijsku činjenicu `OnlyOnceInitAlice()` koja uz restrikciju 5.2 osigurava da Alice može kreirati samo jedan identifikacijski ključ.

```
rule init_alice:
  [ Fr(~curve_identity_key) ]
  --[InitAlice(), OnlyOnceInitAlice()]->
  [ !KeysAlice(~curve_identity_key) ]
```

- **Pravilo dohvaćanja javnog ključa Alice** (`get_public_key_alice`): Ovo pravilo dohvaća javni identifikacijski ključ Alice i šalje ga u mrežu.

```
rule get_public_key_alice:
  let
    curve_identity_key_public = pk(curve_identity_key)
```

```
  in
  [ !KeysAlice(curve_identity_key) ]
  --[]->
  [ !PublicKeyAlice(curve_identity_key_public),
     Out(curve_identity_key_public) ]
```

Pravila za otkrivanje privatnih ključeva

U modelu su definirana specifična pravila za otkrivanje privatnih ključeva sudionika. Ova pravila omogućuju simulaciju scenarija u kojima napadač dolazi do privatnih ključeva, što je ključno za analizu svojstva unaprijedne povjerljivosti.

- **Pravilo otkrivanja identifikacijskog ključa Boba** (`reveal_identity_key_bob`): Ovo pravilo omogućuje otkrivanje Bobovog privatnog identifikacijskog ključa. Kada se primjeni, privatni identifikacijski ključ šalje se u mrežu.

```
rule reveal_identity_key_bob:
  [ !KeysBob(curve_identity_key) ]
  --[RevealIdentityKeyBob()]->
  [ Out(curve_identity_key) ]
```

- **Pravilo otkrivanja predključeva Boba** (`reveal_prekeys_bob`): Ovo pravilo omogućuje otkrivanje Bobovih privatnih predključeva. Kada se primjeni, privatni predključevi se šalju u mrežu.

```
rule reveal_prekeys_bob:
  [ !PrekeysBob(curve_prekey, pqkem_prekey) ]
  --[RevealPrekeysBob()]->
  [ Out(<curve_prekey, pqkem_prekey>) ]
```

Pravilo za objavljivanje predključeva na poslužitelju

U modelu je definirano pravilo koje upravlja postupkom objavljivanja predključeva na poslužitelju. Ovo pravilo omogućuje Bobu da objavi svoje PQKEM javne predključeve zajedno s odgovarajućim identifikatorima i potpisima. Na taj način, Alice može koristiti

ove ključeve za sigurno uspostavljanje komunikacije.

Pravilo objavljivanja ključeva (`publish_keys`): Ovo pravilo objavljuje Bobove predključeve na poslužitelju.

```
rule publish_keys:
  let
    curve_identity_key_public = pk(curve_identity_key)

    curve_prekey_public = pk(curve_prekey)
    curve_prekey_identifier = h(curve_prekey_public)
    curve_prekey_signature = sign(
      curve_prekey_public, curve_identity_key
    )

    pqkem_prekey_public = pk(pqkem_prekey)
    pqkem_prekey_identifier = h(pqkem_prekey_public)
    pqkem_prekey_signature = sign(
      pqkem_prekey_public, curve_identity_key
    )
  in
  [ !KeysBob(curve_identity_key),
  PublishablePrekeysBob(curve_prekey, pqkem_prekey) ]
  --[PublishKeys()]->
  [ PublishedKeys(curve_identity_key_public, curve_prekey_public,
                  curve_prekey_identifier, curve_prekey_signature,
                  pqkem_prekey_public, pqkem_prekey_identifier,
                  pqkem_prekey_signature),
    Out(<curve_identity_key_public, curve_prekey_public,
         curve_prekey_identifier, curve_prekey_signature,
         pqkem_prekey_public, pqkem_prekey_identifier,
         pqkem_prekey_signature>) ]
```

U ovom pravilu, Bob generira javne ključeve za svoj identifikacijski ključ i predključeve. Svaki predključ je identificiran pomoću kriptografskog sažetka javnog ključa i potpisan Bobovim identifikacijskim ključem. Nakon što su svi potrebni podaci generirani, oni se objavljuju na poslužitelju. Objavljivanje je prikazano kreiranjem nove linearne činjenice koja sadrži sve potrebne ključeve i slanjem istih ključeva u mrežu.

Pravilo za slanje inicijalne poruke

U modelu je definirano pravilo koje simulira slanja inicijalne poruke. Ovo pravilo omogućuje Alice da generira potrebne zajedničke tajne te da pošalje inicijalnu poruku Bobu, koristeći objavljene predključeve.

Pravilo slanja inicijalne poruke (`send_initial_message`): Ovo pravilo simulira generiranje i slanje inicijalne poruke.

```
rule send_initial_message:
  let
    kem_ciphertext_1 = kem_encapsulate(~shared_secret_1,
                                       curve_identity_key_public)
    kem_ciphertext_2 = kem_encapsulate(~shared_secret_2,
                                       curve_prekey_public)
    kem_ciphertext_3 = kem_encapsulate(~shared_secret_3,
                                       pqkem_prekey_public)

    secret_key = kdf(
      <~shared_secret_1, ~shared_secret_2, ~shared_secret_3>
    )

    curve_identity_key_public_alice = pk(curve_identity_key)
    ephemeral_key_public = pk(~ephemeral_key)

    initial_ciphertext = senc(~initial_plaintext, secret_key)

    exchange = <curve_identity_key_public_alice, ephemeral_key_public,
```

```
              kem_ciphertext_1, kem_ciphertext_2, kem_ciphertext_3,
              initial_ciphertext>
  exchange_signature = sign(exchange, curve_identity_key)
in
[ !KeysAlice(curve_identity_key),
  PublishedKeys(curve_identity_key_public, curve_prekey_public,
                curve_prekey_identifier, curve_prekey_signature,
                pqkem_prekey_public, pqkem_prekey_identifier,
                pqkem_prekey_signature),
  Fr(~ephemeral_key),
  Fr(~shared_secret_1),
  Fr(~shared_secret_2),
  Fr(~shared_secret_3),
  Fr(~initial_plaintext) ]
--[Eq(verify(curve_prekey_signature,
            curve_prekey_public,
            curve_identity_key_public), true),
   Eq(verify(pqkem_prekey_signature,
            pqkem_prekey_public,
            curve_identity_key_public), true),
   SendInitialMessage(
     ~initial_plaintext, initial_ciphertext, secret_key
   )]->
[ Out(<exchange, exchange_signature>) ]
```

U ovom pravilu, Alice generira kriptografske vrijednosti potrebne za uspostavljanje sigurne komunikacije s Bobom. Konkretno, pravilo `send_initial_message` uključuje:

1. Verifikaciju potpisa predključeva:

```
Eq(verify(curve_prekey_signature,
          curve_prekey_public,
          curve_identity_key_public), true)
```

```
Eq(verify(pqkem_prekey_signature,
          pqkem_prekey_public,
          curve_identity_key_public), true)
```

2. Enkapsulaciju identifikacijskog ključa i predključeva Boba:

```
kem_ciphertext_1 = kem_encapsulate(
  ~shared_secret_1, curve_identity_key_public
)
kem_ciphertext_2 = kem_encapsulate(
  ~shared_secret_2, curve_prekey_public
)
kem_ciphertext_3 = kem_encapsulate(
  ~shared_secret_3, pqkem_prekey_public
)
```

3. Derivaciju tajnog ključa korištenjem KDF i generiranih zajedničkih tajni:

```
secret_key = kdf(
  <~shared_secret_1, ~shared_secret_2, ~shared_secret_3>
)
```

4. Generiranje javnih ključeva za Alicein identifikacijski ključ i jednokratni (engl.*ephemeral*) ključ:

```
curve_identity_key_public_alice = pk(curve_identity_key)
ephemeral_key_public = pk(~ephemeral_key)
```

5. Enkripciju nasumično generirane inicijalne poruke korištenjem deriviranog tajnog ključa pomoću simetrične enkripcijske sheme:

```
initial_ciphertext = senc(~initial_plaintext, secret_key)
```

6. Formiranje poruke koja uključuje generirane vrijednosti:

```
exchange = <curve_identity_key_public_alice, ephemeral_key_public,
            kem_ciphertext_1, kem_ciphertext_2, kem_ciphertext_3,
            initial_ciphertext>
```

7. Slanje poruke u mrežu:

```
Out(exchange)
```

Pravilo za primanje inicijalne poruke

U modelu je definirano pravilo koje prikazuje postupak primanja inicijalne poruke. Ovo pravilo vrši dekripciju i obradu inicijalne poruke.

Pravilo primanja inicijalne poruke (`receive_initial_message`): Ovo pravilo prikazuje primanje i dekripciju inicijalne poruke koju je Alice poslala Bobu.

```
rule receive_initial_message:
  let
    exchange = <curve_identity_key_public_alice, ephemeral_key_public,
                kem_ciphertext_1, kem_ciphertext_2, kem_ciphertext_3,
                initial_ciphertext>

    shared_secret_1 = kem_decapsulate(
      kem_ciphertext_1, curve_identity_key
    )
    shared_secret_2 = kem_decapsulate(
      kem_ciphertext_2, curve_prekey
    )
    shared_secret_3 = kem_decapsulate(
      kem_ciphertext_3, pqkem_prekey
    )

    secret_key = kdf(
```

```
    <shared_secret_1, shared_secret_2, shared_secret_3>
  )

  initial_plaintext = sdec(initial_ciphertext, secret_key)
in
[ !PublicKeyAlice(curve_identity_key_public_alice),
  !KeysBob(curve_identity_key),
  !PrekeysBob(curve_prekey, pqkem_prekey),
  In(exchange) ]
--[ReceiveInitialMessage(
     initial_plaintext, initial_ciphertext, secret_key
   ),
   Eq(
     verify(
       exchange_signature, exchange, curve_identity_key_public_alice
     ),
     true
   )]->
[]
```

U ovom pravilu, Bob koristi svoje privatne ključeve za dekapsulaciju i obradu inicijalne poruke poslane od strane Alice. Konkretno, pravilo `receive_initial_message` uključuje:

1. Primanje poruke koja uključuje javni identifikacijski ključ Alice, jednokratni (engl.*ephemeral*) ključ, KEM šifrate i inicijalni šifrat:

   ```
   In(exchange)
   exchange = <curve_identity_key_public_alice, ephemeral_key_public,
               kem_ciphertext_1, kem_ciphertext_2, kem_ciphertext_3,
               initial_ciphertext>
   ```

2. Dekapsulaciju privatnih ključeva Boba koristeći KEM šifrate iz inicijalne poruke:

```
shared_secret_1 = kem_decapsulate(
  kem_ciphertext_1, curve_identity_key
)
shared_secret_2 = kem_decapsulate(
  kem_ciphertext_2, curve_prekey
)
shared_secret_3 = kem_decapsulate(
  kem_ciphertext_3, pqkem_prekey
)
```

3. Derivaciju tajnog ključa korištenjem KDF i zajedničkih tajni:

```
secret_key = kdf(
  <shared_secret_1, shared_secret_2, shared_secret_3>
)
```

4. Dekripciju inicijalnog šifrata korištenjem deriviranog tajnog ključa:

```
initial_plaintext = sdec(initial_ciphertext, secret_key)
```

5.2.4. Akcijske činjenice

U modelu se koriste različite akcijske činjenice za praćenje ključnih događaja i stanja tijekom izvršavanja protokola. Neke akcijske činjenice ne predstavljaju događaje, već se zajedno s restrikcijama koriste za osiguravanje određenih svojstava, poput osiguravanja da se klijent inicijalizira samo jednom. Sljedeća tablica prikazuje popis akcijskih činjenica korištenih u modelu:

Akcijska činjenica	**Opis događaja**
`InitBob()`	Inicijalizacija klijenta Bob i njegovog identifikacijskog ključa.
`GeneratePrekeysBob()`	Generiranje predključeva.
`InitAlice()`	Inicijalizacija klijenta Alice i njezinog identifikacijskog ključa.
`PublishKeys()`	Objavljivanje predključeva na poslužitelju.
`SendInitialMessage(pt, ct, sk)`	Slanje inicijalne poruke. Akcijska činjenica sadrži izvorni tekst poslane inicijalne poruke, šifrat poruke i derivirani tajni ključ.
`ReceiveInitialMessage(pt, ct, sk)`	Primanje inicijalne poruke. Akcijska činjenica sadrži izvorni tekst primljene poruke, šifrat poruke i derivirani tajni ključ.
`RevealIdentityKeyBob()`	Otkrivanje Bobovog identifikacijskog ključa napadaču.
`RevealPrekeysBob()`	Otkrivanje Bobovih predključeva napadaču.
`Eq(x, y)`	Osigurava jednakost dvije vrijednosti x i y (restrikcija 5.1).
`OnlyOnceInitAlice()`	Osigurava da se Alice inicijalizira samo jednom (restrikcija 5.2).
`OnlyOnceInitBob()`	Osigurava da se Bob inicijalizira samo jednom (restrikcija 5.3).

Tablica 5.1. Popis akcijskih činjenica korištenih u modelu

Ove akcijske činjenice ključne su za provjeru ispravnosti protokola i njegovih sigurnosnih svojstava. Inicijalizacijske činjenice (`InitBob()`, `InitAlice()`) i njihove ograničavajuće inačice (`OnlyOnceInitBob()`, `OnlyOnceInitAlice()`) osiguravaju da se identifikacijski ključevi generiraju i koriste ispravno. Činjenice vezane uz otkrivanje ključeva (`RevealIdentityKeyBob()`, `RevealPrekeysBob()`) omogućuju simulaciju različitih napadačkih scenarija. Dodatno, akcijske činjenice za slanje i primanje poruka (`SendInitialMessage(pt, ct, sk)` i `ReceiveInitialMessage(pt, ct, sk)`) omogućuju provjeru ispravnosti komunikacije između Alice i Boba, dok činjenica `Eq(x, y)`

osigurava ispravnost potpisa u različitim dijelovima protokola.

5.2.5. Leme (engl.*Lemmas*)

Leme u modelu definiraju ključna sigurnosna svojstva protokola i omogućuju njihovu verifikaciju. U modelu su definirana četiri sigurnosna svojstva koja će biti detaljnije opisana u nastavku. Uz te leme definiramo dodatnu lemu `message_exchange` koja služi kao *"sanity check"* kako bi ustanovili ispravno ponašanje protokola.

```
lemma message_exchange:
  exists-trace
  "
  Ex initial_plaintext initial_ciphertext secret_key
     #i1 #i2 #i3 #i4 #i5 #i6.
  InitBob() @ #i1 &
  InitAlice() @ #i2 &
  GeneratePrekeysBob() @ #i3 &
  PublishKeys() @ #i4 &
  SendInitialMessage(
    initial_plaintext, initial_ciphertext, secret_key
  ) @ #i5 &
  ReceiveInitialMessage(
    initial_plaintext, initial_ciphertext, secret_key
  ) @ #i6 &
  #i1 < #i2 & #i2 < #i3 & #i3 < #i4 & #i4 < #i5 & #i5 < #i6
  "
```

Ova lema provjerava postoji li barem jedan trag (engl.*trace*) u kojem se svi potrebni događaji za razmjenu inicijalne poruke izvrše ispravnim redoslijedom. Pomoću leme koja sadrži *exists-trace* dokazujemo da se naš protokol može izvršiti. Ako pokušamo dokazati ovo s modelom koji sadrži pogrešku, lema će biti opovrgnuta. Općenito se preporučuje prvo dokazati *exists-trace* leme prije nego se ispituju druga svojstva [6].

Međusobna autentikacija (engl.*Mutual authentication*)

Međusobna autentikacija osigurava da je poruka koju je primio Bob doista poslala Alice, te da samo Bob može uspješno primiti i dekriptirati inicijalnu poruku. Ovo svojstvo sprječava napadača da se lažno predstavlja kao jedan od klijenta. Lema `authentication` provjerava to svojstvo.

```
lemma authentication:
  "
  All plaintext ciphertext secret_key #i.
  ReceiveInitialMessage(plaintext, ciphertext, secret_key) @ #i &
  (All #j #k.
    GeneratePrekeysBob() @ #j & GeneratePrekeysBob() @ #k ==> #j = #k
  ) &
  not(Ex #j. RevealIdentityKeyBob() @ #j) &
  not(Ex #j. RevealPrekeysBob() @ #j)
  ==>
  (Ex #j. SendInitialMessage(plaintext, ciphertext, secret_key) @ #j)
  "
```

Lema se sastoji od:

1. Uvjeta da je Bob primio inicijalnu poruku u nekom trenutku.
2. Ograničenja koje osigurava da su predključevi za Boba generirani samo jednom. Ovime sprječavamo slanje inicijalne poruke generirane s jednim skupom predključeva i primanja poruke generirane s drugim skupom predključeva.
3. Ograničenja da napadač nije saznao privatne ključeve Boba.
4. Zaključka da postoji trenutak kada je Alice poslala inicijalnu poruku koju je Bob primio.

Ovom lemom provjeravamo da, ako je Bob uspješno primio poruku, da je ta ista poruka morala biti poslana od strane Alice, čime potvrđujemo identitet pošiljatelja i osiguravamo autentičnost.

Integritet (engl.*Integrity*)

Integritet osigurava da se poruka ne može modificirati tijekom slanja između klijenta. Ovo svojstvo sprječava napadača da modificira poruke dok su u tranzitu. Lema `integrity` provjerava to svojstvo.

```
lemma integrity:
  "
  All plaintext_1 ciphertext_1 secret_key_1
      plaintext_2 ciphertext_2 secret_key_2
      #i #j.
  SendInitialMessage(
    plaintext_1, ciphertext_1, secret_key_1
  ) @ #i &
  ReceiveInitialMessage(
    plaintext_2, ciphertext_2, secret_key_2
  ) @ #j &
  (All #k #l.
    GeneratePrekeysBob() @ #k & GeneratePrekeysBob() @ #l ==> #k = #l
  ) &
  not(Ex #k. RevealIdentityKeyBob() @ #k) &
  not(Ex #k. RevealPrekeysBob() @ #k)
  ==>
  plaintext_1 = plaintext_2 &
  ciphertext_1 = ciphertext_2 &
  secret_key_1 = secret_key_2
  "
```

Lema se sastoji od:

1. Uvjeta da je Alice poslala inicijalnu poruku, a Bob primio poruku u nekom trenutku. Korištenjem različitih varijabli za dvije akcijske činjenice omogućujemo da klijenti potencijalno razmjene dvije različite inicijalne poruke.

2. Ograničenja koje osigurava da su predključevi za Boba generirani samo jednom. Ovime sprječavamo korištenje različitih skupova predključeva za slanje i primanje poruka.

3. Ograničenja da napadač nije saznao privatne ključeve Boba.

4. Zaključka da sadržaj poslane poruke (plaintext, ciphertext i secret_key) mora biti identičan sadržaju primljene poruke.

Ovom lemom provjeravamo da sadržaj inicijalne poruke nije mogao biti modificiran ako su klijenti uspješno razmijenili inicijalnu poruku.

Unaprijedna povjerljivost (engl.*Forward Secrecy*)

Unaprijedna povjerljivost osigurava da sesijski ključevi neće biti kompromitirani čak i ako su dugotrajni ključevi, koji su korišteni za generiranje sesijskih ključeva, kompromitirani. Lema `forward_secrecy` provjerava to svojstvo.

```
lemma forward_secrecy:
  "
  All initial_plaintext initial_ciphertext secret_key #i #j.
  SendInitialMessage(
    initial_plaintext, initial_ciphertext, secret_key
  ) @ #i &
  ReceiveInitialMessage(
    initial_plaintext, initial_ciphertext, secret_key
  ) @ #j &
  (Ex #k. RevealIdentityKeyBob() @ #k) &
  not(Ex #k. RevealPrekeysBob() @ #k & #k < #j) &
  #i < #j
  ==>
  not(Ex #k. K(secret_key) @ #k & #k < #j)
  "
```

Lema se sastoji od:

1. Uvjeta da je Alice poslala inicijalnu poruku, a Bob primio istu poruku u nekom trenutku.

2. Uvjeta da je napadač saznao identifikacijski ključ nakon što je poruka poslana i primljena.

3. Ograničenja da napadač nije saznao predključeve prije nego što je poruka primljena.

4. Zaključka da napadač ne može saznati tajni ključ korišten za tu komunikaciju, čak i ako sazna identifikacijski ključ Boba nakon što je poruka primljena.

Ova lema osigurava da čak i u slučaju kompromitacije dugotrajnih identifikacijskih ključeva, povjerljivost prošlih sesija nije ugrožena. Kako bi napadač uspio kompromitirati sesijske ključeve mora istovremeno imati pristup dugotrajnim identifikacijskim ključevima i jednokratnim predključevima.

Sigurnost nakon kompromitacije (engl.*Post-Compromise Security*)

Sigurnost nakon kompromitacije osigurava da, čak i ako napadač kompromitira tajni ključ korišten u jednoj sesiji, to ne ugrožava sigurnost drugih sesija.
Lema `post_compromise_security` provjerava to svojstvo.

```
lemma post_compromise_security:
  "
  All initial_plaintext_1 initial_ciphertext_1 secret_key_1
      initial_plaintext_2 initial_ciphertext_2 secret_key_2
      #i1 #i2 #i3 #i4.
  SendInitialMessage(
    initial_plaintext_1, initial_ciphertext_1, secret_key_1
  ) @ #i1 &
  ReceiveInitialMessage(
    initial_plaintext_1, initial_ciphertext_1, secret_key_1
  ) @ #i2 &
  SendInitialMessage(
    initial_plaintext_2, initial_ciphertext_2, secret_key_2
```

```
) @ #i3 &
ReceiveInitialMessage(
  initial_plaintext_2, initial_ciphertext_2, secret_key_2
) @ #i4 &
(All #i #j.
  RevealPrekeysBob() @ #i & RevealPrekeysBob() @ #j ==> #i = #j
) &
(Ex #i. K(secret_key_2) @ #i) &
#i1 < #i2 & #i2 < #i3 & #i3 < #i4
==>
not(Ex #i. K(secret_key_1) @ #i)
"
```

Lema se sastoji od:

1. Uvjeta da su dvije poruke poslane i primljene u različitim sesijama.
2. Ograničenja da je napadač saznao samo jedan skup predključeva koji je korišten u jednoj od dvije uspostavljenih sesija.
3. Uvjeta da je napadač saznao tajni ključ korišten u drugoj sesiji.
4. Zaključka da napadač ne može kompromitirati tajni ključ korišten u prvoj sesiji, čak i ako je kompromitirao tajni ključ iz druge sesije.

Ova lema osigurava da kompromitacija tajnog ključa iz buduće sesije ne ugrožava sigurnost prošlih sesija.

6. Rezultati sigurnosne analize

Sve leme navedene u 5.2.5. uspješno su dokazane, što znači da su sva ispitana sigurnosna svojstva potvrđena. Analiza je pokazala da je modificirani protokol postigao željene ciljeve u održavanju svojstava međusobne autentičnosti i post-kvantne unaprijedne povjerljivosti.

Uklanjanjem potpisa inicijalne poruke lema za autentičnost je opovrgnuta, što potvrđuje važnost potpisa za provjeru autentičnosti pošiljatelja. Slično tome, uklanjanjem predključeva iz izračuna zajedničkih tajni, lema za unaprijednu povjerljivost je također opovrgnuta. Ovo naglašava značaj predključeva u postizanju post-kvantne unaprijedne povjerljivosti. Time se dodatno potvrđuje važnost ovih modifikacija za osiguranje cjelokupnih sigurnosnih svojstava protokola.

Modificirani model protokola uspješno je dokazao sva svojstva navedena u 2.3. Osim toga, dodatno modificirani model, koji koristi post-kvantni algoritam za digitalne potpise, također ostvaruje svojstvo post-kvantne međusobne autentikacije, što nije ostvareno u originalnoj specifikaciji. Iako je ovo svojstvo vrlo važno za sigurnost protokola, ono nije ostvareno u prvoj reviziji jer je primarni cilj protokola bio spriječiti napade tipa "Sakupljaj sada, dekriptiraj kasnije" (engl.*"Harvest now, decrypt later"*). Ovi napadi postaju izuzetno opasni s razvojem kvantnih računala koja mogu retroaktivno dekriptirati zabilježene komunikacije [3].

Iako su glavna sigurnosna svojstva bila uspješno dokazana, valja napomenuti da svojstvo kriptografskog poricanja navedeno u 2.3. nije bilo uključeno u trenutni model. Kriptografsko poricanje omogućava sudionicima da poriču svoje sudjelovanje u komunikaciji, što može biti korisno u određenim scenarijima, ali nije ključno sigurnosno svojstvo protokola.

Vrijeme izvršavanja dokazivanja također je značajan aspekt formalne verifikacije. Analiza je pokazala da je vrijeme izvršavanja bilo relativno kratko kada su korišteni samo KEM izračuni. Međutim, dodavanjem Diffie-Hellman izračuna, vrijeme izvršavanja dokazivanja postalo je znatno duže i presporo za učinkovitu analizu.

Ovi rezultati naglašavaju važnost formalne verifikacije u analizi i potvrđivanju sigurnosnih svojstava kriptografskih protokola. U ovom slučaju, analiza je pokazala da modificirani model PQXDH protokola ispunjava ključna sigurnosna svojstva kao što su autentičnost, integritet i unaprijedna povjerljivost. Uklanjanjem kritičnih komponenti kao što su potpisi i predključevi, analiza je dodatno pokazala kako su ove komponente ključne za održavanje sigurnosnih svojstava protokola.

7. Zaključak

U ovom diplomskom radu analizirana su sigurnosna svojstva PQXDH protokola koristeći alat za formalnu verifikaciju Tamarin. Fokus rada bio je na formalnom modeliranju i verifikaciji PQXDH protokola s ciljem da se analizira njegova otpornost na napade kvantnim računalima te da se potvrde ključna sigurnosna svojstva kao što su povjerljivost, integritet, autentičnost poruka i unaprijedna povjerljivost.

U sklopu rada uspješno je izvršena formalna verifikacija i modeliranje PQXDH protokola, pri čemu su sva ključna sigurnosna svojstva potvrđena. Analiza je jasno pokazala važnost određenih dijelova protokola za ostvarivanje željenih sigurnosnih svojstava. Tijekom analize identificirane su i određene poteškoće, od kojih je glavna poteškoća bila neuspješnost modeliranja protokola prema izvornoj specifikaciji, što je zahtijevalo modifikacije modela. Konkretno, bilo je potrebno zamijeniti Diffie-Hellman izračune KEM izračunima kako bi se ubrzala verifikacija.

Ovaj rad ističe ključnu ulogu formalne verifikacije u procesu razvoja i analize kriptografskih protokola. Formalna verifikacija pruža način za otkrivanje i ispravljanje potencijalnih ranjivosti u ranim fazama razvoja protokola, prije nego što one postanu ozbiljan sigurnosni problem. Korištenjem alata za formalnu verifikaciju moguće je temeljito ispitati sigurnosna svojstva protokola što omogućuje bolju otpornost protokola na trenutne i buduće prijetnje.

Literatura

[1] M. Marlinspike i T. Perrin, “The x3dh key agreement protocol”, *Open Whisper Systems*, sv. 283, br. 10, 2016.

[2] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, i D. Stehlé, “Crystals-kyber algorithm specifications and supporting documentation”, *NIST PQC Round*, sv. 2, br. 4, str. 1–43, 2019.

[3] E. Kret i R. Schmidt, “The pqxdh key agreement protocol”, 2023.

[4] “Cryspen | an analysis of signal’s pqxdh”, Jun 2024. [Mrežno]. Adresa: https://cryspen.com/post/pqxdh/

[5] S. Meier, B. Schmidt, C. Cremers, i D. Basin, “The tamarin prover for the symbolic analysis of security protocols”, u *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*. Springer, 2013., str. 696–701.

[6] “Tamarin prover manual”, Jun 2024. [Mrežno]. Adresa: https://tamarin-prover.com/manual/

[7] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, i D. Stehlé, “Crystals-dilithium: A lattice-based digital signature scheme”, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, str. 238–268, 2018.

Sažetak

Formalna verifikacija post-kvantnog protokola pomoću alata Tamarin

Ivan Futivić

Ovaj diplomski rad bavi se sigurnosnom analizom PQXDH protokola, post-kvantne modifikacije X3DH protokola razvijenog za aplikaciju Signal. Cilj rada bio je analizirati sigurnosna svojstva protokola koristeći alat za formalnu verifikaciju Tamarin. Provedena je analiza sigurnosnih svojstava poput autentičnosti, integriteta i unaprijedne povjerljivosti. Rezultati su pokazali da PQXDH protokol zadovoljava sva ključna sigurnosna svojstva. Formalna verifikacija također je omogućila identifikaciju kritičnih dijelova protokola, potvrđujući njihovu neophodnost u osiguravanju navedenih svojstava i cjelokupne sigurnosti protokola. Ovaj rad također potvrđuje važnost formalne verifikacije u razvoju sigurnih kriptografskih protokola otpornih na napade kvantnim računalima.

Ključne riječi: Tamarin, X3DH, PQXDH, formalna verifikacija, post-kvantna kriptografija

Abstract

Ivan Futivić

This thesis describes a security analysis of the PQXDH protocol, a post-quantum modification of the X3DH protocol developed by Signal. The goal of the thesis is to analyze the security properties of the protocol using a formal verification tool called Tamarin. These properties include authenticity, integrity, and forward secrecy. The results show that the PQXDH protocol meets all key security properties. Formal verification also enabled the identification of some critical parts of the protocol, confirming their necessity in ensuring these properties and the overall security of the protocol. This paper also underscores the importance of formal verification in the development of secure cryptographic protocols resistant to attacks by quantum computers.

Keywords: Tamarin, X3DH, PQXDH, formal verification, post-quantum cryptography