DIPLOMAMUNKA

Papp Gyula

Debrecen
2009

Debreceni Egyetem
Informatika Kar

PKI alapú vizsgáztatórendszerek

Témavezető:
Prof. Dr. Pethő Attila
egyetemi tanár

Készítette:
Papp Gyula
programtervező matematikus

Debrecen
2009

Tartalomjegyzék

Bevezetés **III**

1. A probléma leírása **1**
1.1. Pár szó a vizsgáztatásról . . . 1
1.2. A modell . . . 1
1.3. Az elektronikus vizsgáztató rendszer modellje . . . 3
1.4. A rendszerrel szemben támasztott követelmények . . . 7

2. A vizsgáztatórendszerek főbb sématípusai **8**
2.1. Jelölésrendszer . . . 9
2.2. A használható módszerek . . . 9
2.2.1. Titokmegosztó rendszer . . . 9
2.2.2. Nyilvánosan ellenőrizhető titokmegosztás . . . 9
2.2.3. RSA kriptorendszer . . . 10
2.2.4. Kvadratikus maradékprobléma . . . 11
2.2.5. Az ElGamal kriptorendszer . . . 12
2.2.6. Vakaláírás . . . 12
2.2.7. Bitműveletek . . . 13
2.2.8. Homomorf kódolás . . . 15
2.2.9. A robosztus küszöb ElGamal kriptorendszer . . . 15
2.2.10. Mixhálók . . . 17

3. A vizsgáztatórendszer elméleti modellje **18**
3.1. A rendszer emberi komponensei . . . 18
3.1.1. A vizsgázó . . . 18
3.1.2. A javító tanár . . . 20
3.1.3. A vizsgabiztos . . . 20
3.2. A rendszer gépi komponensei . . . 21

3.3. A vázlatos modell 22
3.4. A probléma formalizálása 23
3.4.1. A vizsgázó 23
3.4.2. A javító tanár 24
3.4.3. A vizsgabiztos 26
3.5. A vizsgáztató rendszer 28
3.5.1. A vizsgáztatáshoz használt papír 30

4. A vizsgáztatórendszer egy lehetséges elméleti megközelítése 31
4.1. A vizsgáztatórendszer modelljének megközelítése 31
4.2. Az általunk támasztott követelmények és problémáik 32
4.3. A vizsgázó 32
4.3.1. A vizsgázó anonimizálása és rendszerben való szerepe 32
4.3.2. A vizsgázó megközelítése matematikai szempontból 35
4.3.3. A panasz benyújtása 38
4.4. A javító tanár 38
4.4.1. A javító tanár anonimizálása és rendszerben való szerepe 38
4.4.2. A javító tanár megközelítése matematikai szempontból 41
4.4.3. A javító tanár személyének felfedése 42
4.5. A vizsgabiztos és a dolgozatok szétosztása 44
4.5.1. A probléma megközelítése elméleti szempontból 44
4.5.2. A probléma megközelítése gyakorlati szempontból 46

5. A vizsgáztatórendszer egy lehetséges gyakorlati megközelítése 50
5.1. Pár szó a gyakorlati alkalmazásról 50
5.2. Egy lehetséges megvalósítás 51
5.3. A program működése 53
5.4. Eltérések az elméleti modelltől 54

Függelék 55
1. Egy ötlet a vizsgázó azonosítójának előállítására 55
2. Egy ötlet a vizsgáztató azonosítójának előállítására 56
3. A digitális aláírásról röviden 56

Irodalomjegyzék 59

Forráskód 62

Bevezetés

„Holnap matematikából tudásfelmérőt írunk!” – mondta egykori tanítóm az általános iskolában. Hazamentem, és egész este lázasan kezdem átismételni a számonkérendő anyagot. Talán egész este nem is tudtam aludni, mert egyfolytában csak a másnapi matematikadolgozatra gondoltam. Másnap aztán eljött az idő. Tanítónk üdvözölt minket, osztálynévsort olvasott és beírta a hiányzókat. Az adminisztáció végén kiosztotta a feladatlapokat, mi megírtuk a dolgozatot, és az óra végén beszedte. „Mikor kapjuk ki?” – tettük fel a kérdést. Tanítónk válasza csak ennyi volt: „Majd jövő héten.”

Pár év múlva elérkezett az idő, hogy felvételizzek az egyetemre. Jelentkezési lapot adtam be, és vártam az egyetem visszajelzését, hogy mikor és hol felvételizhetek. A levél kézhezvétele után lázasan vártam a felvételire. Mikor megérkeztem a felvételi napján, felmentem egy terembe, ahol belépés után igazoltam magam, majd megkaptam a feladatlapot és a hozzá tartozó üres lapokat. Az idő letelte után beadtam a dolgozatot és vártam...

Ezek a nem is olyan távoli múltban játszódó történetek mind egyazon eseménycsoportba tartoznak: a *vizsgázáshoz*. A vizsgázás egy olyan folyamat, mely során a vizsgáztató meggyőződik arról, hogy a vizsgázó személy eleget tesz bizonyos elvárt kritériumoknak. A egy modern vizsgának a következő két fontos tulajdonsággal kell rendelkeznie: az első az, hogy *objektívnek* kell lennie, vagyis a tényleges tudást kell mérnie; a második az, hogy nem szabad *szubjektívnek* lennie, vagyis a vizsgázó szemszögéből kiindulva felépíteni a vizsgát.

A tudást felmérni igen sokféle szempontból lehet. Például az egyik út az, hogy mit tud vagy épp ellenkezőleg, mit nem tud a vizsgázó. Például, a magyar felvételi rendszer – amíg hagyományos értelemben létezett – 2005-ig kellően teljesítette a fenti két követelményt. Voltak könnyebben és nehezebben megoldható feladatok, melyekre adott megoldások segítségével lehetett mérlegelni a vizsga erősségét, és a felvételizők egyéni képességeit.

Amennyiben megpróbálnánk modellezni a vizsgázás folyamatát, két lehetséges út kínálkozik: az első út az, hogy mi magunk, az alapoktól kezdve formalizáljuk a vizsgázás és vizsgáztatás minden egyes lépését, és ezt egy működőképes rendszerré alakítjuk. Igen, ez jó megoldásnak tűnhet, viszont fennállhat a veszélye annak, hogy újrafeltalálnánk olyan fogalmakat és rendszereket amelyek már léteznek, s ezáltal a befektetett erőfeszítéseink nem térülnek meg; a

második út az, hogy rokon problémákat keresünk a vizsgáztatással és az ott elért eredményeket alkalmazzuk, természetesen a mi problémánkra átültetve.

Ha eléggé kitartóan keresünk, akkor nem is kell nagyon mélyre ásni magunkat az információk tengerében, mert ez a rokon jellegű probléma a mindennapi köztudatunkban él. Ez pedig nem más, mint a *választás*. Ennek felépítése nagyon hasonlít a vizsgáztatásra, kivéve azt, hogy nem tudjuk megmondani ki melyik jelöltre szavazott. Ha ezt a kis különbséget figyelembe vesszük, és az ott elért eredményeket a megfelelően átültetjük a mi vizsgáztatórendszerünkre, akkor nemcsak időt takarítunk meg, hanem már kiforrott rendszereket alkalmazva megtudhatjuk a jelenlegi szavazórendszerek erősségeit és gyengeségeit, s ennek tükrében tervezhetjük meg saját vizsgáztató rendszerünket.

A vizsgáztatórendszereket kizárólag az informatika szemszögéből fogjuk vizsgálni, különös figyelmet fordítva a rendszerben lévő szereplők és az őket összekötő kommunikációs csatorna biztonságára. Ez a dolgozat a következő kérdésre fogja keresni a választ: *Hogyan épül fel egy biztonságos vizsgáztatási protokoll, a jelenlegi szavazórendszerek módszereit figyelembe véve?*

A biztonság alatt azt értjük, hogy a kulcsszereplők (vizsgázó és javító tanár) a folyamat során nem használhatják a természetes azonosítójukat, anonimak, de a vizsga végén a tanulóhoz hozzá lehet rendelni a jegyét és bizonyos körülmények között a tanár is azonosítható. Ezen kívül a dokumentumokat digitálisan aláírva továbbítják, ami lehetetlenné teszi azok későbbi módosítását.

Az első fejezet röviden összefoglalja azt, hogy milyen követelményeket szeretnénk támasztani egy vizsgáztatórendszerrel szemben. Felsorolja a vizsgázásban – mint folyamatban – résztvevő személyeket, és az őket összekötő lehetséges kommunikációs csatornákat, különös figyelmet fordítva az esetleges visszaélések megakadályozására.

A második fejezet röviden bemutatja a vizsgáztatórendszereknél használt módszereket az információk hitelességének megőrzéséhez. Ezen módszerekből néhány alkalmazásra is kerül a következő fejezetekben.

A harmadik fejezet a vizsgáztatórendszer bővebb és pontosabb kifejtését tartalmazza. Itt kerülnek pontosításra az első fejezetben megfogalmazott követelmények. Először vázlatosan leírjuk mit várunk el a vizsgázás folyamatában résztvevő személyektől, majd pontosítjuk azt.

A negyedik fejezet a rendszer egy lehetséges matematikai megközelítését tartalmazza. Itt már a második fejezetben használt módszerek közül néhány kerül felhasználásra. Külön szekció foglalkozik a vizsgázó, a javító tanár és a vizsgabiztos személyekkel.

Az ötödik fejezetben a függelékben található program és az elméleti modell közötti különbségeket tárgyalja.

Végezetül, a függelékben találhatunk meg egy lehetséges gyakorlati megvalósítást, a hozzátartozó adatállományokat és egy lehetséges futási eredményt is.

1. fejezet

A probléma leírása

1.1. Pár szó a vizsgáztatásról

A vizsgázás sokban hasonlít egy szavazórendszerre, kivéve azt, hogy a vizsga végén tudjuk, hogy ki melyik dolgozatot adta be.

Egy teljes vizsgáztatórendszer alapvetően két fő komponensből áll. Az első komponens maga az *ember*, mint a vizsgában résztvevő személy. Ez a komponens három halmazra bontható. Ezek a halmazok a vizsgázó(k), a vizsgabiztos(ok) és a javító tanár(ok) halmaza. Ezen halmazcsoportok között zajlik a vizsga, mint folyamat.

A második komponens a *hardver*, mely biztosítja a három csoport közötti kommunikációt, és ezt *kommunikációs csatornának* nevezzük. Megköveteljük, hogy ez a csatorna ne csak a kapcsolatot biztosítsa a három csoport között, hanem a közöttük folyó információcsere *titkosságát* is.

1.2. A modell

A hagyományos és az elektronikus vizsgáztatás nagyon sok ponton hasonlít egymásra, kivéve azt, hogy az első esetben hagyományos, míg a második esetben „elektronikus” papírt használunk. Elektronikus papír lehet akár egy szöveges állomány, de lehet akár egy álló- vagy mozgókép is. Maga a vizsgáztató rendszer három fő halmazból fog állni. Ezek között fog zajlani az információk cseréje:

- vizsgázó,
- javító tanár,
- vizsgabiztos.

Mind a vizsgabiztos, mind a vizsgázó és a javító tanár lehet akár egy- vagy több személy is. Sőt, megengedett, hogy a javító tanár lehessen akár egy program[1] is. Itt szeretném megjegyezni, hogy ezen halmazok egyidejűleg *diszjunktak*, vagyis egy vizsgán nem lehet vizsgázó az, aki vizsgabiztos vagy javító tanár. Az viszont megengedett, hogy egy javító tanár vizsgabiztos vagy vizsgabiztos javító tanár legyen.

Vizsgázó

Bármely természetes személy, aki jogosult, az vizsgát tehet. Nyilván, aki nem kíván vizsgázni, az ki lesz zárva a folyamat többi részéből is, viselve az ezzel járó esetleges szankciókat is. A vizsgázó addig nem tehet vizsgát, amíg nem rendelkezik a vizsgabiztos által kibocsátott azonosító jellel, mely feljogosítja erre. Továbbá megköveteljük, hogy a vizsga ideje alatt a személyazonossága ismeretlen legyen.

Javító tanár

A vizsgázó dolgozatát javítja ki, és elküldi a vizsgabiztosnak. Megköveteljük, hogy csak az arra jogosult tanár javíthassa ki a vizsgázó dolgozatát. Ezt az engedélyt a vizsgabiztostól kapja. Természetes igény, hogy a javító tanár se vizsgázhasson egy időben a vizsgázókkal, és személye ismeretlen legyen a vizsgázás folyamata alatt.

Vizsgabiztos

Ő felügyeli és irányítja a vizsgázás minden lépését. Ő hagyja jóvá a javító tanár személyét, ő ellenőrzi a vizsgázó jogosultságát és hirdeti ki a vizsga eredményeit is. Természetesen, a vizsga időtartama alatt ő nem tehet vizsgát. Feltehetjük, hogy őt egy felsőbb szervezet (állami hivatal, egyetemek vagy főiskolák vezetése) jelöli ki, továbbá megengedjük azt is, hogy személye ismert legyen a vizsga ideje alatt.

Vizsga

A vizsga lehet igaz/hamis választás, egy vagy több lehetőség választása egy csoportból vagy egyszerűen kérdéseket tesznek fel, és arra a vizsgázó esszé jelleggel válaszol. Ez utóbbi a legtipikusabb egy vizsga esetén.

[1] Ez utóbbira tipikus példa a személygépkocsi vizsga.

Hitelesség

Nyilván azt szeretnénk, hogy a vizsgabiztos ne legyen becstelen és csak a dolgozatok javítása után tudja megnézni, hogy ki melyik dolgozatot írta. Sajnos fel kell tételezni, hogy maga a vizsgabiztos is egy becstelen személy. Ezt meg lehet oldani időzáras megoldással vagy úgy, hogy több vizsgabiztost alkalmazunk, akik csak egyszerre tekinthetik meg a dolgozatokat.

Kommunikáció

Szinte a kommunikáció a legfontosabb az ilyen rendszerek esetén, ugyanis ezen zajlik a résztvevők közötti „beszélgetés”. Ezt általában valamilyen kommunikációs csatorna segítségével oldják meg. Alapvetően három kommunikációs csatorna típust különböztetünk meg:

- A *lehallgathatatlan csatorna* egy titkosított csatorna a vizsgában résztvevők között. Jó tulajdonsága, hogy senki nem tudja bebizonyítani azt, hogy kinek mit küldött. Az üzenetek így nem változtathatóak meg.
- A *névtelen csatorna* garantálja a feladó személyének anonimitását, így nem követhető vissza a kapott üzenet. Hívják még visszakövethetetlen névtelen csatornának is.
- A *lehallgathatatlan névtelen csatorna* az előző két csatorna tulajdonságait ötvözi. Fizikai átviteli biztonságot és a küldő fél számára anonimitást biztosít.

1.3. Az elektronikus vizsgáztató rendszer modellje

A vizsgáztatás a következő lépéseken át valósul meg:

1. A vizsgabiztos akkreditálja a javító tanárokat.
2. A vizsgázó jogosultsága ellenőrzésre kerül. Amennyiben
   (a) jogosult, akkor azonosítót kap (anonimitás);
   (b) nem jogosult, akkor kizárjuk a vizsgáról és ugrás **9.**-re.
3. Az azonosítót ráírja a dolgozatára, és megírja azt.
4. A megírt dolgozatot elküldi a vizsgabiztosnak.

5. A vizsgabiztosnál

   (a) ellenőrzésre kerül, hogy ez valóban vizsgadolgozat-e, ha

      i. igen, akkor tovább a **(b)**-re;

      ii. nem, akkor a dolgozat eldobásra kerül, és ugrás **9.**-re.

   (b) az aktuális vizsgára szól-e az azonosító, ha

      i. igen, akkor tovább a **(c)**-re;

      ii. nem, akkor a dolgozat eldobásra kerül, és ugrás **9.**-re.

   (c) a vizsgázó adott-e már be vizsgadolgozatot, ha

      i. igen és az aktuális azonosító még nem szerepelt, akkor a dolgozat eldobásra kerül, és ugrás **9.**-re;

      ii. igen és az aktuális azonosító szerepelt vagy még nem adott be, akkor tovább **6.**-ra.

6. A vizsgabiztos elküldi a dolgozatokat a javító tanárnak.

7. A javító tanár ráírja az azonosítóját a dolgozatra, kijavítja és visszaküldi a vizsgabiztosnak.

8. A vizsgabiztos

   (a) feloljda a dolgozatok anonimitását, eredményt hirdet és a kijavított dolgozatot visszaküldi a vizsgázónak;

   (b) amennyiben a vizsgázó

      i. nem ért egyet az elért eredménnyel, akkor vissza **5.**-re;

      ii. egyetért az elért eredménnyel, akkor menjünk **9.**-re.

9. A vizsgázás folyamata véget ért.

Most lássuk a fenti lépéseket egyenként. A vizsgáztatás egy úgynevezett „nulladik lépéssel” kezdődik. Ezt *akkreditációs* (előkészítő) lépésnek is nevezhetjük. Ebben a lépésben a vizsgabiztos felveszi javító tanár adait, jegyzékbe veszi, hogy mit tanít és besorolja őt azon kategóriákba, mely kategóriákban ő javítási funkciót tölthet be. Természetesen megkapja saját azonosítóját, mely a vizsgázó szemszögéből biztosítja számára anonimitást.

A második lépésben történik a vizsgázó jogosultságának ellenőrzése. Itt gyakorlatilag arról van szó, hogy a vizsgázó bebizonyítja a személyazonosságát, a vizsgabiztos ellenőrzi az ő jogosultságát.

A következő lépésben, amennyiben a vizsgázó jogosult vizsgát tenni, megkapja azonosítóját, mely biztosítja számára a megfelelő anonimitást a vizsga ideje alatt. Amennyiben nem jogosult, akkor a vizsgáról automatikusan kizáródik.

A negyedik lépés a vizsgázó szemszögéből a legfontosabb: ráírja azonosítóját a vizsgalapjára és megírja a dolgozatot, majd az ötödik lépésben elküldi azt a vizsgabiztosnak.

A hatodik lépés egy nagyon fontos mozzanata a folyamatnak. Itt a dolgozat már beérkezett a vizsgabiztoshoz, de mégsem lehetünk biztosak abban, hogy valóban érvényes vizsgadolgozatról van szó. Ennek igazán akkor van nagy jelentősége, ha a vizsgázó panasszal akar élni a kapott vizsgajegyét illetően, ugyanis ekkor már több idő is eltelhetett a vizsgázás és a javítás között. A következő lépésben a vizsgabiztos elküldi az érvényes dolgozatot a javító tanárhoz. Itt szeretném megemlíteni, hogy célszerű korlátozni a vizsga ideje alatt megírt dolgozatok többszörös beküldhetőségének számát.

A nyolcadik lépésben a javító tanár ráírja – a vizsgabiztostól kapott – azonosítóját és kijavítja a dolgozatot és visszaküldi a vizsgabiztoshoz.

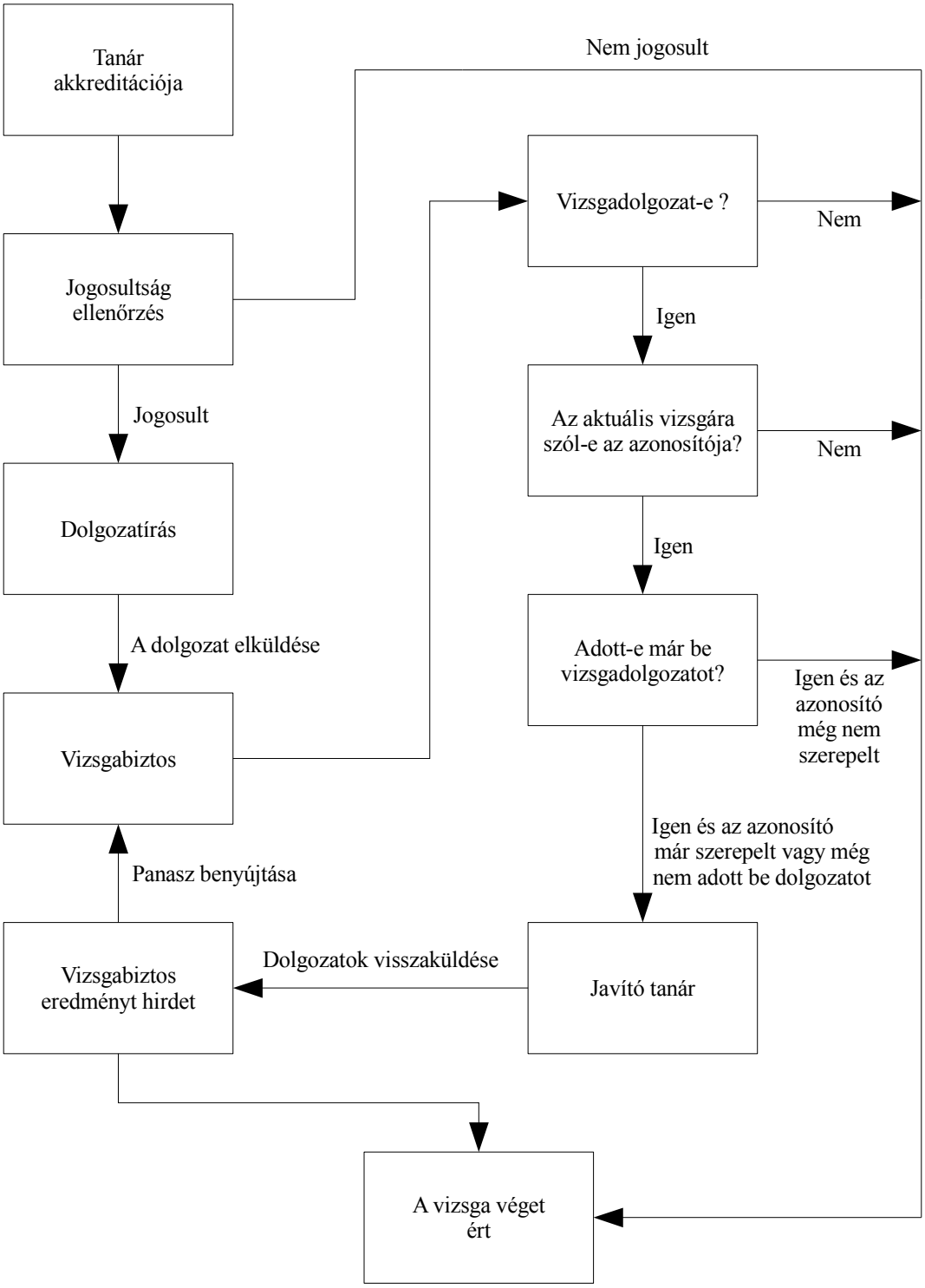
A kilencedik lépésben a vizsgabiztos megkapja a dolgozatot, feloldja a vizsgázó anonimitását és eredményt hirdet. Ha a vizsgázó elégedett a kapott érdemjegyével, akkor a vizsgázás folyamatának vége. Azonban ha nem elégedett, akkor meg kell engednünk a vizsgázónak azt, hogy panasszal élhessen. Ebben az esetben vissza kell lépnünk a folyamat egy korábbi lépéséhez.

Természetesen megengedjük, hogy az egész folyamat alatt a vizsgázó ellenőrizhesse a javító tanár azonosítójának érvényességét és azt is, hogy a javító tanár ugyanezt megtehesse.

Mivel a vizsgáztató rendszerünk működése – a fenti néhány pont alapján – ismert, így szemügyre vehetjük azt is, hogy mi az erőssége és gyengesége rendszerünknek. A vizsgáztató rendszerünk akkor fog igazán jól működni, ha a vizsgában résztvevők azonosítóját gyakorlatilag lehetetlen felfedni. Ha ez a feltétel teljesül, akkor még mindig van egy fontos tényező, ami befolyásolhatja a vizsgázás kimenetelét. Ez pedig nem más, mint az emberi tényező. Sajnos ezt nem mindig lehet „szabályok” és „lépések” sorozatával korlátozni.

Éppen ezért, az emberi tényezők minimálisra csökkentésével biztosíthatjuk rendszerünk hatékonyságát. Amennyiben az emberi tényezőt figyelmen kívül hagyjuk, akkor kizárólag az azonosító felfedhetetlenségével kell foglalkoznunk. A fenti néhány pont alapján jól látható, hogy a vizsgabizottság az, ahová a folyamat többször is visszatérhet. Éppen ezért ez a legfontosabb része a folyamatnak.

Az 1.1. ábra a vizsgázás elvi modelljét írja le. Ha jól megnézzük, akkor ez tartalmazhat egy körfolymatot is, ami nagyon veszélyes az ilyen típusú rendszerek esetén.



1.1. ábra. A vizsgáztatás elvi modellje

1.4. A rendszerrel szemben támasztott követelmények

Egy, a gyakorlatban alkalmazható vizsgáztatórendszernek a következő tulajdonságokat kell kielégítenie:

- *Jogosultságellenőrzés:* Ebben a folyamatban a vizsgabiztos leellenőrzi, hogy a vizsgázó valóban az a személy, akinek mondja magát, és ha igen, akkor jogosult-e a vizsgázásra. Természetesen, a vizsgázó egyszerre csak egy tárgyból tud vizsgát tenni.

- *Az azonosító létrehozása:* A vizsgabiztos létrehoz egy azonosítót, amit még ő maga sem ismer és átadja a vizsgázónak.

- *Azonosító ellenőrizhetősége:* Minden vizsgázásra jogosult személy kap egy azonosítót, amit ráír a dolgozatára és csak ő ismeri. Persze ellenőrizheti bárki azt, hogy ez az azonosító valóban érvényes-e (teljesen egyedi, még nem használta senki).

- *Ellenőrzés általánosan:* Bárki ellenőrizhesse, hogy a vizsgázás alatt minden rendben történt, valamint azt, hogy valóban csak olyan dolgozatok lettek értékelve, amelyeket ténylegesen a vizsga alkalmával írtak meg.

- *Korrektség:* A csalás lehetőségét teljesen ki kell zárni a rendszerből. Meg kell akadályozni, hogy többen egy dolgozatot írhassanak meg.

- *Robosztusság:* Ki kell zárni, hogy a vizsgázók egy csoportja vagy akár egy vizsgázó is meg tudja hiúsítani a vizsgát.

- *Azonosíthatatlanság:* Akkor jó a rendszerünk, ha a vizsga végéig nem tudjuk megmondani azt, hogy melyik vizsgázó melyik dolgozatot írta. Így biztosíthatjuk az egyik legfontosabb tulajdonságot, az anonimitást, s ezáltal kizárhatjuk teljesen a csalás lehetőségét rendszerünkből. Így egy esetlegesen ártó szándékkal bíró társunk nem ismerheti dolgozatunk válaszait, és nem tud „másolni” vagy szövetkezni az éppen vizsgáztató tanárral.

2. fejezet

A vizsgáztatórendszerek főbb sématípusai

Ebben a fejezetben megismerhetjük azokat az eljárásokat és megoldásokat melyeket felhasználhatunk egy vizsgáztatórendszer tervezése során. A fejezet elején felsorolom az általam használt jelöléseket, majd bemutatok néhány eljárást, melyeket használni lehetne egy vizsgáztatórendszer esetén.

Itt szeretném megjegyezni, hogy a titkosítórendszerek általános követelményeit 1883-ban *Auguste Kerckhoffs von Nieuwenhof* holland nyelvész fogalmazta meg *La criptographie militaire* című munkájában [5]. Ebben a következő két fontos szempontra hívja fel a figyelmet:

- ha egy rendszer elméletileg nem feltörhetetlen, akkor a gyakorlatban legyen az;
- a rendszer részleteinek kompromittálódása ne okozza a rendszer egészének kompromittálódását.

Az első gondolat röviden azt fejezi ki, hogy az elméleti korlátok nehogy kivitelezhetőek legyenek a gyakorlatban, illetve ha mégis, akkor igen nagy idő- és/vagy tárigénye legyen. A második gondolat pedig azt fejezi ki, hogy ha egy támadó információt szerez a rendszer egy részéről, ennek ismeretében még ne legyen képes veszélyeztetni a rendszer egészét.

Természetesen a ma használatos rendszereknél már újabb elvárások is vannak, de a fenti két gondolat továbbra is érvényben van. **Ez a Kerckhoffs-elv.**

2.1. Jelölésrendszer

N	A vizsgabiztosok száma
A_j	A j-edik vizsgabiztos
t	A megbízhatatlan vizsgabiztosok száma
Z_p	A pozitív egész számok tere *modulo p*
Z_n	Az egész számok halmaza *modulo n*, azaz $\{0, 1, \ldots, n-1\}$
Z_n^*	Z_n azon elemeinek halmaza, amelyek relatív prímek n-hez.
$x \in_R X$	Az X halmaz egy véletlen eleme (egyenletes eloszlással)
$a \oplus b$	A bitenkénti kizáró vagy művelet

2.2. A használható módszerek

2.2.1. Titokmegosztó rendszer

A titokmegosztó rendszerek működésének alapja az, hogy a titkot osszuk szét N vizsgabiztos között úgy, hogy később csak egy általunk meghatározott csoportja tudja visszaállítani azt. A többi biztosnak nincs tudomása a titokról. Erre példa Shamir $(t+1, N)$ titokmegosztó rendszere [2]-ból, amely megengedi bármely $t+1$ biztosnak, hogy hozzáférhessen a titokhoz.

Működése a következő: legyen titkunk $s \in F$, ahol F egy véges test, és legyen s_j az a titokrész, amelyet megkap minden általunk jóváhagyott A_j biztos ($j = 0, \ldots, t+1$). Ekkor válasszunk egy véletlen t-edfokú f polinomot F felett melyre igaz, hogy $f(0) = s$, továbbá az $s_j = f(j)$. Ez biztosítja számunkra a titok megfelelő elosztását $t+1$ vizsgabiztos között.

A titok helyreállításához használhatunk például Lagrange-interpolációt:

$$s = f(0) = \sum_{j \in A} f(j)\lambda_{j,A} = \sum_{j \in A} s_j \lambda_{j,A},$$

ahol

$$\lambda_{j,A} = \prod_{l \in A - \{j\}} \frac{l}{l-j}.$$

Amennyiben t vizsgabiztosnak van ismerete az f-ről, akkor sem tud tenni semmit az $f(0) = s$ egyenlet megoldása érdekében.

2.2.2. Nyilvánosan ellenőrizhető titokmegosztás

Ennek a rendszernek lényege az, hogy megengedi a titok szétosztójának ellenőrzését, hogy megbizonyosodjunk róla, valóban érvényes részek kerültek-e szétosztásra ($t+1$ vizsgabiztos

számára). Továbbá kilátásba helyezi azt, hogy elkapjuk a megbízhatatlan biztosoknak küldött részeket is.

Legyen Z_p és legyenek kiválasztott generátorelemeink G, g. Továbbá legyen A_j biztos titkos kulcsa z_j és nyilvános kulcsa $h_j = g_{zj}$. Ekkor el kell küldenünk a vizsgabiztoshoz g_s-t.

Keressünk egy véletlen t-edfokú polinomot Z_p felett, úgy hogy teljesüljön:

$$p(x) = \sum_{k=0}^{t} \alpha_k x^k,$$

ahol $\alpha_0 = s$ és $\alpha_1, \ldots, \alpha_t \in Z_p$. A polinom titoktartó és művelete a $C_k = G^{\alpha_k}$, ahol $0 \leq k \leq t$, valamint a kódolt megosztások $H_j = h_j^{p(j)}$, ahol $j = 1, \ldots, N$ nyilvánosak. Továbbá a küldő felfedi a kódolt részek egyezőségét a következő módon: legyen $X_j = \prod_{k=0}^{t} C_k^{j^k} = G^{\sum_{k=0}^{t} \alpha_k j^k} = G^{p(j)}$, és a küldő bebizonyítja a $\log_g X_j = \log_{h_j} H_j$-t.

A titok helyreállításánál az A_j biztos visszafejti S_j részét, számítva $S_j = g^{p(j)}$-t, ahol $S_j = H_j^{1/z_j}$, vagyis bebizonyítja a $\log_g h_j = -\log_{H_j} S_j$-t. Továbbá feltehetjük, hogy $t + 1$ A_j hatóság ($j \in A$) előállítja S_j helyes értékét ($j \in A$). A g^s titok helyreállítható Lagrange-interpolációval:

$$\prod_{j \in A} S_j^{\lambda_{j,A}} = \prod g^{p(j)\lambda_{j,A}} = g^{\sum_{j \in A} p(j)\lambda_{j,A}} = g^{p(0)} = g^s,$$

ahol $\lambda_{j,A} = \prod_{l \in A - \{j\}} \frac{l}{l-j}$ a Lagrange-együttható.

2.2.3. RSA kriptorendszer

Ez az egyik legelterjedtebb rendszer, melyet a gyakorlatban kulcscserére használnak. Lényege az, hogy a küldőnek van egy nyilvános és egy titkos kulcsa. A nyilvános kulcsát publikálja, és ezzel üzenetet tudunk küldeni neki. A titkos kulcsával pedig a neki szánt üzeneteket meg tudja fejteni. Ez a rendszer igen hatékony, mert a nyilvános kulcs ismeretében még nem tudjuk megfejteni a neki szánt üzeneteket. Legyen két személy Aliz és Bob, akik üzenetet akarnak váltani.

A kulcsok előállításához Aliznak generálni kell két véletlen prímszámot p-t és q-t, körülbelül azonos nagyságrendben, majd ki kell számítania az $n = pq$-t és $\phi = (p-1)(q-1)$-t. Miután ezt elvégezte, választania kell egy egész e számot, melyre $1 < e < \phi$, hogy teljesüljön a $gcd(e, \phi) = 1$ egyenlőség. Ezt követően pedig számítania kell egy d egészet is, melyre $1 < d < \phi$ és teljesíti az $ed \equiv 1 \bmod \phi$ egyenlőséget. Ha ezzel elkészült, akkor a nyilvános kulcsa (n, e), a titkos kulcsa d.

A kódoláshoz válasszunk egy egész m számot, melyre $0 \leq m < n$, ekkor Bob, aki üzenetet akar küldeni Aliznak, ki kell számítania a $c = m^e \bmod n$-t és elküldeni neki. Aliz, miután megkapta c-t, számítania kell az $m = c^d \bmod n$-t és megkapja az eredeti m üzenetet.

2.2.4. Kvadratikus maradékprobléma

Legyen $n \in \mathbb{N}$. Azokat az $y \in Z_n^*$ elemeket, amelyek felírhatóak $y = x^2 \bmod n$ alakban valamely $x \in Z_n$-nel, *kvadratikus maradék modulo n*-nek nevezzük. Ezt a halmazt Q_n-nel jelöljük. Ha $n = p$ egy prímszám, akkor a Legendre-szimbólumot így definiáljuk:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{ha p | a ;} \\ 1, & \text{ha } a \in Q_p; \\ \text{-1}, & \text{ha } a \notin Q_p. \end{cases}$$

Ha n összetett és felírható $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ alakban, akkor a Jakobi szimbólumot így definiáljuk:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \dots \left(\frac{a}{p_k}\right)^{e_k}.$$

Már léteznek hatékony algoritmusok $(\frac{a}{n})$ kiszámítására. Amennyiben ez kvadratikus maradék, akkor $(\frac{a}{n}) = 1$. Viszont vigyáznunk kell, mert léteznek úgynevezett *álnégyzetesek* is, melyek kielégítik a fenti egyenletet. Az álnégyzetesek halmazát $\widetilde{Q_n}$-nel jelöljük.

Ha $n = pq$ két különböző prímszám szorzata, akkor $|Q_n| = |\widetilde{Q_n}| = \frac{(p-1)(q-1)}{4}$.

A kvadratikus maradékprobléma a következő: legyenek $a, n \in \mathbb{N}$. Kérdés, hogy a kvadratikus maradék-e modulo n.

Ha $n = p$, ekkor könnyen eldönthető, hogy $a \in Z_p$ kvadratikus maradék modulo p vagy sem, továbbá ez következik-e a Legendre-szimbólum definíciójából és $(\frac{a}{p})$ könnyen számítható-e.

Ha $n = p_1^{e_1} \dots p_k^{e_k}$ összetett, akkor a egy kvadratikus maradék-e, ha ez kvadratikus maradék modulo p_i minden $i = 1, \dots, k$ esetén. Ha ismert n faktorizációja, akkor ez ellenőrzéssel megoldható, vajon $(\frac{a}{p_i}) = 1$, minden $i = 1, \dots, k$-ra. Amennyiben nem ismert, akkor ez nem igazán hatékony eljárás az ellenőrzésre. Ha $n = pq$, akkor a jó megoldást meg tudjuk becsülni $\frac{1}{2}$ valószínűséggel. Ez elhiteti velünk, hogy a kvadratikus maradékproblémát lehet faktorizálni, de nem ad hozzá kellő megerősítést.

2.2.5. Az ElGamal kriptorendszer

Az ElGamal nyilvános kriptorendszer alapja bármely csoportcsalád, amely a diszkrét logaritmust használja. Általában Z_p egy rendezett q csoportjának G_q alcsoportját használja, ahol p, q nagy prímszámok, melyek kielégítik a $q|p-1$-t.

Kulcsgenerálás. Aliz ekészíti a nyilvános kulcsát és a hozzátartozó titkos kulcsát. Aliznak a következőt kell tennie:

1. Generálnia kell egy nagy p prímet és a Z_p^* csoport egy g generátorát;
2. Választ egy véletlen α számot, melyre $1 \leq \alpha \leq p-2$, és kiszámítja a $h = g^\alpha$-t;
3. A nyilvános kulcs (p, g, h) és a privát kulcs α.

Kódolás. Bob megkapja Aliz nyilvános kulcsát. Ha Bob kódolt üzenetet akar küldeni Aliznak, ahol $0 \leq m < p$, akkor a következőket kell tennie.

1. Választ egy véletlen k számot, melyre $0 \leq k \leq p-2$;
2. Kiszámítja az $x = g^k$, $y = mh^k$ értékeket;
3. Elküldi a kódolt $c = (x, y)$ üzenetet Aliznak.

Dekódolás. Ahhoz, hogy az m üzenet visszanyerhető legyen $c = (x, y)$-ból, Aliznak a következőket kell tennie:

1. Az α privát kulcs használatával kiszámítja az $r = x^{p-1-\alpha}$-t (megjegyezzük, hogy $r = x^{p-1-\alpha} = x^{-\alpha} = (g^k)^{-\alpha} = g^{-k\alpha}$);
2. Visszanyerjük m-et az $m = yr \bmod p$ számítással.

2.2.6. Vakaláírás

Szükségünk van olyan jelzésekre, melyek biztosítják számunkra a megfelelő hitelességet (csak az aláíró tudja aláírni az üzenetet) és nyilvánosan ellenőrizhető (vagyis bárki által ellenőrizhető, hogy a megkapott üzenet aláírása érvényes-e).

Ha az aláíró rendelkezik (n, e) RSA nyilvános és d privát kulccsal, akkor alá tudja írni az $m \in Z_n$ üzenetet úgy, hogy kiszámítja a $s = m^d \bmod n$-t. Ha az m üzenet aláírása s, akkor ellenőrizhetjük az érvényességét, ha kiszámítjuk vajon $m =^? s^e \bmod n$.

A vak aláírásnak az a lényege, hogy az aláíró nem fogja tudni azt, hogy mit is ír alá valójában[1].

Ha az aláíró (Aliz) rendelkezik az (n, e) RSA nyilvános és d titkos kulccsal, akkor a fogadónak (Bob), hogy megkapja az m üzenet vak aláírását a következőket kell tennie:

1. Bob elvakítja az m üzenetét $m' = mr^e \bmod n$, ahol $r \in_R$ egy véletlenszám és ezt elküldi Aliznak;

2. Aliz aláírja az m' üzenetet és elküldi $s' = m'^d \bmod n$-t a fogadónak;

3. Bob kinyeri Aliz aláírását (az m üzenet s aláírását):

$$s = \frac{s'}{r} = \frac{m'^d}{r} = \frac{m^d r^{ed}}{r} = \frac{m^d \cdot r}{r} = m^d \bmod n.$$

Formálisan, a vakaláírás rendszere az M üzenettérrel együtt egy szám-ötös: $(\eta, \chi, \sigma, \delta, \Gamma)$, ahol

- η egy polinom idejű probabilista algoritmus, amely elkészíti az aláíró nyilvános kulcsát (pk) és a hozzátartozó titkos kulcsot (sk);

- χ egy polinom időben vakító algoritmus, melynek $m \in M$ üzenet a bemenete, a nyilvános kulcs pk és r egy véletlen sztring, majd megkonstruáljuk az m' üzenetet;

- σ polinom idejű aláíró algoritmus, amelynek bemenete egy m' vakított üzenet, és az sk titkos kulcs előállítja a vakaláírást az m' üzenetre;

- δ egy polinom időben visszaállító algoritmus, melynek bemenete egy s' vakaláírás és egy véletlen r sztring amely kitömöríti az s vakaláírást m-en;

- Γ egy polinomidejű aláírásellenőrző algoritmus, amelynek inputja egy (m, s) üzenetjelpár és pk nyilvános kulcs kimenete vagy *igen* vagy *nem*.

2.2.7. Bitműveletek

Tegyük fel, hogy Aliz küldeni akar egy b bitet Bobnak, és ezt úgy kívánja megtenni, hogy nem kívánja azt felfedni. Bob viszont követeli, hogy ezt Aliz ne tudja megváltoztatni a későbbiek folyamán.

[1]Képzeljük el az esetet, mikor a vizsgázó elküldi a „félig kész" azonosítóját a vizsgabiztoshoz, és ő vakon aláírja azt.

Először Aliz kódolja a b bitet és ezt elküldi Bobnak. Ezt követően Bob nem tudja visszaállítani a b-t, amíg Aliz el nem küldi neki a dekódoláshoz szükséges kulcsot. Ekkor a b titkosítását *blob*nak nevezzük. Általában egy bitművelet rendszere alatt egy függvényt értünk: $\xi\{0,1\} \times X \to Y$, ahol X, Y véges halmazok. Vagyis b egy kódolása nem más, mint a $\xi(b,k)$ bármely értéke, ahol $k \in X$. A bitműveleteknek a következő tulajdonságokat kell kielégítenie:

- *titkosítás* – Bob ne tudja meghatározni b értékét $\xi(b,k)$-ból;
- *kötés* – Aliz később meg tudja nyitni $\xi(b,k)$-t, b, k felhasználásával saját rendszerében. Továbbá ne legyen képes megnyitni a blobot egyszerre 0 és 1-ként sem.

Ha Aliz biztos akar lenni a bitsztringben, akkor biztosnak kell lennie minden egyes bit helyességében.

Azt a bitműveletet, amelyben Aliz képes megnyitni a blobot, mint 0 és mint 1-ként *csapóajtó bitműveletnek* nevezzük.

A bitműveletnek a következőket kell teljesítenie:

Tételezzük fel, hogy p egy nagy prím, Z_p g generátora és $G \in Z_p$ ismertek. G diszkrét logaritmusa a g bázison nem ismert Aliznak és Bobnak sem (G-t véletlenszerűen választottuk). A bitművelet egy $\xi : \{0,1\} \times Z_p \to Z_p$ függvény, melyre igaz:

$$\xi(b,k) = g^k G^b.$$

Legyen $log_g G = a$. A blob megnyitható, mint b, k feltárásával, és megnyithatjuk, mint $\neg b$ $k - a$ feltárásával, ha $b = 0$ vagy $k + a$, ha $b = 1$. Ha Aliz nem ismeri a-t, akkor nem képes megnyitni a blobot, $\neg b$-ként.

Másrészt, ha Bob nem ismeri k-t, akkor nem képes b-t meghatározni csak $\xi(b,k) = g^k G^b$-t látva.

A csapóajtó bitművelet rendszer abban az esetben áll fenn, ha a ismert Aliz számára.

Ha a ismert Bob számára és Aliz megnyitja a blobot a lehallgathatatlan csatornán Bobnak, akkor Bob képes hazudni egy harmadik személynek b-ről. Egyszerűen ő azt állítja, hogy fogadta $k - a$ vagy $k + a$-t k helyett. A bitművelet rendszere megengedi a vizsgálónak (Bob), hogy hazudjon a blob megnyitásáról, és ezt a műveletet *kaméleon bitműveletnek* nevezzük.

Az s sztring minden bitművelete helyett Aliz egyszerűbben el tudja végezni a következő $\xi(s,k) = G^s g^k$, ahol $0 \leq s < p$. Aztán a ismeretében Aliz képes lesz megnyitni a $\xi(s,k)$-t, bármely s', k'-ra, amely kielégíti az $as' + k = as' + k'$-t.

2.2.8. Homomorf kódolás

Tekintsünk egy probabilista kódolási rendszert. Legyen P az egyszerű üzenetek halmaza és legyen C a kódolt üzenetek halmaza, továbbá P és C is csoport a $\otimes$ bináris művelet felett. Legyen például E egy probabilista kódolási rendszer, és ez is elkészíti magának a saját nyilvános és titkos kulcsát. Legyen $E_r(m)$ az m üzenet kódoltja, ahol r (melyre igaz lehet, hogy $r \in E$) véletlenparaméterek(et) használunk a kódoláshoz.

Azt mondjuk, hogy a probabilista kódolási rendszer homomorf az $(\otimes, \oplus)$ műveletre, ha például E kódoló rendszer, legyen $c_1 = E_{r_1}(m_1)$ és $c_2 = E_{r_2}(m_2)$, melyekre létezik r, hogy

$$c_1 \otimes c_2 = E_r(m_1 \oplus m_2).$$

Például, az ElGamal kódolási rendszer homomorf. Itt, P az egészek modulo p ($P = Z_p$) halmaz, és $C = \{(a, b)|a, b \in Z_p\}$ elemek halmaza. A $\oplus$ művelet szorzás modulo p. Az $\otimes$ művelet a kódolt szövegeken definiálva van, mint szorzás minden komponenssel, majd modulo p. Ha a két egyszerű szöveg m_0 és m_1, akkor a kódoltjuk:

$$E_{k_0}(m_0) = (g^{k_0}, h^{k_0} m_0)$$

és

$$E_{k_1}(m_1) = (g^{k_1}, h^{k_1} m_1),$$

ahol k_0 és k_1 véletlenszámok.

Ezek megtartják a következő tulajdonságot:

$$E_{k_0}(m_0)E_{k_1}(m_1) = (g^{k_0} g^{k_1}, h^{k_0} h^{k_1} m_0 m_1) = (g^k, h^k m_0 m_1) = E_k(m_0 m_1),$$

ahol $k = k_0 + k_1$.

Ennek következtében az ElGamal kriptorendszerben a kódolt szövegek szorzásával elérhetjük a megfelelő egyszerű szövegek szorzását is.

2.2.9. A robosztus küszöb ElGamal kriptorendszer

Ennek a kriptorendszernek az a célja, hogy a titkos kulcsot megossza a vizsgabiztosok között úgy, hogy az üzeneteket csak akkor lehessen dekódolni, ha a legfontosabb vizsgabiztosok együttműködnek. Szükségünk van a kulcsgenerátor cseréjére és a dekódoló protokollra az ElGamal kriptorendszerben. Az üzeneteket a szokásos módon kódolhatjuk.

Az egész folyamat két fontos lépésből áll:

1. **Kulcsgenerálás**

   A kulcsgenerálási protokoll eredménye az, hogy minden A_j vizsgabiztos birtokolni fogja az s titok egy s_j részét (a privát kulcsot az ElGamal kriptorendszerben) és a nyilvános kulcs publikusan lesz elkészítve. A vizsgabiztosok megkapják a nyilvános $h_j = g^{s_j}$ értékeiket. Továbbá, az s_j megosztásokból az s titok rekonstruálható bármely $t + 1$ megosztásból. Bármely t megosztáshalmaz nem mond semmit az s titokról. A megvalósításhoz Shamir $(t + 1, N)$ titokmegosztórendszere használatos. Egy megbízható harmadik személynek számítania és elosztania kell ezeket a titokrészeket a vizsgabiztosoknak úgy, hogy a lehallgathahtalan csatornát használja.

   Így megtartja a következő tulajdonságot:

   $$s = \sum_{j \in A} s_j \lambda_{j,A},$$

   ahol

   $$\lambda_{j,A} = \prod_{l \in A - \{j\}} \frac{l}{l - j}.$$

   A nyilvános kulcs pedig: (p, g, h), ahol $h = g^s$.

2. **Dekódolás**

   A dekódoláshoz a $(x, y) = (g^k, h^k m)$ titkosított szöveget az s titok feltárása nélkül, a hatóságoknak a következőket kell tenniük:

   (a) Minden A_j vizsgabiztosnak szétosztják a $w_j = x^{s_j}$-t, és zéró tudás nélkül be kell bizonyítaniuk, hogy

   $$\log_g h_j = \log_x w_j.$$

   (b) Legyen A bármely $t + 1$ biztos halmaza, akik elfogadják a zéró tudást. Ekkor az egyszerű szöveg visszaállítható, mint

   $$m = \frac{y}{x^s},$$

   és

   $$x^s = x^{\sum_{j \in A} s_j \lambda_{j,A}} = \prod_{j \in A} w_j^{\lambda_{j,A}}.$$

A legtöbb t biztosnál az s_j titkok felfedhetőek, mintha $t+1$-ből ismernénk az s_j értékeket, mint egy titkos kulcsot és számítanánk (Lagrange-interpolációt használva), majd az üzenetet közvetlenül kinyernénk ElGamal dekódolással.

2.2.10. Mixhálók

A mixhálók alapötlete a keverés és módosítás (például dekódolás vagy kódolás) néhány objektumszekvenciát, hogy elrejtsük az összefüggést az eredeti és végső elemek sorrendje között. David Chaum javasolta ezt az ötletet 1981-ben, mint a névtelen csatorna egy megvalósítását [4].

Van n darab keverőszerverünk, melyek nevei: $M_1, \ldots, M_n$, és mindegyiknek van saját E_j nyilvános és D_j titkos kulcsa. Mikor valaki akar egy m üzenetet küldeni a névtelen csatornán át, akkor kódolnia kell az

$$E_1(E_2(\ldots(E_n(m))\ldots)$$

értéket és elküldenie M_1-nek.

M_1 addig vár, amíg több kódolt üzenetet kap. Ezt követően fogadja őket és eltávolítja a kódolás egy szintjét, szabályosan megkeveri, majd elküldi őket az M_2-nek.

Az M_j-edik keverőszerver fogadja a kódolt üzeneteket. Eltávolítja a kódolás egy szintjét, megkeveri őket és elküldi $E_{j+1}(E_{j+2}(\ldots E_n(m))\ldots)$-t M_{j+1}-nek. Az utolsó keverőszerver dekódolja az üzenetet és elküldi őket a címzettekhez.

Megkövetelhetjük, hogy a keverőszerver adja bizonyítékát az üzenetek helyes keverésének és kódolásának.

3. fejezet

A vizsgáztatórendszer elméleti modellje

3.1. A rendszer emberi komponensei

A vizsgázásban, mint folyamatban személyek három nagy csoportja vesz részt. Ők a *vizsgázó(k*), a vizsgáztató *tanár(ok)* és az őket felügyelő *vizsgabiztos(ok)*. Mint tudjuk, ezek a halmazok a vizsgázás idején diszjunktak és egyaránt mindenkinek mindenkit lehet ellenőrizni. Ebben a fejezetben azt vizsgáljuk, miként lehet ezt megtenni elméletben, matematikai eszközök segítségével. Ezt követően pedig a három csoport érdekeit egyesítve, a lehető legmegbízhatóbb utat keressük a három csoport érdekeinek a lehető legjobb egyesítésére.

3.1.1. A vizsgázó

A vizsgázó szempontjából a következő fontos szempontokat kell vizsgálnunk:

- jogosult-e valaki az aktuális vizsgára;
- rendelkezik-e az aktuális vizsgára feljogosító azonosítóval;
- levizsgázott-e;
- panaszt nyújtott-e be a vizsgabizottsághoz;
- az érdemjegyet kézhez kapta-e.

Mindezen szempont a vizsgázó részéről garantálja azt, hogy a vizsgázásban nem történt kihágás, vagy más módon való visszaélés sem a vizsgázó, sem a vizsgáztató tanár vagy a vizsgabiztos részéről.

A vizsgázó részéről szinte a legfontosabb tulajdonság az, hogy egyáltalán jogosult-e vizsgát tenni. Amennyiben ez fennáll, akkor kapjon a vizsgabiztostól egy hitelesített azonosítót és „írja rá” ezt a dolgozatára. Ellenkező esetben, ha valamely személy esetében ez nem áll fenn, akkor legyen kizárva a vizsgázás teljes folyamatából. Ez utóbbi gondolat leginkább a más személyazonossággal rendelkező, vizsgát tenni akaró személyek kizárására vonatkozik. Továbbá megköveteljük, hogy a vizsgabiztos által hitelesített azonosító teljes anonimitást biztosítson a vizsga egész ideje alatt, és a továbbiakban ne lehessen újrafelhasználni azt.

Ha egy vizsgát tenni kívánó személy rendelkezik azonosítóval, az még nem garantálja azt, hogy ténylegesen is jogosult vizsgát tenni. Itt elegendő csak arra gondolni, hogy az azonosító formátumát valaki ismeri és készít egy vele megegyezőt vagy egy már vizsgázásnál használt azonosítót akarnak újrafelhasználni. A rendszernek biztosítania kell az azonosítók ellenőrizhetőségét is, valamint a bizonytalan erdetű azonosítók esetleges cseréjét is.

Amennyiben valaki levizsgázott, akkor ezt valamilyen jelzéssel a vizsgabizottság tudomására kell hoznia. Ezt például megteheti úgy, hogy az általa kapott azonosító rendelkezik egy időbélyeggel és ez csak egy adott időintervallumban érvényes. Amennyiben, a vizsgázó idő előtt befejezi dolgozatát, elküldheti a vizsgabizottságnak, mely garantálja számára a vizsgázás folyamatának érdemi végét. Továbbá meg lehet azt is engedni, hogyha valakinek eszébe jutott még egy pár gondolat, és a vizsga ideje még nem telt le, akkor elküldhesse újra dolgozatát, bár ezt a lehetőséget célszerű korlátozni. Amennyiben a vizsgázásra kiírt idő letelt, és a vizsgázó még nem küldött be dolgozatot, lehetőséget kell rá adni, hogy ezt legfeljebb egyszer megtehesse.

Ha valaki szeretne élni panaszbenyújtással, lehetőséget kell adni számára. Ebben az esetben vizsgálni kell, hogy az illető személy jogosult volt-e az aktuális vizsgára, rendelkezett-e arra a vizsgára érvényes azonosítóval, és valóban levizsgázott-e. Amennyiben ezek a feltételek teljesültek, akkor a vizsgabiztos újraellenőriztetheti a dolgozatát a javító tanárral.

A vizsgázás folyamata végén a vizsgázó érdemjegyet kap. Amennyiben elégedetlen az általa kapott jeggyel, akkor fennáll az a lehetőség, hogy a vizsgázó kétségbe vonhatja a vizsgabiztosok vezetői képességeit, és külső segítséget kérhet. Erre is megvan a számára fenntartott lehetőség[1], viszont ennek a dolgozatnak nem témája az ilyen, és az ehhez hasonló esetek ismertetése. Amennyiben elfogadja az érdemjegyet, akkor ez regisztrálásra kerül, amelyről hivatalos elismervényt kap.

A vizsgázó szemszögéből ezek a szempontok fontosak mind vizsgázási, mind személyiségi jogok kapcsán. Ezen feltételek biztosítják számára az anonimitást, a vizsgázást, valamint jogot a panasz benyújtásának lehetőségére is.

[1]Például fordulhat a helyi oktatási ügyekkel foglalkozó képviselőhöz, vagy az ombdusmanhoz.

3.1.2. A javító tanár

A javító tanár szempontjából a következő szempontok a legfontosabbak:

- jogosult-e javító tanárnak;
- rendelkezik-e az adott tárgy javításához szükséges azonosítóval;
- javíthatja-e az adott dolgozatot;
- értékelheti-e a adott dolgozatot.

Az első pont érdekes gondolatot fejt ki. Jogosult-e egy tanár javítani egy adott tárgyból és értékelni? Itt elegendő csak arra gondolni, hogy az adott személy rész vett-e a megfelelő továbbképzéseken. Amennyiben nem, akkor már az akkreditáció folyamatában ki kell zárnunk. Ha teljesített minden követelményt (melyet ellenőrizhetünk elektronikus, postai vagy akár személyes úton is), akkor meg kell kapnia a megfelelő tantárgyi besorolást és a hozzá tartozó egyéni azonosítót is.

Ha egy tanár „átvilágított” az akkreditáció folyamatán, még akkor sem lehetünk biztosak abban, hogy hitelesített javító tanári azonosítóval rendelkezik, így mindig lehetőséget kell biztosítanunk a rendszeren belül bárkinek, hogy ellenőrizhesse, valóban jogosult-e javítani az adott tárgy dolgozatait.

Ha egy javító tanár dolgozatot javít, nem biztos, hogy csak olyan dolgozat kerül hozzá, amelyet ő javíthat. Például egy javító tanárhoz kerülhet egy véletlenül elküldött dolgozat is (például a rendszer hibájából adódóan), amelyet ő nem javíthatna. Amennyiben ezt mégis megteszi, akkor ez a dolgozat legyen újraértékelve vagy figyelmen kívül hagyva. Amennyiben ezt javíthatja és értékelheti, akkor kézjegyével el kell látnia azt.

Az értékelési folyamat szinte a legfontosabb egy javító tanár számára. Vajon az a tanár értékeli-e, aki hivatott-e rá? Ezt csak a vizsgabizottság által kiadott azonosító döntheti el, amely nyilvánosan ellenőrizhető. Így kizárhatjuk azt az esetet, hogy valaki más javítja, és valaki más írja rá a kapott érdemjegyet. Akkor áll összhangban a folyamat, ha a javítást és az értékelést csak az adott javító tanár végzi el.

3.1.3. A vizsgabiztos

A vizsgáztatás folyamatában ő vállalja a munka oroszlánrészét. Összehangolja a vizsgáztatást, mint folyamatot, ellenőrzi a tanár és a vizsgázó jogosultságát, azonosítókat oszt ki, és meggyződik a megfelelő jogosultságok teljesüléséről. Őt már csak az állami szervek tudják felügyelni.

Azonban őt is ellenőrizni kell, az esetleges szabálytalanságok megakadályozása érdekében. Mivel a vizsgáztatás legfelsőbb pillérén helyezkedik el, így itt könnyebb szabálytalanságokat elkövetni.

Sajnos feltételezni kell azt, ha egy folyamat ellenőrizve van egy biztos által, akkor őt is ellenőrizni kell. Gondoljunk csak arra, hogy a biztosok egy része titokban szövetkezik, és megpróbál „kedvezőbb” érdemjegyeket osztani a tanári érvekkel ellentétben.

Itt a következő szempontokra kell figyelni:

- jogosult-e vizsgabizottsági tagnak;
- érvényes azonosítóval rendelkezik-e.

Az első fenti gondolat nemcsak alkalmassági, hanem etikai kérdést is felvet. Sajnos ki kell zárni minden megfelelő képesítés nélküli, vagy más úton gyanúba keveredett személyt. Mivel itt a rendszer legfelsőbb pillérén állunk, itt igen nehéz dolgunk akad. Viszont, ez megtehető bármely állami szerv segítségével is. Például, csak állami szerv által kijelölt személyek vehetnek részt a vizsgabizottság munkájában.

Amennyiben rendelkezik azonosítóval, az még nem biztos, hogy vizsgabiztos is. Fennállhat minden esetben, hogy egy „ismeretlen személy” látszólag érvényes vizsgabiztosi azonosítóval rendelkezik, és mégsem vizsgabiztos. Ekkor meg kell győződnünk arról, hogy valóban jogosult-e tagnak lenni, és ha ez nem teljesül, akkor azonnali hatállyal ki kell zárni a vizsgabiztosok soraiból, és ezen okból kifolyólag megfontolni teljes vizsga újrakezdésének lehetőségét is.

Itt szeretném megemlíteni, hogy azonosítóval rendelkezik ugyan, de megengedett, hogy személye ismert legyen a vizsgázás teljes folyamata alatt.

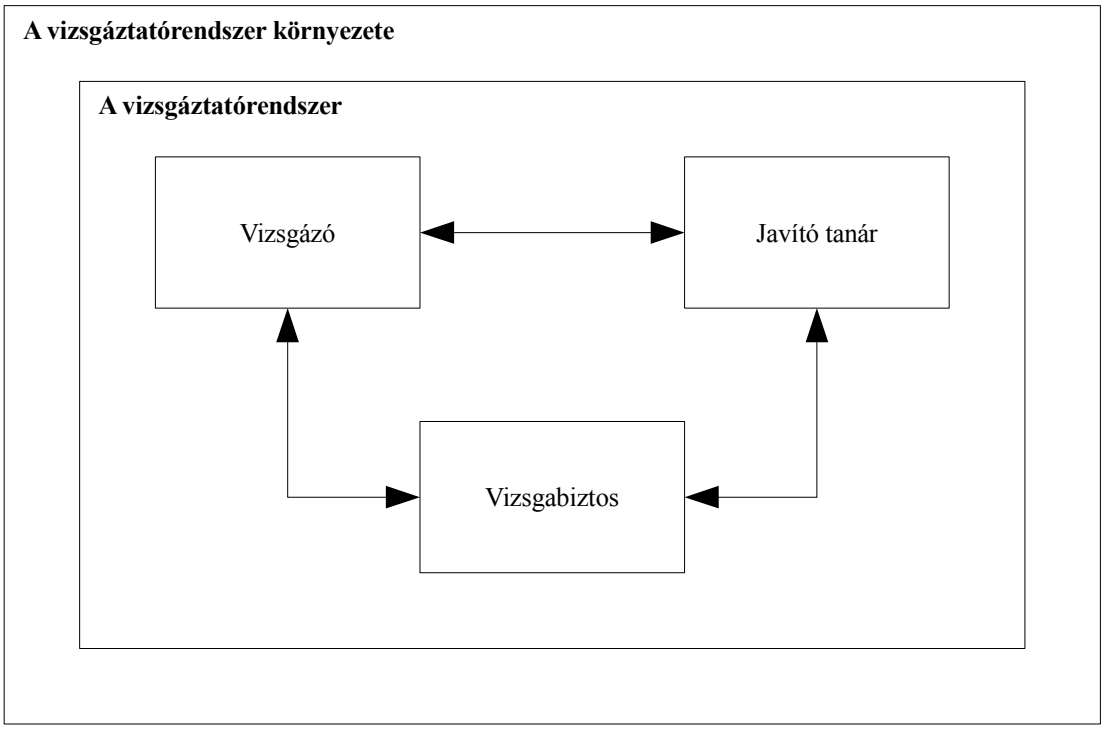
3.2. A rendszer gépi komponensei

A gépi komponensek tekintetében csak egyetlen elemünk van. Ez pedig nem más, mint az emberi komponenseket összekötő kommunikációs csatorna, melynek nemcsak a kommunikációt, hanem a közöttük lévő információcsere titkosságát is biztosítania kell. Ahhoz, hogy ez megvalósítható legyen, az előző fejezetben láthattunk rá tíz különböző módszert, melyekből teteszőlegesen – akár kombinálva is – választhatunk.

Ezen rendszerek kapcsolattartásához pedig segítséget nyújt az első fejezetben bemutatott három kommunikációs módszer, melyek segítségével kereshetjük a megoldást. Mindhárom kommunikációs formának megvannak a maga előnyei és hátrányai, de bármely módszert is választjuk ki, biztosnak kell lennünk benne, hogy mindenki csak azt kapja, amit számára küldtek, továbbá mindenki csak annyit „észlel” a folyamatból, amennyit megengedünk neki.

3.3. A vázlatos modell

A vizsgáztatórendszerünk modelljének megkonstruálását talán kezdhetnénk egy összefüggéseket tartalmazó ábra készítésével is, amely feltárja a kapcsolatokat az emberi és gépi komponensek között.



3.1. ábra. A rendszer elvi felépítése

Figyeljük meg a a 3.1. ábrát. Egyszerűsége ellenére nagyon jól tükrözi rendszerünk bonyolultságát. Vegyük egyenként részeit, haladjunk belülről kifelé.

Az ábrából jól kitűnik, hogy az emberi és gépi komponensek együtt alkotják a rendszer egy egészét és szoros összefüggésben állnak egymással. A három emberi komponens legbelül helyezkedik el, közöttük a nyilak jelképezik a gépi összetevőt, a kommunikációs csatornát. Az ábra jól mutatja a három emberi résztvő teljes függetlenségét egymástól, azonban a vizsgáztatórendszerben való elhelyezkedésük miatt azzal szoros kapcsolatban állnak.

Nagyon jól kitűnik, hogy a kompononensek közötti „egy vezeték elszakadása” vagy „megcsapolása” még nem veszélyezteti a vizsgáztatórendszer teljes működését, de ez a rendszerben könnyen anomáliához vezethet. Ezért nagyon fontos a kommunikációs csatorna biztonságának kérdése, valamint egy „szakadás esetén” a megfelelő tartalékhálózat esetenkénti biztosítása.

Amennyiben egy szinttel feljebb lépünk, akkor a három emberi és az egy gépi komponens alkotja a vizsgáztatórendszerünket. Ez az a rendszer, melynek megalkotására teszünk kísérletet, s mondhatnánk is, hogy: „Mi elvégeztük a dolgunkat.” Azonban ez nem ilyen egyszerű, mert a rendszerünk is egy nagyobb rendszer kisebb része.

Mint minden rendszer ez is egy nagyobb rendszer része, melyet nevezhetünk valós világnak is. Az ábrán ezt a vizsgáztatórendszer környezeteként láthatjuk. Azért fontos ezt is megemlíteni, mert minden rendszernek egyedi viselkedése van és ez kihat a többi rendszerre egyaránt. Így jól tükrözi az ábra azt a fenti gondolatot, hogy a vizsgáztatórendszerünk „a rendszer egy egésze”. Ezen a szinten működnek a 3.1.3. alfejezetben leírt állami szervek és ellenőrző bizottságok is.

3.4. A probléma formalizálása

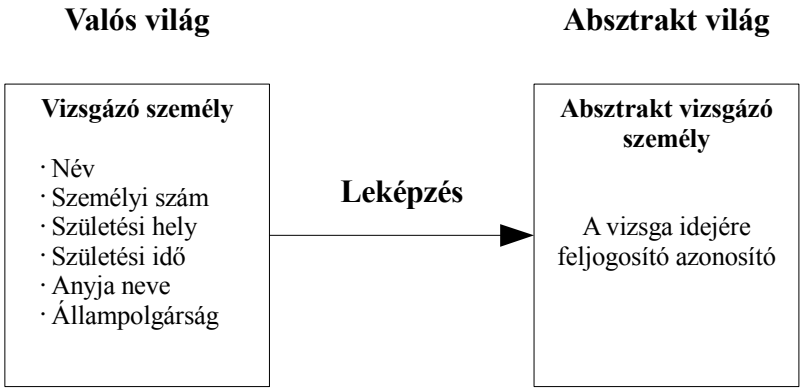
Ebben a szekcióban modellezni fogjuk az emberi és gépi résztvevőinket, mégpedig úgy, hogy az általunk fontosnak tartott tulajdonságaikat kiemeljük, a modellezés szempontjából kevéssé fontosakat pedig elhanyagoljuk.

Természetesen minden résztvevőnek más és más tulajdonságát kell modellezni, így ez a rész a legfontosabb, mert itt válnak az informatika számára „kézzel foghatóvá” a valós világ személyei és eseményei.

3.4.1. A vizsgázó

Minden vizsgázó rendelkezik személyazonossággal. Itt többek között a név (leánykori név), születési hely és idő, anyja neve, személyi szám és állampolgárság jöhet elsőként számításba. Továbbiakban figyelembe vehetjük a lakhely, ideiglenes lakhely, telefonszám, e-mail cím, TB azonosító, stb. adatokat is, bár ezek vizsgáztatórendszerünk szempontjából kevéssé fontosak.

Azonban, a vizsgáztatórendszer szempontjából ezek az adatok teljesen értelmezhetetlenek, hiszen hiába azonosítják jól a vizsgázót, ez túl sok adat tárolását (titkosítását) vonná maga után. Éppen ezért célszerű ezeket az adatokat felhasználva egy *egyedi azonosító* számot adni a vizsgázónak, melyet „rá tud írni” dolgozatára a vizsga végeztével, majd azt követően automatikusan *érvénytelenné* válik. Ezt a folyamatot a a 3.2. ábra szemlélteti.



3.2. ábra. A vizsgázó bekerülése a rendszerbe

Jól látható az ábrán, hogy több adatot összevonunk és abból készítünk egy azonosítót, melynek a következő tulajdonságokkal kell rendelkeznie:

- egyedi;
- anonimitást biztosít a vizsga ideje alatt;
- a vizsga végén automatikusan érvénytelenné válik.

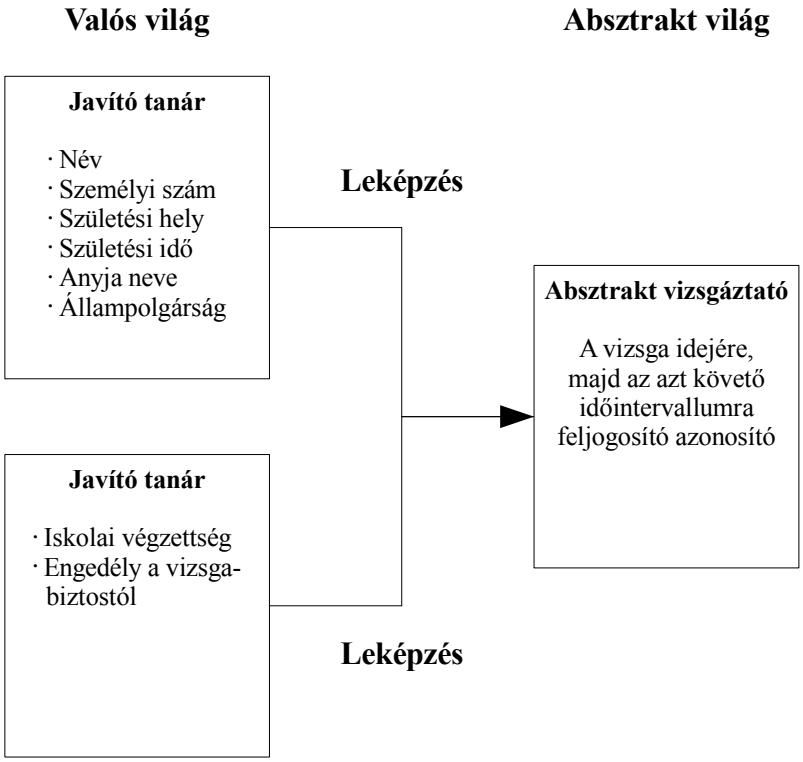
Ezek nagyon fontos szempontok, és bármely tulajdonság be nem teljesülése komplikációkat okozhat. Egyedinek kell lennie, mert így egyértelmű a vizsgázó személye, anonim, mert nem tudhatja senki a vizsga végéig, hogy ki melyik dolgozatot adta be, és automatikusan érvényét kell vesztenie a vizsga végén, nehogy valaki ezzel próbálkozzon dolgozatot beadni vagy csak a rendszerbe bekerülni. Itt meg lehet engedni azt, hogy a lejárt azonosítót esetenként tároljuk.

Továbbá egy nagyon fontos megállapítás – tervezési szempontból –, amit feltétlenül figyelembe kell vennünk: a vizsgafolyamat végén a vizsgázónak érdekében áll, hogy személyazonosságát felfedje.

Ez egy nagyon fontos része a rendszernek, és jól kigondoltnak kell lennie. A kérdés a következő: „Hogyan lehetne előállítani egy ilyen azonosítót?” Ennek egy lehetséges megvalósítását olvashatjuk a függelékben, az 55. oldalon.

3.4.2. A javító tanár

A javító tanár modellezése is hasonló folyamat, mint a vizsgázó esetében. Itt a modellezés folyamán azonban még néhány pluszinformációra is szükség lesz. Ez többek között a javítható tantárgyak listája és egy jóváhagyási engedély a vizsgabiztos részéről. Ennek egy lehetséges leképzése a 3.3. ábrán látható.



3.3. ábra. A javító tanár bekerülése a rendszerbe

Ezzel az azonosítóval a vizsga idejére, illetve azt követően, a tanár feljogosítást kap a számára beérkezett, javítható dolgozatok javítására és értékelésére. A tanári azonosítónak természetesen különböznie kell a vizsgázók azonosítójától, különben fennállhat a veszélye annak, hogy valaki saját magának javíthatja ki a dolgozatát, ami teljességgel megengedhetetlen. A tanári azonosítókkal szemben támasztott követelmények:

- egyedi;
- anonimitást biztosít a vizsga ideje alatt;
- a vizsga végén ne váljon automatikusan érvénytelenné;
- javítást, értékelést, panaszbenyújtást, érdemjegy osztást lehetővé kell tennie, mielőtt érvénytelenné válna.

Az első két pont megegyezik a vizsgázók témakörben leírtakkal. Azonban, az utolsó két pont magyarázatra szorul. Ezen két pont oka inkább biztonsági, valamint a későbbiek folyamán ellenőrizhetőségi jellegű. Mikor megkezdődik a vizsga, a vizsgabiztos segítségével elkészítik azonosítóikat. Ez azt jelenti, hogy a tanárok már a vizsga kezdetén tudni fogják, hogy ki melyik tárgy dolgozatát fogja javítani, majd értékelni. Erre azért van szükség, hogy a későbbiekben

esetlegesen előforduló szabálytalanságokat megakadályozzák. Továbbá a tanári azonosítók számára még megengedhetjük azt is, hogy a vizsgát követően még érvényesek legyenek egy előre rögzített időintervallumban is.

Amennyiben a vizsgázó levizsgázott, és a dolgozatát elküldte a biztosnak, akkor a tanárnak javítania és értékelnie kell azt, majd kézjegyével ellátni. Ez azt jelenti, hogy a vizsga lejárta után még rendelkezésre kell állnia egy olyan érvényességi időintervallumban, melyben még feljogosítja a tanárt az előbb megnevezett műveletek elvégzésére. Továbbá lehetővé kell tennie azt is, hogy panasz benyújtása esetén bizonyítani lehessen a javító tanár személyét is. Ennek egyik lehetséges formája, hogy a tanári azonosítókat meg kell őrizni egy bizonyos, előre meghatározott ideig[2].

Továbbá – eltérően a vizsgázótól –, a tanárnak nem áll érdekében személyazonosságának felfedése a vizsgafolyamat végén, csak abban az esetben, ha vizsgabiztos visszaélést tapasztal, vagy megállapítja szakmai inkompetenciáját.

Az azonosítót hasonlóan is elkészíthetjük, mint az előző alfejezetben, természetesen hozzávéve az ahhoz szükséges információkat. Erre is láthatunk egy példát a függelék 56. oldalán.

3.4.3. A vizsgabiztos

A vizsgabiztos irányítja és menedzseli a vizsgázás teljes folyamatát, felügyeli az abban résztvevő személyek jogosultságait és azonosítókat hitelesít számukra. Ennek ismeretében elmondhatjuk, hogy a vizsgáztatórendszer emberi komponensei közül az ő munkájuk a legfontosabb. Így az ő modellezésüket a rendszerben különös odafigyeléssel kell véghezvinnünk.

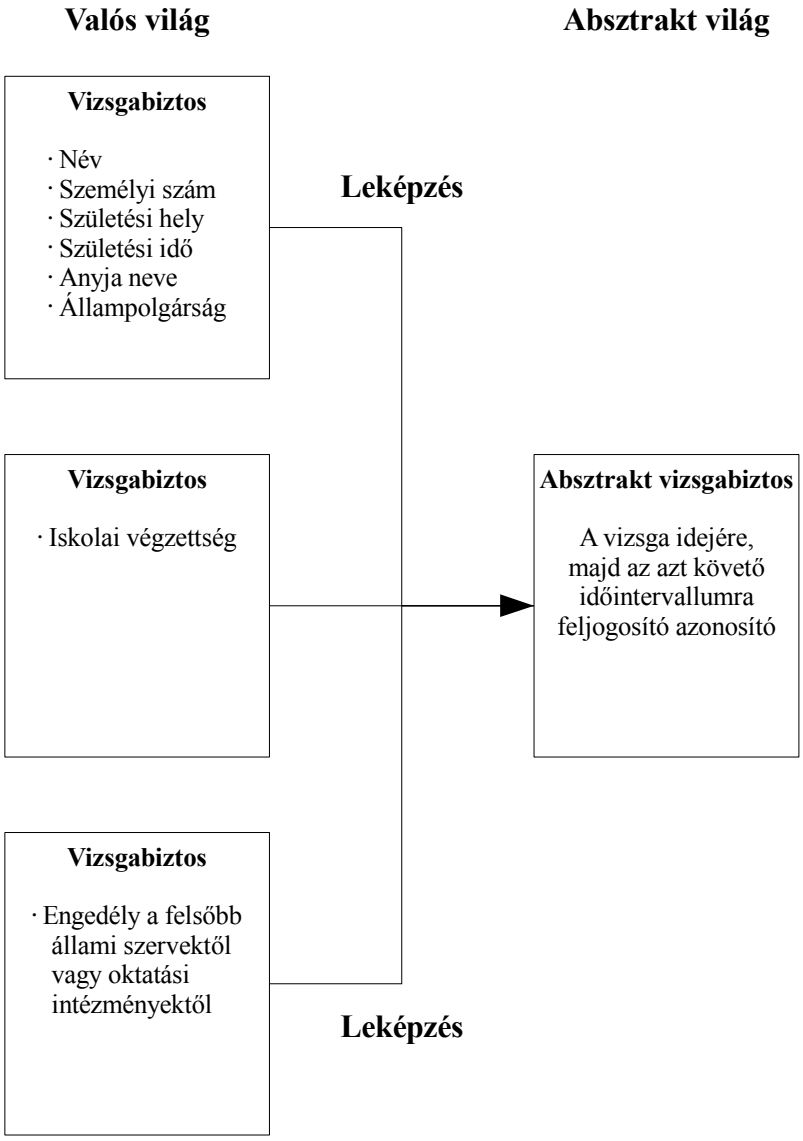
Mivel ők állnak a „jéghegy csúcsán”, így a teljes felelősséget a vizsgák helyes és teljes lebonyolításáért ők vállalják. Bármilyen panasz érkezik hozzá, csak az ő segítségével lehet azt elbírálni, majd jóváhagyatni. Amennyiben nem érkezik panasz a kijavított dolgozatok kapcsán, akkor az érdemjegyet tartalmazó dolgozatokat kézjegyükkel kell ellátni. Így nekik is szükséges egy azonosító. Természetesen különbözni kell a vizsgázó vagy tanári kézjegytől. Egy fontos szempont, amit itt újra meg kell említenünk: rendelkezik azonosítóval, bár személyének anonimitását nem feltétlenül kell biztosítanunk a vizsgázás ideje alatt[3].

A rendszerbe bekerüléskor ugyanazokra a képesítésekre van szüksége, mint egy javítást végző tanárnak, továbbá rendelkeznie kell egy felsőbb szintről kapott – például állami szerv által kibocsátott – engedéllyel is. Ezt a 3.4. ábra szemlélteti.

Az ábrán jól látható, hogy itt már három információhalmazból kell valamilyen azonosítót elkészíteni a vizsgabiztos részére. A vizsgabiztos részére készítendő azonosítónak a következő

[2]Ez ma is megfigyelhető a magyar közoktatási rendszerben a dolgozatok tárolásának időtartamát illetően.

[3]Erre a legtipikusabb példa az érettségi. A tanárok általában előre közlik, hogy ki lesz az elnök és melyik intézményből fog érkezni.



3.4. ábra. A vizsgabiztos bekerülése a rendszerbe

tulajdonságokkal kell rendelkeznie:

- egyedi;
- nem válhat automatikusan érvénytelenné;
- javítást, értékelést, panaszbenyújtást, érdemjegy osztást lehetővé kell tennie;
- akár tartós használatra is alkalmasnak kell lennie.

Az egyedi tulajdonság megegyezik az előző két alfejezetben közöltekkel. Ez biztosítja azt, hogy ne lehessen egy vizsgabiztost „összetéveszteni” egy vizsgázóval vagy egy javító tanárral.

A második gondolat egy érdekes kérdést vet fel. Miért ne váljon automatikusan érvénytelenné? Erre a válasz az előző alfejezetben leírtakhoz hasonlóan, a biztonság és a későbbi ellenőrizhetőség miatt van szükség. Továbbá a kijavított és értékelt dolgozatok aláírásához is szükséges, mert a vizsgadolgozat csak ezzel az aláírással együtt érvényes, továbbá az érdemjegyről szóló elismervénynek is tartalmaznia kell.

Az utolsó pont inkább egy kissé filozofikus kérdést vet fel. Képzeljük el azt az esetet, hogy tartósabb ideig ugyanaz a vizsgabizottság és ugyanazokból a tárgyakból vezetik a vizsgázást. Ekkor felmerülhet a kérdés, hogy kell-e számukra új azonosító vagy a régit használják. Ilyen esetekben – mivel saját maguk részére állítják elő, vagy részükre generálják – célszerűbb egy azonosítót használni, mint minden alkalommal újat generáltatni, s ezáltal csökkenthető a vezető pozíciókban való esetleges kihágások elkövetése. Természetesen, egy adott idő után célszerű ezeket az azonosítókat is lecserélni, az esetleges kihágások elkerülésének érdekében.

3.5. A vizsgáztató rendszer

A rendszer, mely a vizsgáztatást már gépi úton végzi, szabályok közbeiktatásával fog működni. Ebben az alfejezetben össze kell gyűjtenünk minden olyan szabályt, amelyek mentén működésre lehet hívni egy olyan rendszert, mely az előre definiált szablyok segítségével, az emberi beavatkozás minimalizálásával képes elvégezni a rá kiszabott feladatokat.

Az elvárásaink a következőek:

- megbízható;
- egységes;
- rugalmas;
- robosztus;
- determinisztikus;
- tartós rendelkezésre állás.

Az első szempont röviden azt fejezi ki, hogy a rendszerünknek egy esetleges áramkimaradás, hardverhiba, vagy bármilyen emberek által elkövetett mulasztást követően is működőképesnek kell lennie – *adatredundancia* kialakulása nélkül –, továbbá működését biztosítania kell minden körülmények között.

A második fő szempontunk azt mondja ki, hogy a rendszernek mindenhol egyformán kell működnie. Nem állhat fenn az a lehetőség, hogy a rendszer egyik pontján vizsgázóként, a másik pontján vizsgáztatóként lép be valaki ugyanazzal az azonosítóval.

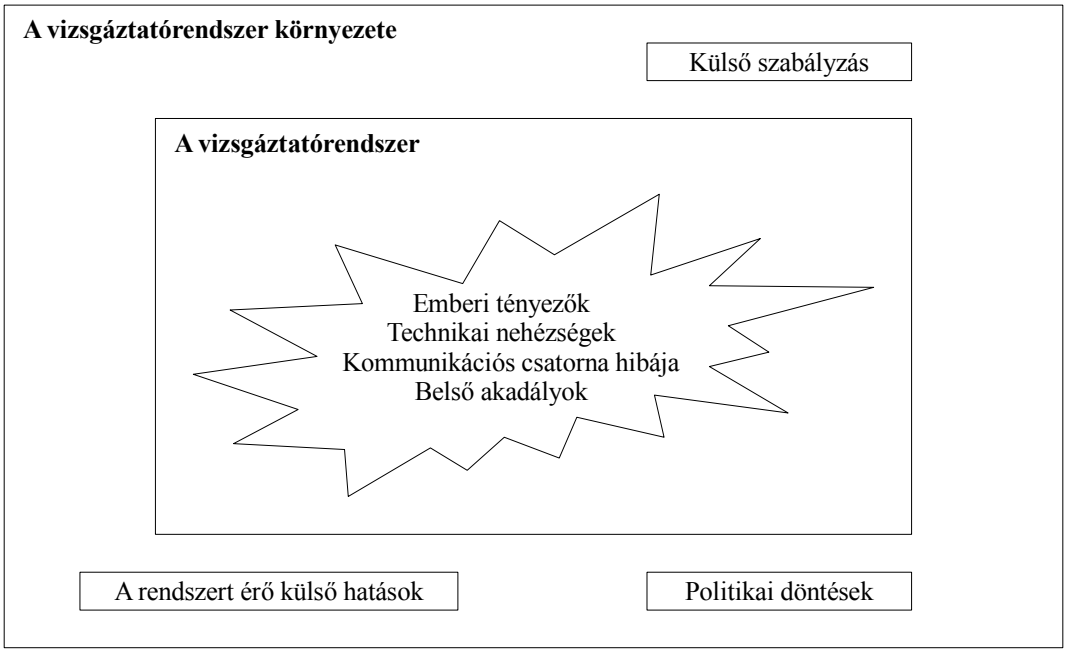
A harmadik szempontunk azt mondja ki, hogy ha egy új törvény vagy szabály lép életbe, akkor a rendszer minimális módosításával – lehetőleg egy ponton –, a rendszer tovább tudja folytatni működését.

A negyedik szempont azt a követelményt támasztja a rendszerünkkel szemben, hogy kellően sok felhasználó esetén is tudja végezni feladatát, és ez ne vezessen anomáliához.

A determinisztikusság kérdése szinte a legfontosabb. És a mi szempontunkból is az. A rendszernek mindenhol és mindig – a rendelkezésreállási idő alatt – az előre megadott szabályok szerint kell működnie, a mi előírásaink szerint az elvárt eredeményt szolgáltatva.

Az utolsó szempont azt fejezi ki, hogy akár huzamosabb ideig is működőképesnek kell lennie, s az esetleges rendelkezésre nem állási idő letelte után az üzembe állítása ne okozzon komplikációkat.

Azt, hogy rendszerünket milyen behatások érhetik, és mi ellen kell helyt állniuk, ezt a legjobban a a 3.5. ábrával szemléltethetjük.



3.5. ábra. A vizsgáztatórendszert érhető külső és belső hatások

3.5.1. A vizsgáztatáshoz használt papír

Az elektronikusan történő vizsgázásnál nyilván a hagyományos papír szóba sem jöhet, talán csak piszkozati papírként. A vizsgáztatást valamilyen „elektronikus papíron” kellene megoldani. Szerencsére napjainkban igen sokféle formája létezik az elektronikus papírnak. Gondoljunk csak bele abba, hogy a szövegszerkesztéshez használt papír is elektronikus jellegű[4], vagy az Interneten való barangoláskor a felbukkanó honlapok, vagy elektronikus levelek írásakor is elektronikus papírt használunk, sőt, tekinthetünk annak akár egy dialógusablakot is.

Célszerű ezeket a megoldásokat figyelembe venni az elektronikus papír kiválasztásánál. Az egyik legjobb választásnak a *html* alapú elektronikus papír tűnik, mert nemcsak hiperformázást tesz lehetővé, hanem megfelelő protokollt választva, még a rajta lévő információ tartalma is titkosítható.

Ezt követően fel kell ruházni egy olyan módszerrel is, mely egyfajta zárként üzemel rajta. Erre azért van szükség, hogy csak az arra illetékes személy(ek) tudják olvasni. Sőt akár megengedhetjük azt is, hogy a felnyitások regisztrálva legyenek.

[4]Természetesen, a mai formájában tekintve a szövegszerkesztést.

4. fejezet

A vizsgáztatórendszer egy lehetséges elméleti megközelítése

4.1. A vizsgáztatórendszer modelljének megközelítése

Ebben a fejezetben kísérletet teszünk egy vizsgáztatórendszer megkonstruálására, a matematika eszközeit felhasználva. A legfőbb cél az, hogy egy biztonságos vizsgáztatási protokollt hozzunk létre, lehetőleg minimális emberi beavatkozást igénylően.

Hogyan kellene megtervezni ezt a rendszert, vetődhet fel a kérdés. Az első gondolat talán az lehetne, hogy gyűjtsük össze az igényeinket és az ahhoz szükséges megszorításokat, majd ennek mentén építsük fel a teljes vizsgáztatórendszert. Ebben az esetben mondhatnánk, hogy nagyon nehéz dolgunk akad, ugyanis az egész rendszer megtervezése elsőre nagyon nehéz, figyelembe véve az általunk támasztott követelményeket. Éppen ezért ezt az ötletet célszerű elvetni.

Azonban akad egy másik módszer is, amelyet alkalmazni lehetne. Ez a módszer pedig nem más, mint a mesterséges intelligencia területén nagyon jól ismert *problémaredukciós technika*. Ennek lényege az, hogy van egy nagy problémánk. Ezt követően – közvetlen megoldás hiányában – osszuk fel ezt a problémát diszjunkt halmazokra, és azokat a részproblémákat oldjuk meg külön. Amennyiben ezeket sem sikerül megoldani, akkor osszuk fel még további részproblémákra mindaddig, amíg úgynevezett *egyszerű problémákhoz* nem jutunk.

Ha egy egyszerű problémát megoldottunk, akkor vegyük a többi egyszerű problémát, oldjuk meg, majd azt követően állítsuk össze a részproblémát. Ha az összes részproblémát megoldottuk, akkor a részproblémák összeállítása után haladjunk „feljebb” a probléma megoldása felé, vagyis redukáljuk (szűkítsük) a problémát.

Ez egy igen jó gondolatmenetnek tűnik, és ezt a megoldást is kellene választanunk. Először megtervezzük a rendszert, amint azt a 3.1. ábrán jól lehet látni, és folyamatosan kapcsoljuk

hozzá a rendszer komponenseit, és a hozzájuk tartozó elvárásainkat.

Ebben a fejezetben a következőképpen fogjuk elképzelni vizsgáztató rendszerünket: ábrák és minimális matematikai eszközök segítségével felvázoljuk elképzeléseinket, majd azt követően kifejtjük bővebben, matematikai eszközök segítségével.

4.2. Az általunk támasztott követelmények és problémáik

Először is tisztáznunk kell, hogy mit szeretnénk és mik az elvárásaink. Ezt leírjuk, majd kifejtjük bővebben.

A vizsgáztatási rendszer három szereplője:

- vizsgázó;
- javító tanár;
- vizsgabiztos.

A közöttük felmerülő probléma pedig a következő: a vizsgázó megírja a dolgozatát és elküldi a rendszeren belüli megfelelő szereplőnek. A tanár javítja és eljuttatja a megfelelő személyhez. A felmerülő probléma az, hogy a küldött dokumentumnak egész idő alatt **változatlannak kell lennie**, valamint a tanár és a vizsgázó **anonim folyamat**nak kell lennie a vizsgázás teljes folyamata alatt. Továbbá, akad még egy probléma. Szükséges egy **hitelesített aláíró kulcspár** is. Ezekre a felmerülő kérdésekre keressük a választ.

4.3. A vizsgázó

4.3.1. A vizsgázó anonimizálása és rendszerben való szerepe

1. A vizsgára való jelentkezéskor a vizsgabiztos ellenőrzi, hogy milyen vizsgán vehet részt.
2. Elkészíti **t**ermészetes **a**zonosítóit és a **v**izsga **a**zonosítóját tartalmazó szót, amelyet szükség esetén kiegészít egy véletlenszámmal: **S:=va#ta#vsz**, ahol # a konkatenáció jele.
3. Ezt követően elkészíti az anonim azonosítóját *Vernam-kódolás* segítségével:

$$\mathbf{S_a} := \mathbf{0^k vsz_1} + \mathbf{S},$$

ahol a 0^k művelet a 0 szám k-szor egymást követő konkatenációja, és hossza megegyezik *va* hosszával, valamint vsz_1 egy véletlenszám, amely *ta#vsz* hosszával egyezik és + az *xor* műveletet jelöli.

4. Jelentkezéskor $\mathbf{S_a}$-t és $\mathbf{vsz_1}$-t külön-külön aláíratja a vizsgabiztossal, így megkapja $\mathbf{S_{av}}$ és $\mathbf{vsz_{1v}}$ szavakat. A $\mathbf{vsz_1}$#$\mathbf{vsz_{1v}}$ szavakat a vizsga végéig tárolja.

5. A vizsgadolgozatra ráírja $\mathbf{S_a}$#$\mathbf{S_{av}}$-t, megírja a dolgozatot, majd digitálisan aláírja és elküldi az illetékes személynek. Itt szeretném megjegyezni, hogy a digtális aláírást egy hitelesített aláíró kulcspár segítségével végzi el. Itt megengedhetjük azt, hogy a dolgozatok egy hitelesítőszerveren keresztül – a vizsgabiztost megkerülve –, közvetlenül az arra illetékes tanárhoz kerüljenek.

6. Amennyiben nem elégedett dolgozatának értékelésével, akkor szintén $\mathbf{S_a}$#$\mathbf{S_{av}}$-t használja felszólalásra.

7. A vizsgafolyamat végén a vizsgabiztos nyilvánosságra hozza $\mathbf{S_a}$#$\mathbf{S_{av}}$-t és a hozzá tartozó jegyet.

8. A vizsgázó elküldi $\mathbf{vsz_1}$#$\mathbf{vsz_{1v}}$-t a vizsgabiztosnak, aki ellenőrzi azt, majd felfedi a vizsgázó személyét és hozzárendeli a kapott jegyet.

Most elemezzük egyenként a lépéseket és vizsgáljuk meg, hogy mit miért kell tennünk a folyamatban.

Az első lépés elég egyszerű. Itt nem más történik, mint a vizsgázó természetes azonosítókkal (személyi igazolvány, vizsgaértesítésről szóló dokumentum, stb.) bebizonyítja, hogy ő az akinek mondja magát, és szeretne vizsgán részt venni. Ezt a vizsgabiztos ellenőrzi, hogy a valóságnak megfelel-e.

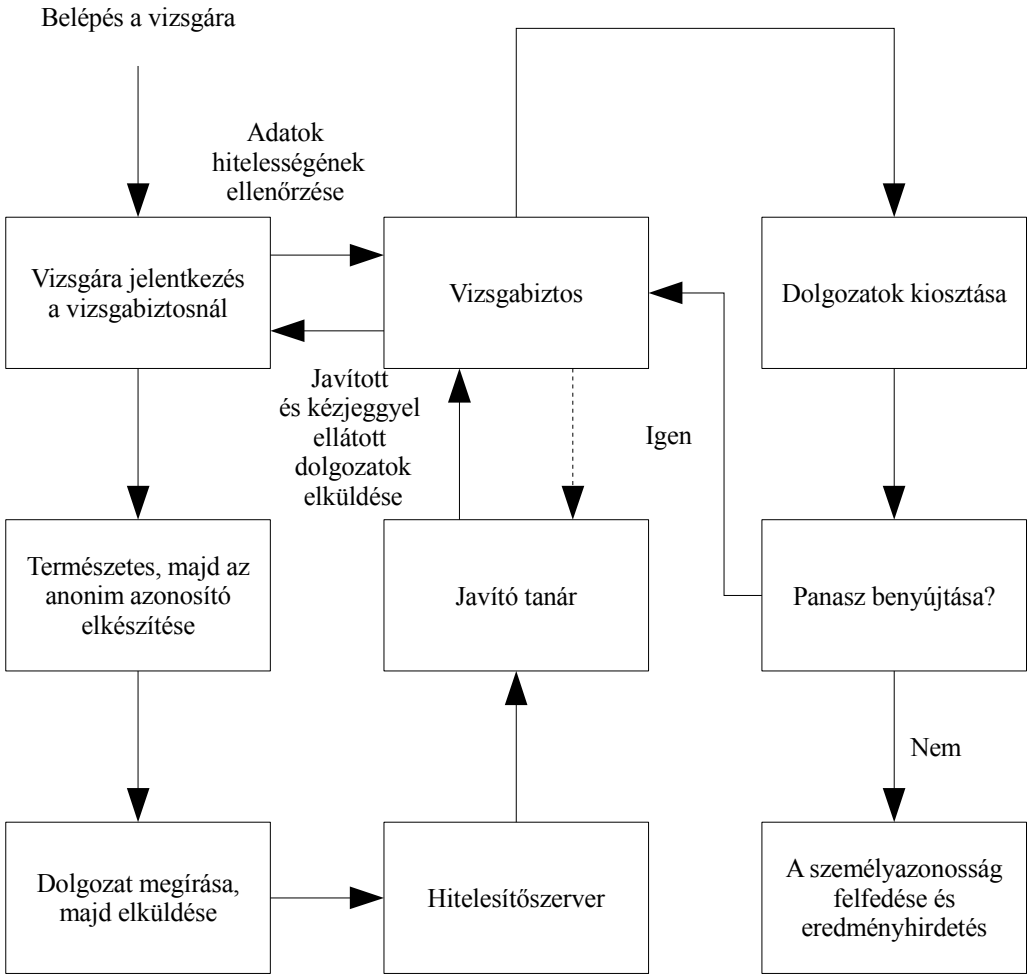
A második lépésben elkészíti az azonosítóját a függelék 55. oldalán leírtak szerint.

A harmadik részben ezt titkosítja Vernam-kódolás segítségével úgy, hogy a vizsgaazonosító **ne** legyen titkosítva. Ezt úgy érhetjük el, hogy azon karakterek helyét, ahol a vizsgaazonosító helyezkedik el, az 0-kal lesz az *xor* műveletnek alávetve. Így a vizsgázó személyazonossága titkosítva lesz, de mégis tudni fogjuk, hogy melyik vizsgára jogosítja fel azonosítója.

A negyedik lépésben az azonosító hitelesítése következik. Ehhez elegendő az S_a-t és vsz_1-et hitelesíttetni (vakon aláírattatni) a vizsgabiztossal, mert ezen adatok ismeretében minden további művelet lehetségessé válik, és a folyamat visszakövethető lesz. Valamint itt szeretném megjegyezni, hogy vsz_1#vsz_{1v}-et a vizsga végéig tárolja, mert ennek segítségével tudja majd a vizsgabiztos ellenőrizni, hogy tényleg ő írta alá vakon vsz_1-et.

Az ötödik lépésben „ráírja lapjára” S_a#S_{av}-t – mely megtehető egy egyszerű konkatenálással is –, majd megírja a dolgozatát, és elküldi javítás-értékelésre. Ekkor azonban felmerülhet egy probléma. Mégpedig az, hogy **bízhatunk-e** abban a személyben aki javítani fogja a dol-

gozatokat. Az általános válasz az, hogy nem[1]. Ezt minden erővel meg kell akadályoznunk. Ezt meg lehet tenni *elektronikus aláírás* segítségével, mely garantálja azt, hogy ha a dolgozat megváltozott – az általunk nem kívánt módon –, azt jelezni fogja. A digitális aláírásról röviden a függelék 56. oldalán olvashatunk.



4.1. ábra. A vizsgázó szerepe a rendszerben

A hatodik lépés azt fejezi ki, ha a vizsgázó nem elégedett a javítással, akkor panaszbenyújtási lehetősége van, kizárólag **egy** alkalommal. Ekkor a vizsgabiztos megkérdezi a tanárt – anonim válaszcsatornán keresztül –, hogy elfogadja-e ezt a felszólalást és dönt a kérdésben.

A hetedik lépésben a vizsgabiztos nyilvánosságra hozza $\mathbf{S_a}\#\mathbf{S_{av}}$-t és a hozzá tartozó jegyet. Ekkor még nem tudjuk, ki kapta ezt az érdemjegyet, csak annyit tudunk, hogy az adott azonosítóval rendelkező ember. Amennyiben nem elégedett a vizsgázó az általa kapott jeggyel, felszólalással élhet. Ehhez az *anonim válaszcsatornát* használhatja.

[1]Gondoljunk csak arra az esetre, hogy a tanár kijavítja, vagy épp ellenkezőleg, elrontja válszainkat.

A nyolcadik lépésben a vizsgázó felfedi személyazonosságát a vizsgabiztos előtt olyan módon, hogy a vizsgabiztos ellenőrzi a vizsgázó által birtokolt véletlenszám és annak hitelesített változatát, hogy tényleg ő általa van-e aláírva. Amennyiben problémát nem talál, ezt követően kiszámítja – S_a és vsz_1 ismeretében – az $S_a + 0^k \# vsz_1$-t és megkapja az S-et, melyből már „természetes úton" kinyerhetőek a vizsgázó adatai, és így a jegy hozzárendelése is lehetővé válik.

Itt azt használtuk fel, hogy a vizsgázó önmagától akarja felfedni személyazonosságát a folyamat végén, és ezért bízhatjuk rá a természetes azonosítóját maszkoló véletlenszám tárolását is. A vizsgázó szerepét a rendszerben a 4.1. ábra mutatja be.

4.3.2. A vizsgázó megközelítése matematikai szempontból

Ebben a részben az előző pont matematikai kifejtése következik. Meg kell állapodnunk abban, hogy a rendszerünkhöz egy külső hitelesítőszerver is tartozik, mely időbélyeggel vagy digitális aláírással látja el a hozzá beérkezett dolgozatokat, továbbá megengedhetjük azt is, hogy ezek lenyomatát tárolja is a későbbi – esetlegesen fennálló – problémák elkerülése végett. Ezt egy hitelesített (e_s, n_s) nyilvános és (d_s, n_s) titkos kulcspár segítségével teszi meg.

Továbbá meg kell állapodnunk abban, hogy a vizsgázó rendelkezik egy (e_v, n_v) nyilvános és (d_v, n_v) titkos, hitelesített aláíró kulcspárral is, valamint MD legyen egy mintavételező függvény. Ekkor vizsgázónk szerepe a rendszerünkben – matematikai megközelítésben – a következő lépésekből fog állni:

1. A vizsgabiztos ellenőrzi a vizsgázó személyazonosságát és jogosultságát a vizsgára való belépéshez.

2. A vizsgázó elkészíti természetes azonosítóját (erre egy lehetséges példa a függelékben, az 55. oldalon olvasható), ami nem más, mint természetes adatainak konkatenációja: **S** := **va**#**ta**#**vsz**, ahol *va* vizsgaazonosító, *ta* természetes azonosítók konkatenációja és *vsz* egy – esetenként szükséges – véletlenszám, amely kitölti a fennmaradó üres helyeket. Mivel az aláíró algoritmus az RSA lesz, így célszerű a természetes azonosító hosszát olyan hosszúra megválasztani, amelyet ez az algoritmus kezelni tud (pl. 1024 bit).

3. Ezt követően a természetes azonosító anonimizálása veszi kezdetét Vernam-kódolás segítségével:

$$S_a = \textbf{va}\#\textbf{ta}\#\textbf{vsz}$$
$$xor$$
$$\textbf{00}\#\textbf{vsz}_\textbf{1}....$$

4. Ekkor előáll S_a, majd ezt és vsz_1-et még hitelesíttetni kell a vizsgabiztossal. A vizsgázó először vakítja S_a-t és vsz_1-t, majd aláíratja a vizsgabiztossal. Megengedhető, hogy a vakításhoz vsz_1-et használjuk. Legyen a vizsgabiztos nyilvános kulcsa (n_{vb}, e_{vb}) és titkos kulcsa d_{vb}. Ekkor az S_{av} előállítása:

   (a) Az S_a vakítása:

   $$c_{v\_temp1} = S_a \cdot vsz_1^{e_{vb}} \bmod n_{vb};$$

   (b) Az S_a hitelesítése:

   $$S_{av\_\text{hitelesített}} = c_{v\_temp1}^{d_{vb}} \bmod n_{vb};$$

   (c) Az S_{av} kinyerése:

   $$S_{av} = \frac{S_{\text{av\_hitelesített}}}{vsz_1} = \frac{c_{v\_temp1}^{d_{vb}}}{vsz_1} = \frac{(S_a \cdot vsz_1^{e_{vb}})^{d_{vb}}}{vsz_1} = \frac{S_a^{d_{vb}} \cdot vsz_1^{e_{vb} \cdot d_{vb}}}{vsz_1} =$$

   $$= \frac{S_a^{d_{vb}} \cdot vsz_1}{vsz_1} = S_a^{d_{vb}} \pmod{n_{vb}}.$$

   Majd vsz_{1v} előállítása:

   (a) A vsz_1 vakítása:

   $$c_{v\_temp2} = vsz_1 \cdot vsz_1^{e_{vb}} \bmod n_{vb};$$

   (b) A vsz_1 hitelesítése:

   $$vsz_{1\_\text{hitelesített}} = c_{v\_temp2}^{d_{vb}} \bmod n_{vb};$$

   (c) A vsz_{1v} kinyerése:

   $$vsz_{1v} = \frac{vsz_{1\_\text{hitelesített}}}{vsz_1} = \frac{c_{v\_temp2}^{d_{vb}}}{vsz_1} = \frac{(vsz_1 \cdot vsz_1^{e_{vb}})^{d_{vb}}}{vsz_1} = \frac{vsz_1^{d_{vb}} \cdot vsz_1^{e_{vb} \cdot d_{vb}}}{vsz_1} =$$

   $$= \frac{vsz_1^{d_{vb}} \cdot vsz_1}{vsz_1} = vsz_1^{d_{vb}} \pmod{n_{vb}}.$$

   Miután elvégezte a fenti műveleteket, a $vsz_1 \# vsz_{1v}$-t magánál tartja a vizsgázás folyamatának végéig.

5. A vizsgadolgozatára „ráírja” (hozzáfűzi) az $S_a \# S_{av}$-t. S_a-ból ellenőrizhető, hogy a megfelelő dolgozatot írta-e meg, továbbá S_a és S_{av} ismeretében pedig ellenőrizni lehet, hogy előírás szerint jelentkezett be a vizsgára. Azonban az azonosító „ráírása” még nem védi

meg egy valódi veszélytől. Ez pedig nem más, mint a dolgozat változatlanságának biztosítása. Ezt digitális aláírás segítségével lehet megvalósítani. Ennek menete a következő:

(a) Legyen a vizsgázó dolgozata m;

(b) Ebből mintát kell vennie: $s = (MD(m))^{d_v} \bmod n_v$, majd a dolgozathoz hozzáfűzi az s-t;

(c) A dolgozat ellenőrizhetőségéhez a következőt kell tennie: $m_1 = s^{e_v} \bmod n_v$, majd $m_2 = MD(m)$. Amennyiben $m_1 = m_2$ akkor a dolgozata nem változott meg, egyébként panasszal élhet, a dolgozta megváltozását illetően.

6. Amennyiben nem elégedett, akkor a felszólalására is $S_a \# S_{av}$-t írja, melyet digitálisan aláír. A felszólalás iterálása nincs engedélyezve, ezt csak egy alkalommal teheti meg a vizsgázó. Ennek oka az, hogy többszörös felszólalás esetén a vizsgabiztos megkérdőjelezheti a javító tanár szakmai hozzáértését. Továbbá, megengedhető az is, hogy közvetlenül kommunikálhasson a vizsgabiztossal egy *anonim válaszcsatorna* segítségével.

7. A vizsgafolyamat végén a vizsgabiztos nyilvánosságra hozza $S_a \# S_{av}$-t és a hozzátartozó jegyet.

8. Az utolsó lépésben a vizsgázó személyének felfedése történik. Először is a vizsgázó elküldi a vizsgabiztosnak a $vsz_1 \# vsz_{1v}$-t, majd a vizsgabiztos „lemásolja” a dolgozaton szereplő $S_a \# S_{av}$-t. Ezt követően S_{av}-ra végrehajtja az $S_a \stackrel{?}{=} S_{av}^{e_{vb}} \bmod n_{vb}$-t, vsz_{1v}-re a $vsz_1 \stackrel{?}{=} vsz_{1v}^{e_{vb}} \bmod n_{vb}$-t, és meggyőződik arról, hogy valóban teljesülnek-e a kívánt egyenlőségek. Amennyiben egyeznek, akkor S_a és vsz_1 ismeretében kiszámítja S-t:

$$S = \mathbf{S_a} \; xor \; \mathbf{00} \# \mathbf{vsz_1}$$

Amennyiben ez megtörtént, akkor a vizsgabiztos visszakapja a vizsgázó S természetes azonosítóját, amelyben az adatai már közvetlenül olvashatóak. Ezt követően a jegy hozzárendelése egyszerűen megtehető. Amennyiben nem tudjuk kiszámítani közvetlenül S_a-t vagy vsz_1-t, akkor feltehetjük, hogy a probléma esetleg a csatorna zajából vagy a vizsgázó esetleges hibájából következett be, és kérhetjük akár az egész vizsgázási folyamat újrakezdését is.

Ezekből a lépésekből áll a vizsgázó matematikai megközelítése, és jól kitűnik, hogy fel van használva az érdekeltsége abban, hogy az eljárás végén felfedje személyazonosságát. Azonban

vannak esetek, mikor a vizsgázó nem akarja felfedni személyazonosságát (a mi esetünkben nem akarja elküldeni $vsz_1 \# vsz_{1v}$-t a vizsgabiztosnak). Az esetek nagy részében az ok, hogy a vizsga sikertelen volt. Mi viszont szögezzük le azt, hogy aki nem fedi fel személyazonosságát, az automatikusan elégtelen osztályzatot kap.

4.3.3. A panasz benyújtása

Tisztáznunk kell még egy fontos kérdést. Mit kell tennünk abban az esetben, ha a vizsgázó panaszt nyújt be a kapott érdemjegyét illetően? Erről csak annyi szót ejtettünk, hogy felszólalására ráírja az $S_a \# S_{av}$-t, ezt követően digitálisan aláírja és eljuttatja a megfelelő személynek.

Mikor a dolgoztatokat a javító tanár értékelte, elküldi a vizsgabiztoshoz, aki eredményt hirdet a rajta lévő azonosító alapján. Ekkor a vizsgázó már tudja az eredményét, de személye még ismeretlen. Amennyiben panasszal szeretne élni, akkor az anonim válaszcsatorna segítségével írhat a vizsgabiztosnak, aki megkérdezi a javító tanárt – ugyanezen a módon –, hogy elfogadja-e a felszólalást. Amennyiben a tanár elfogadja a felszólalást, akkor a vizsgabiztos dolga, hogy döntsön a kérdésben.

A vizsgázónak azonban csak egyszer van ilyen lehetősége, így nem alakulhat ki körfolyamat a rendszeren belül (amit szeretnénk is elkerülni). Amennyiben több panasz érkezik a dolgozat javítását-értékelését illetően, akkor a vizsgabiztos megkérdőjelezheti a javító tanár szakmai hozzáértését, majd felfedheti személyazonosságát és kizárhatja a rendszerből.

4.4. A javító tanár

4.4.1. A javító tanár anonimizálása és rendszerben való szerepe

A javító tanár szempontjából kissé másképp kell eljárnunk, mint a vizsgázó esetében. A legfőbb különbség az, hogy *személye ismeretlen a vizsgázás teljes folymata alatt, valamint azt követően is*. Csak abban az esetben kell személyazonosságát felfedni, ha a vizsgabiztos visszaélést tapasztal, vagy megállítja a tanár szakmai inkopetenciáját.

Ebből következik, hogy nem érdekelt személyazonosságának felfedésében, tehát nem lehet rá bízni az anonimitását biztosító maszk tárolását. Ehhez szükséges egy teljesen megbízható *felügyelő*. Ekkor a javító tanár és szerepe a rendszerben a következő:

1. A tanár dokumentumokkal bizonyítja szakmai kompetenciáját, majd igazolja magát természetes azonosítói segítségével.

2. Ezt követően a felügyelő elkészíti a **ko**mpetenciát és a **t**ermészetes **a**zonosítóját tartalmazó szót, melyet szükség esetén kiegészít egy véletlenszámmal: **T** := **ko**#**ta**#**vsz**, ahol # a konkatenáció jele.

3. A felügyelő ezt követően elkészíti anonim azonosítóját:

$$\mathbf{T_a} := \mathbf{0^k}\#\mathbf{vsz_1} + \mathbf{T},$$

   ahol a 0^k művelet a 0 szám k-szor egymást követő konkatenációja és hossza megegyezik *ko* hosszával, valamint vsz_1 egy véletlenszám, amely *ta#vsz* hosszával egyezik és + az *xor* műveletet jelöli. Ezt követően vsz_1-et szétosztja a vizsgabiztosok között, majd aláíratja T_a-t a vizsgabiztossal, így megkapja T_{av}-t. Ezt követően átadja $T\#T_a\#T_{av}$-t a tanárnak és törli vsz_1-et.

4. Ettől fogva $T_a\#T_{av}$ lesz a tanár aláírt és hitelesített anonim azonosítója. Ezekből az adatokból csak a biztos(ok) – vagy külön erre a célre létrehozott csoport(ok) – képes(ek) visszaállítani a tanár személyazonosságát.

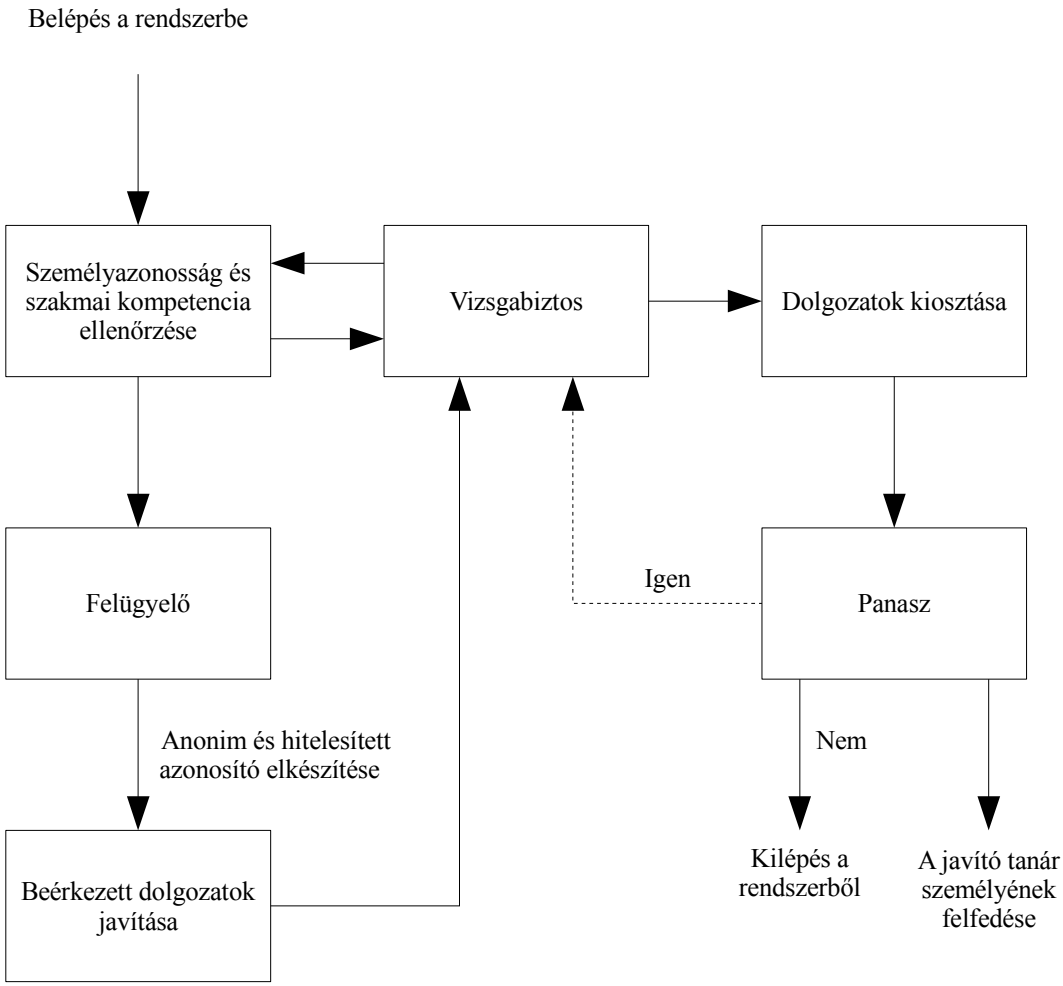
Ezek után vizsgáljuk egyenként a folyamat lépéseit, kitérve a fenti pontokból kimaradt esetleges részletekre is.

Az első lépés elég egyszerű. Itt nem más történik, mint a javító tanár természetes aznosítókkal (személyi igazolvány, útlevél, stb.) bebizonyítja, hogy ő az akinek mondja magát, továbbá szakmai hozzáértését. Ezt a vizsgabiztos ellenőrzi, hogy a valóságnak megfelel-e.

A második lépésben a felügyelő elkészíti természetes azonosítóját. Erre egy példát a függelék 56. oldalán láthatunk.

A harmadik lépésben a felügyelő elkészít számára egy anonim, vizsgabiztos által hitelesített azonosítót *Vernam-kódolás* segítségével, melyet a vizsgázás folyamatában használ. Ebben a lépésben két fontos dolgot kell megemlítenünk. Először is a vsz_1 véletlenszámot **szét kell osztani** a vizsgabiztosok között, majd törölni kell. Amennyiben nem törli, fennáll a felügyelő részéről a visszaélés lehetősége. A szétosztást meg lehet valósítani például Shamir-titokmegosztó rendszerének segítségével, melyről a 2. fejezetben olvashatunk.

A negyedik pontban előáll a tanár azonosítója, melyet a dolgozatok aláírására, és az esetleges panaszbenyújtások esetén, személyazonosságának felfedésére használnak. Ezzel csak ő rendelkezik, és ebből önmaga nem képes visszaállítani személyazonosságát. Ehhez a vizsgabiztos(ok)ra – vagy külön erre a célra létrehozott csoport(ok)ra – is szüksége van. A tanár szerepét a vizsgáztatás folyamatában a 4.2. ábra szemlélteti. Az ábrából jól kitűnik, hogy a rendszerben való szerepe nem olyan körülményes, mint a vizsgázó esetében. Először igazolja magát, majd szakmai kompetenciáját. Ezt követően kap egy azonosítót, mellyel a rendszerben



4.2. ábra. A javító tanár szerepe a rendszerben

a rá kiszabott feladatok elvégzését tudja megvalósítani. Ezt követően kijavítja a dolgozatokat, ráírja azonosítóját és elküldi a vizsgabiztoshoz.

A javító tanár esetében azonban kétféleképpen lehetséges a rendszer elhagyása. Az első esetben szabályosan megy végbe, ha nem érkezik egynél több felszólalás a dolgozat javítását-értékelését illetően a vizsgázó részéről a vizsgabiztos felé. Ekkor a biztos megkérdezi a javító tanárt, hogy elfogadja-e a felszólalást. Amennyiben elfogadja, akkor a dolgozatot felül kell vizsgálnia és újra kell értékelnie. Ha nem érkezik panasz, akkor elhagyja a rendszert.

Amennyiben egynél több felszólalás érkezik, akkor a vizsgabiztos(ok) feltételezhetik a tanár szakmai inkompetenciáját, felfedhetik személyazonosságát és kizárhatják a rendszerből. Természetesen, ki kell zárni a „farkast kiált” eseteket, mert ekkor a problémát már lehet, hogy a vizsgázó személyében kell keresnünk.

Meg kell említeni azt, hogy a tanár a rá kiszabott feladat ellátását követően digitálisan aláírja dolgozatot, és ezt hozzáfűzi a dolgozathoz. Erre egy esetleges vizsgálat vagy panaszbenyújtás esetén lehet szüksége, hogy be tudja bizonyítani azt, hogy eljárása szabályszerű volt.

4.4.2. A javító tanár megközelítése matematikai szempontból

Ebben a részben az előző pont matematikai kifejtése következik. Meg kell állapodnunk abban, hogy a rendszerünkhöz egy felügyelő is tartozik – mely lehet akár egy szerver is –, továbbá a tanár rendelkezik egy (e_t, n_t) nyilvános és (d_t, n_t) titkos, hitelesített aláíró kulcspárral is, a dolgozatok digitális aláírásához.

A javító tanár szerepe rendszerünkben – matematikai megközelítésben – a következő lépésekből fog állni:

1. A tanár dokumentumokkal bizonyítja szakmai hozzáértését, majd igazolja magát természetes azonosítói segítségével.

2. A felügyelő elkészíti a természetes azonosítóját (erre egy lehetséges példa a függelékben, az 56. oldalon olvasható), ami nem más, mint természetes adatainak konkatenációja: **T** := **ko#ta#vsz**, ahol *ko* kompetencia, *ta* természetes azonosítók konkatenációja és *vsz* egy – esetenként szükséges – véletlenszám, amely kitölti a fennmaradó üres helyeket. Mivel az aláíró algoritmus az RSA lesz, így célszerű a természetes azonosító hosszát olyan hosszúra megválasztani, amelyet ez az algoritmus kezelni tud (pl. 1024 bit).

3. Ezt követően a felügyelő választ egy vsz_1 véletlenszámot, melynek hossza megegyezik *ta#vsz* hosszával, majd Vernam-kódolás segítségével kiszámítja:

$$T_a = \textbf{ko\#ta\#vsz}$$
$$xor$$
$$\textbf{00\#vsz}_\textbf{1}....$$

   Miután az anonosító titkosítva van, vsz_1 -et szét kell osztania a vizsgabiztos(ok) között. Ezt a következőképpen teszi meg: legyen N vizsgabiztos és legyen $t + 1$ vsz_1 részeinek száma. Választ $t + 1$ darab véletlenszámot: $a_1, \ldots, a_{t+1}$ ($a_i \in \mathbb{N}$), melyek segítségével megkonstruál egy véletlen f polinomot. Ezek a véletlenszámok akár eshetnek egy véletlenszerűen kiválasztott intervallumba is. Ekkor a polinom alakja a következő lesz:

$$f(x) = vsz_1 + a_1x^1 + a_2x^2 + \ldots + a_{t+1}x^{t+1},$$

   ahol $x \in \mathbb{R}$.

Ebből jól látszik, hogy $f(0) = vsz_1$ fog teljesülni. Ezt követően x_i értékekhez kiszámítja a megfelelő $f(x_i) = y_i$ értékeket ($i = 1, \ldots, t+1$), és ezen (x_i, y_i) értékpárokat osztja szét a biztosok között.

Ha befejezte a véletlenszám szétosztását, akkor még a T_a-t alá kell íratnia a vizsgabiztossal. Ezt a következő eljárással tudja megtenni, ahol (e_{vb}, n_{vb}) a vizsgabiztos nyilvános, és (d_{vb}, n_{vb}) a titkos kulcsa:

(a) T_a-t vakítja: (például vsz_1 segítségével)

$$c_{\text{vakított}} = T_a \cdot vsz_1^{e_{vb}} \bmod n_{vb};$$

(b) T_a hitelesítése:

$$T_{a\_\text{hitelesített}} = c_{\text{vakított}}^{d_{vb}} \bmod n_{vb};$$

(c) T_{av} kinyerése:

$$T_{av} = \frac{T_{a\_\text{hitelesített}}}{vsz_1} = \frac{c_{\text{vakított}}}{vsz_1} = \frac{(T_a \cdot vsz_1^{e_{vb}})^{d_{vb}}}{vsz_1} = \frac{T_a^{d_{vb}} \cdot vsz_1^{e_{vb} \cdot d_{vb}}}{vsz_1} =$$

$$= \frac{T_a^{d_{vb}} \cdot vsz_1}{vsz_1} = T_a^{d_{vb}} \pmod{n_{vb}}.$$

Ezt követően átadja $\mathbf{T\#T_a\#T_{av}}$-t a javító tanárnak, és törli vsz_1-et.

4. Ezt követően a tanár azonosítója $\mathbf{T_a\#T_{av}}$ lesz. Ezekből az adatokból csak a vizsgabiztos(ok) – vagy külön erre a célra létrehozott csoport(ok) – képes(ek) visszaállítani a tanár természetes azonosítóját.

4.4.3. A javító tanár személyének felfedése

A javító tanár esetében a probléma, személyazonosságának elfedése. Ez – mint láttuk az előző pontban –, csak a vsz_1 értékétől és a felügyelő megbízhatóságától kell függővé tennünk, akiről feltettük, hogy törli az értéket a szétosztás és vakítás után. Mint ismeretes, csak akkor lehet felfedni személyazonosságát, ha egynél több panasz érkezik egy általa javított dolgozat esetében. Ebben a pontban azt fogjuk vizsgálni, hogy mikor és hogyan lehet felfedni egy tanár személyazonosságát.

Az első dolog, amit meg kell említenünk – az előző pontot figyelembe véve –, hogy több vizsgabiztos között osztjuk szét vsz_1-et. Azonban, a mi rendszerünk azt is megengedi, hogy csak egy vizsgabiztos legyen. Ebből nyilvánvalóan az következik, hogy egy javító tanár személyazo-

nossága, csak egy személytől függ. Ezen a ponton el kell gondolkodnunk, hogy célszerű-e egy személy „kezébe adni” egy vagy több ember „életét”. Itt elegendő csak arra gondolnunk, hogy a vizsgabiztos és a javító tanár nézeteltérésbe kerülnek, és egymás „ellenlábasaivá” válnak.

Ezt a helyzetet tudnunk kell kerülni. Az egyik feloldási módja az lehet, hogy maximálisan megbízunk a vizsgabiztosban és rábízzuk vsz_1 tárolását, minden körülmények között. A másik lehetséges módja az, hogy a polinom helyreállításához szükséges adatokat, az erre szakosodott szervezeteknek vagy csoportoknak küldjük el[2]. Azonban, felmerülhet még egy ötlet erre vonatkozólag: hozzunk létre mi magunk „virtuális vizsgabiztosokat”, akiknek az lenne a feladata, hogy a polinom helyreállításához szükséges adatokat biztonságban tároljanak.

A következő probléma a felügyelő megbízhatósága. A felügyelő lehet egy valós személy is, de lehet egy szerver is. Mindenféleképpen érdemes egy szervert választanunk felügyelőnek – ezt meg is tettük –, mert így az emberi beavatkozást minimalizálhatjuk ezen a ponton, és ez a döntés *egyszerűsíti* rendszerünket. Ebben az esetben egy *algoritmust* kell feltételezni, melynek működése *véges* és *determinisztikus*, valamint – a teljes megbízhatóság kedvéért – elláthatjuk akár egy naplózó folyamattal is.

Ezt követően már csak egy problémánk van: *Hogyan lehet felfedni egy tanár személyazonosságát?* A válasz egyszerű: vissza kell állítanunk a polinomot (például *Lagrange-interpoláció* segítségével), és vennünk kell a 0 pontban a függvényértékét. Ebből megkaphatjuk a vsz_1-et és a javító tanár $T_a \# T_{av}$ azonosítójának felhasználásával – melyet egy problémás dolgozatról szerzünk be – felfedjük személyazonosságát, majd kizárjuk rendszerből. Természetesen, egy vizsgabiztos esetén az eljárás egyszerűsödik.

Tegyük fel, hogy rendszerünkben $t+1$ vizsgabiztos van és N részre osztottuk vsz_1-et. Ekkor a polinom konstrukciója a következő:

1. A vizsgabiztosok gyűjtsék össze az összes szétosztott $(x_i, f(x_i))$ párt.

2. Ezt követően meg kell határozni a polinomot. Már ejtettünk róla szót, hogy erre a *Lagrange-interpolációt* kell használni, melynek alakja a következő:

$$L_{t+1}(x) = \sum_{i=1}^{t+1} f(x_i) \sum_{\substack{j=0 \\ j \neq i}}^{t+1} \frac{x - x_j}{x_i - x_j},$$

   melynek segítségével a keresett polinom kiszámítható.

3. Amennyiben a polinom előállt, azt követően venni kell a 0 helyen a helyettesítési értékét, melyből következik vsz_1.

[2]Ez a körülmény feltehetőleg csak még bonyolultabbá tenné rendszerünket.

4. Ezt követően vennünk kell a tanár $T_a \# T_{av}$ azonosítóját, majd ellenőriznünk kell, hogy T_{av}-ból számítható-e T_a (vagyis $T_a =^? T_{av}^{e_{vb}} \bmod n_{vb}$). Ezt biztos(ok) ellenőrzi(k) le, majd a következő művelet segítségével felfedi(k) a javító tanár személyét:

$$T = \mathbf{T_a}$$
$$xor$$
$$\mathbf{00\#vsz_1}$$

Ezt követően még mindig felmerülhet egy érdekes eset. Ennek oka, hogy a tanár továbbra is titokban szeretné tartani személyét, és egy ilyen esetben megpróbálhatja „megvédeni magát”. Képzeljük el azt, hogy mi történne akkor, ha a tanár – tudatában a lehetséges következményeknek – egyszerűen „beleír” a kapott azonosítójába, és a felismerhetetlenségig eltorzítja úgy, hogy T_{av}-ból ne lehessen T_a-t számítani. Ekkor a helyzetünk nagyon egyszerű, hiszen egy kizárásos módszerrel azonosítani tudjuk őt a rendszerben, majd ezt követően kizárhatjuk.

Itt szeretném megemlíteni, hogy a tanárt csak akkor célszerű kizárni a renszerből, ha több alkalommal való kizárás vagy panasszal való élés történt. Ennek oka lehet például az, hogy a tanár részére hibás megoldókulcsot küldtek, vagy egy feladatot „tömegesen félreértelmeztek”.

Végezetül ne feledkezzünk meg arról sem, hogy egy javító tanár lehet akár egy vizsgabiztos is, és egy vizsgabiztos is lehet javító tanár. Amennyiben ilyen eset fordulna elő, akkor valamelyik „kiváltságáról” ideiglenesen le kell mondania – a vizsga időtartamát illetően –, viselve az ezzel járó jogokat és kötelességeket is.

4.5. A vizsgabiztos és a dolgozatok szétosztása

4.5.1. A probléma megközelítése elméleti szempontból

Ez a szekció a legfontosabb részét képzi vizsgáztató rendszerünknek. Éppen ezért – a bonyolult modellek elkerülése érdekében –, célszerű ezt a fenti két feladatot együttesen kezelni. Először célszerű felvázolni elképzeléseinket, majd részletesebben kifejteni azt.

Mint azt korábban említettük, a vizsgabiztos személye a vizsgáztatási folyamat során végig ismert. Amennyiben több biztos is van, akkor célszerű lenne, ha nem tudná sem a vizsgázó, sem a javító tanár, hogy ki gondozza dolgozatukat, valamint azt, hogy kitől kapta. Éppen ezért célszerű lenne – az emberi beavatkozás minimalizálása céljából –, ha a dolgozatok kiosztására nem is lenne hatással, hanem azt egy „gépezetre” bíznánk. Ezt elérhetjük egy hitelesítőszerver[3]

[3]Erről szóltunk a 4.3.2. szekcióban is.

beiktatásával, mely hitelesített aláíró kulcspárral rendelkezik, és joga van a dolgozat digitális aláírásához. Természetesen, a szerver *teljesen megbízható* eleme a vizsgáztató rendszerünknek.

Itt kell megemlítenünk újra, hogy a vizsgabiztosok között lehetnek javító tanárok is. Ebben az esetben, egy – kellően szélsőséges – ellentmondást állapíthatunk meg az előző fejezetben leírtakkal szemben. Ez pedig nem más, mint a javító tanár személyazonosságának elfedése. Amennyiben ezt összevetjük az 1.2. ponbtan leírtakkal, akkor láthatjuk, hogy ez csak akkor lehetséges, ha javító tanár(ok) és a vizsgabiztos(ok) halmaza *diszjunkt*. Ezen a módon viszont feloldható ez a látszólagos ellentmondásosság.

Végezetül meg kell állapondunk abban, hogy a vizsgabiztos rendelkezik egy (e_{vb}, n_{vb}) nyilvános, és (d_{vb}, n_{vb}) titkos *hitelesített aláíró kulcspárral*. Ennek a hitelesítési folyamatával azonban nem áll módunkban foglalkozni, mert ez valószínűleg meghaladná ezen fejezet kereteit.

Most lássuk vázlatosan a folyamatot:

1. A dolgozatok beérkeznek a hitelesítőszerverhez, ahol megállapításra kerül érvényességük, majd tárgyuk.

2. Ezt követően, – a szerver aláírásával együtt – a megfelelő tanárhoz kerülnek.

3. A tanár – javítást követően –, anonim azonosítóját, javítását és digitális aláírását hozzáfűzi a dolgozathoz, majd elküldi a vizsgabiztoshoz.

4. A vizsgabiztos ellenőrzi a tanár és a diák azonosítóját, valamint a dolgozat változatlanságát.

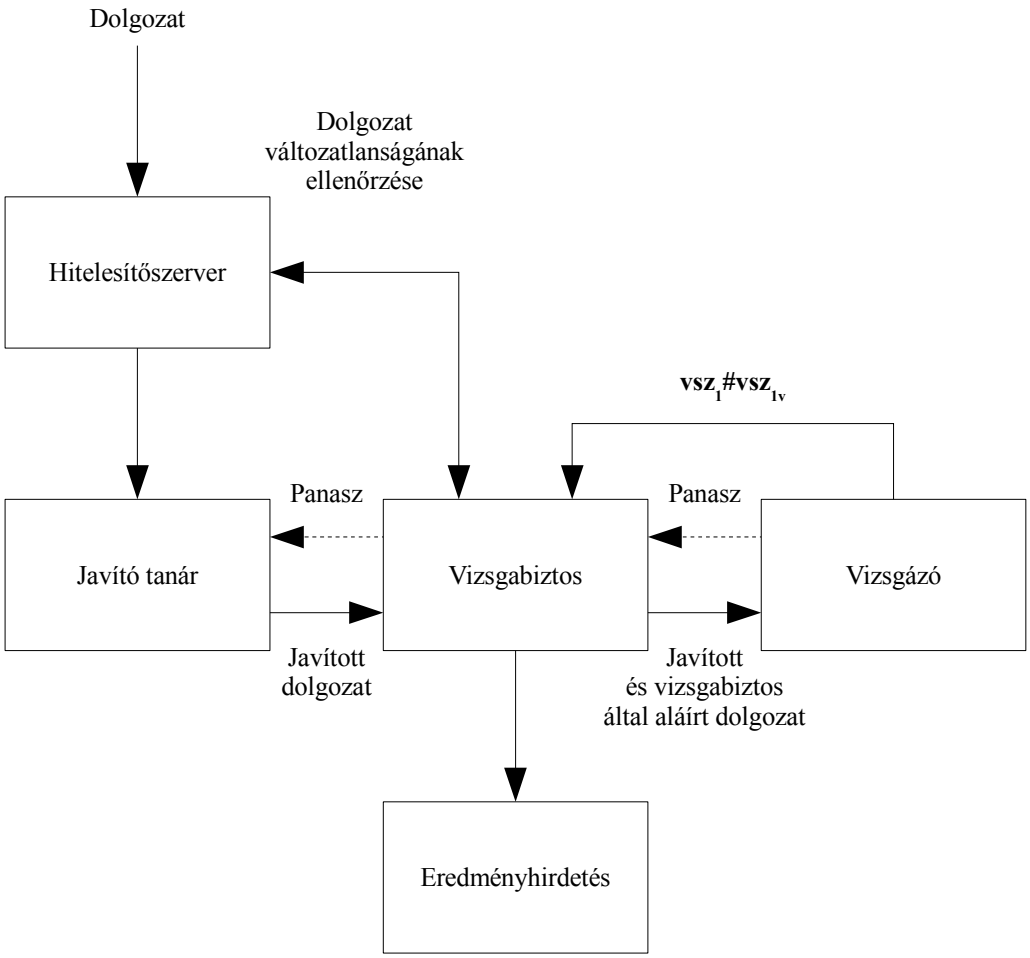
5. Ezt követően eltávolítja a tanár azonosítóját, aláírja a dolgozatot és elküldi a vizsgázónak.

6. Amennyiben a vizsgázó elfogadja a javítást, elküldi $vsz_1 \# vsz_{1v}$-t a vizsgabiztosnak, aki megállapítja személyazonosságát és rögzíti a vizsgajegyet.

7. Ellenkező esetben megfogalmazza észrevételeit és aláírva elküldi azt a vizsgabiztosnak.

8. A vizsgabiztos aláírva továbbítja a dolgozatot, a javítást és az észrevételeket a tanárnak, aki dönt az észrevételekkel kapcsolatban, majd eljuttatja ezt – aláírásával együtt – a vizsgabiztoshoz, aki felülvizsgálja ezt, és közli a vizsgázóval.

9. Amennyiben a vizsgázó elfogadja az eredményt, akkor vissza **6.**-ra, egyébként közvetlenül folytatja a tárgyalást a vizsgabiztossal.



4.3. ábra. A vizsgabiztos szerepe a rendszerben

Ezt a folyamatot a 4.3. ábra mutatja be. Mint az ábrából jól kitűnik, középpontjában a vizsgabiztos áll, mint folyamatot menedzselő személy. Továbbá jól kivehető az is, hogy a vizsgáztató rendszer személye köré épül, vagyis tőle függ a folyamat zavartalansága, más szavakkal a *teljes felelősség* a vizsga lebonyolításáért. A következő alfejezetben részletesen ki fogjuk fejteni a fenti pontokat, a matematika eszközeit használva.

4.5.2. A probléma megközelítése gyakorlati szempontból

4.5.2.1. A dolgozat beérkezése

Miután a vizsgázó megírta dolgozatát, „ráírta” azonosítóját és digitálisan aláírta, elküldi a javító tanár részére javítás-értékelésre. Ez a dolgozat közvetlenül a hitelesítőszerverhez fog megérkezni. Ezt követően a szerver lemásolja a vizsgázó $S_a \# S_{av}$ azonosítóját, és ellenőrzi azt, hogy ezzel az azonosítóval adott-e már be valaki dolgozatot. Amennyiben már adott be, akkor a

dolgozatot dobjuk el (így megelőzve a dolgozatok duplikált javításának esetét). Ellenkező esetben ellenőrzi azt, hogy S_{av}-ből számítható-e S_a. Ezt úgy teszi meg, hogy ellenőrzi a következő egyenlőség teljesülését:

$$S_a =^{?} S_{av}^{e_{vb}} \bmod n_{vb} = (S_a^{d_{vb}})^{e_{vb}} \bmod n_{vb} = S_a.$$

Amennyiben a fenti egyenlőség teljesül, akkor a szerver „biztos” lehet abban, hogy a vak aláírás korrekt, és az a vizsgázó küldte, aki arra jogosult. Ezt követően megvizsgálja, hogy milyen vizsgáról érkezett a dolgozat. Ezt úgy teszi meg, hogy veszi S_a-ból *va* karaktereinek megfelelő hosszt, és azon a szakaszon *xor* műveletet hajt végre.

4.5.2.2. A dolgozat elküldése

Miután a szerver kiszámította az „irányítószámot”, azt követően a dolgozatot digitálisan aláírja, és elküldi a megfelelő javító tanárhoz. Az aláírásra azért van szükség, hogy bizonyítani lehessen a dolgozat változatlanságát egy esetleges panasz, vagy más felmerülő probléma esetén. Erről már szóltunk a 4.3.2. szekció kapcsán. Ezt követően digitálisan alá kell írnia a vizsgázó dolgozatát. Ennek folyamata a következő lesz:

1. Legyen a vizsgázó dolgozata m;
2. A szerver mintát vesz: $s = (MD(m))^{d_s} \bmod n_s$, majd a dolgozathoz hozzáfűzi az s-t;
3. A dolgozat ellenőrizhetőségéhez a következőt kell tennünk: $m_1 = s^{e_s} \bmod n_s$, majd $m_2 = MD(m)$. Amennyiben $m_1 = m_2$, akkor biztosak lehetünk benne, hogy a dolgozat nem változott meg; ellenkező esetben, a dolgozatot el kell dobnunk, és a megfelelő jelzéssel közölni ezt a rendszer felé.

Abban az esetben, ha csak egy tanár rendelkezik akkreditációval egy tárgy javítását illetően, akkor a szerver csak neki tudja elküldeni a dolgozatot. Ellenkező esetben, a javító tanár kiválasztására egy lehetőség az, hogy annak küldi el, aki legrégebben vár dolgozatra. Ez jó megoldásnak tűnhet, viszont célszerűbb – a visszaélések csökkentése érdekében – annak küldeni, aki jogosult rá, és ezt a szerver *véletlenszerűen* választja ki közülük.

Fontos megjegyeznünk, hogy csak azon tanárokhoz küldhető – ezekben az esetekben – dolgozat, akiknél a legkevesebb van, és ha ezek száma egy határ alá csökken, akkor ezen személyek mellé kell vennünk azon személyeket is, akikél eggyel több dolgozat van. Ezt a döntési algoritmust figyelembe véve kell a szervernek elküldeni a dolgozatot, az arra illetékes javító tanárhoz.

4.5.2.3. A dolgozat javítása

Mikor a szerver elkülde a dolgozatot az illetékes tanárnak, a javító tanár javítja, majd értékeli a dolgozatot. Ezt követően hozzáfűzi $T_a \# T_{av}$ azonosítóját a dolgozathoz a javítással együtt, digitálisan aláírja, majd ezt elküldi a vizsgabiztosnak.

4.5.2.4. Az eredmények rögzítése

A dolgozat beérkezését követően a vizsgabiztos ellenőrzi a tanár és a vizsgázó azonosítóját. Ezt úgy teszi meg, hogy veszi $S_a \# S_{av}$-t és $T_a \# T_{av}$-t a dolgozatról, majd végrehajtja rá az $S_a =^? S_{av}^{e_{vb}} \bmod n_{vb}$-t és a $T_a =^? T_{av}^{e_{vb}} \bmod n_{vb}$-t. Amennyiben az egyenlőségek igazak, akkor ellenőrzi a dolgozat változatlanságát a 4.5.2.2. pontban közöltek szerint. Amennyiben mindent rendben talált, akkor feljegyzi a tanár és a diák azonosítóját, valamint az eredményt. Ellenkező esetben akár el is dobhatja a dolgozatot, vagy jelzést adhat a rendszer irányába a felmerült problémát illetően.

4.5.2.5. A dolgozat eljuttatása a vizsgázóhoz

Ezt követően a dolgozatról eltávolítja a tanár aláírását. Ekkor a dolgozat tartalmazni fogja a vizsgázó dolgozatát, aláírásával együtt, valamint egy lenyomatot a szervertől. Ezt a vizsgbiztos digitális aláírás segítségével hitelesíti. Erre a visszaélések esélyének csökkentése miatt van szükség, valamint így biztosak lehetünk abban, hogy ezt a dolgozatot a vizsgabiztostól közvetlenül kaptuk meg.

4.5.2.6. A vizsgázó

Miután a vizsgázó „kézhez kapta” dolgozatának javítását és értékelését, eldönti, hogy egyetért-e a kapott vizsgajeggyel. Amennyiben egyetért, elküldi a vizsgabiztosnak $vsz_1 \# vsz_{1v}$-t, aki meg tudja állapítani személyazonosságát és rögzíti a vizsgajegyét. Erről a 4.3.2. szekció 8. pontjában olvashatunk.

4.5.2.7. A panasz benyújtása

Amennyiben a vizsgázó nem elégedett dolgozatának javítás-értékelésével, akkor felszólalással élhet. Ezt úgy tudja megtenni, hogy megfogalmazza észrevételeit és aláírva elküldi a vizsgabiztosnak.

4.5.2.8. A vizsgabiztos intézkedése

Miután a vizsgabiztos kézhez kapta a felszólalást, továbbítja azt – a vizsgadolgozattal együtt – a javító tanár részére. Ezt ismételt aláírás segítségével teszi meg. Ezt követően a javító tanárnak nyilatkoznia kell arról, hogy mely észrevételeket fogadja el, és módosítja-e az eredményt. Miután a javító tanár felülvizsgálta a javítást, aláírva elküldi ezt a vizsgabiztos részére, aki dönt az eredményről és közli azt a vizsgázóval.

4.5.2.9. A vizsgázás folyamatának vége

Miután a felülvizsgálati eredmény megérkezett a vizsgabiztoshoz, ezt közli a vizsgázóval. Amennyiben a vizsgázó elfogadja ezt a döntést, akkor 4.5.2.6. pontban leírtak szerint folytatódik a vizsgázási folyamat. Ha nem fogadja el ezt a döntést, akkor kövzetlenül folytatja a tárgyalást a vizsgabiztossal.

5. fejezet

A vizsgáztatórendszer egy lehetséges gyakorlati megközelítése

5.1. Pár szó a gyakorlati alkalmazásról

Ebben a fejezetben kísérletet teszünk egy, a gyakorlatban is jól használható vizsgáztatórendszer megkonstruálására. Ennek a rendszernek eleget kell tennie az 1.4. szekcióban felsorolt követelményeknek, ezen követelményeket pedig hosszan fejtettük ki a 4. fejezetben.

Azonban, az elméleti modell sajnos nem minden esetben van összhangban a gyakorlati megvalósíthatósággal. Ez röviden azt fejezi ki, hogy az elméleti megközelítésben gyakran nem mindig vesszük figyelembe a gyakorlati megvalósítás folyamán felbukkanó **fizikai korlátokat**.[1]

Egy rendszer akkor van **jól megtervezve**, ha mind az elméleti, mind a gyakorlati megközelítés teljes **összhangban** áll egymással. Amennyiben ez nem teljesen lehetséges, akkor is létezik elfogadható megoldás. Ebben az esetben az elméleti és gyakorlati megvalósítás között **kompromisszum**ot kell kötnünk. Ez azt fejezi ki, hogy az elméleti modell szűkítésével a gyakorlati modell megvalósíthatóságát segítjük elő, a megfogalmazott követelményeket figyelembe véve.

Ezt a pár gondolatot figyelembe véve kell megvalósítanunk vizsgáztatórendszerünket is. Itt is felbukkanhatnak fizikai korlátok. Elegendő csak arra gondolnunk, hogy a kulcsméretet az elméleti modellben kellően nagyra választva, azonnal beleütközünk a számítási idő nagyságának fizikai megnyilvánulásával.

[1] Gondoljunk csak a Turing-gépre, ahol elméletben végtelen memória és szalag áll rendelkezésünkre, azonban a gyakorlatban ez nem kivitelezhető.

5.2. Egy lehetséges megvalósítás

A vizsgáztatórendszer egy lehetséges megoldását láthatjuk a 62. oldaltól. Először vegyük szemügyre a rendszert alkotó állományokat, tárjuk fel funkcióikat, majd megismerve őket vizsgáljuk meg a teljes rendszer működését, összevetve az elméleti modellel. A program egy vizsgabiztos, egy vizsgázó, egy felügyelő és egy javító tanár esetére mutatja be a rendszer működését.

1. diplomamunka.cpp

Ez az állomány alkotja a rendszer egészét. Irányítja és menedzseli a vizsgázó, a vizsgabiztos, a felügyelő és javító tanár viselkedését.

2. sha1.h

Ebben az állományban egy lenyomatkészítő osztály található, melyet a dolgozat digitális aláírásához használ a vizsgázó, javító tanár és a vizsgabiztos. Egyetlen attribútuma van, ez pedig nem más, mint egy lenyomat egy adott dolgozatról. Az állomány a [7] címről származik.

3. globalis.h

Ez az állomány **statikus** globális változókat, és mindenki által hívható **statikus** metódusokat tartalmaz. A tervezésnél azért volt célszerű statikus osztályba sorolni ezeket a programelemeket, hogy csak egyszer foglaljanak memóriát, és ne terhelje a processzort példányosítással. Többek között itt található a Vernam-kódolás, az adatokat kiolvasó és természetes azonosítót elkészítő metódus, valamint az aláírások hitelességét ellenőrző metódus is. Továbbá, itt kapott helyet a természetes azonosító hosszát és a kulcsméretet beállító globális konstans is.

4. rsa.h

Ez az állomány tartalmazza a kulcsgenerátort. Egyetlen paramétere a kívánt kulcsméret. Ezt megadva, előállítja a kívánt kulcsokat, majd a megfelelő exportáló metódusok meghívását követően exportálja számunkra azokat. Az állomány a [8] címről származik.

5. Tablazat.h

Ebben az állományban egy olyan osztály található, mely konténerobjektumokat állít elő. Szerepük akkor lesz a rendszerben, mikor a vizsgabiztos feljegyzi a tanár és a vizsgázó azonosítóját, valamint a kapott érdemjegyet. Bár attribútumai nyilvánosak, megfelelően alkalmazva „titkossá tehetőek”.

6. VizsgaBiztos.h

Ez az osztály valósítja meg a vizsgázás folyamatát menedzselő személyt. A vizsgabiztos három nyilvános, és két titkos attribútummal rendelkezik. Példányosításkor – mivel személye ismert, így azt nem kell tárolni –, csak a megfelelő kulcsok rendelődnek hozzá, majd a későbbiekben tárolja a javító tanár személyazonosságát elfedő véletlenszámot és a kapott erdményeket is, továbbá tartalmaz még néhány, csak általa végrehajtható metódust is.

7. Vizsgazo.h

Ez az osztály valósítja meg a vizsgázó személyét. Három privát és kettő publikus adattagot tartalmaz. A privát adattagok közül kettő, a vizsgázó személyazonosságának visszaállításához szükséges információkat tartalmaz, továbbá az osztály rendelkezik még a dolgozat megírásához szükséges metódusokkal is. Példányosításkor beállítja magának a vakított azonosítóját, melyet aláírat a biztossal.

8. JavitoTanar.h

Ez az osztály valósítja meg a javítást végző személyt. Három titkos – melyből kettő személyazonosságának elfedését és tárolását biztosítja –, és kettő nyilvános attribútuma van. Továbbá rendelkezik a dolgozat javítás-értékeléséhez szükséges metódusokkal is. Bár a személyazonosságát fedő attribútumai titkosak, azt mégsem közvetlenül ő állítja be.

9. Felugyelo.h

Ez az osztály valósítja meg a javítást végző személy adatainak beállítását. Attribútumai nincsenek, csak egyetlen nyilvános metódusa.

10. A program működéséhez szükséges adatállományok

A program működéséhez szükséges három adatállomány: az első a *Kiss_Peter.dat*, mely a vizsgázó személyes és vizsgára vonatkozó adatait, a második a *Nagy_Peter.dat*, mely a javítást végző személy adatait, a harmadik a *Dolgozat.dat*, mely egy dolgozatmintát tartalmaz.

11. A program fordítását segítő állomány

A program Linux rendszer alatt készült, így opcionálisan tartalmaz egy fordítást segítő *Makefile*-t is.

12. A rendszer működéséhez szükséges további programok

A program *C++* nyelven készült, a *GNU MP* könyvtár felhasználásával, amely a [9] címről származik.

5.3. A program működése

Az előzetes technikai információk ismertetése után lássuk a program működését lépésenként, összevetve az elméleti megvalósítással:

1. A rendszer létrehozza a vizsgabiztost, a felügyelőt és a javító tanárt.
2. A felügyelő beállítja a javító tanár személyazonosságát, és átadja a vizsgabiztosnak a javító tanár adatait rejtő véletlenszámot, majd ellenőrzésre kerül, hogy a tanár azonosítója hiteles-e.
3. A rendszer létrehozza a vizsgázó objektumot, aki ezt követően beállítja és hitelesítteti az azonosítóját és véletlenszámát a vizsgabiztossal, majd a rendszer ezt le is ellenőrzi.
4. A vizsgabiztos átadja a megírandó dolgozatot a vizsgázó részére aki megírja, majd azt követően digitálisan is aláírja és elküldi a javítást végző személynek.
5. A javítást végző személy ellenőrzi az aláírást, leosztályozza, majd digitálisan aláírja a dolgozatot és elküldi a vizsgabiztosnak.
6. A vizsgabiztos ellenőrzi a vizsgázó és a javító tanár digitális aláírását, majd azonosítójuk érvényességét, majd meggyőződik róla, hogy a vizsgázó a megfelelő dolgozatot írta-e meg, valamint azt, hogy a javító tanár a megfelelő dolgozatot javította-e. Ezt úgy teszi meg, hogy veszi a dolgozaton szereplő azonosítókat és összeveti azok tantárgykódjait.

Amennyiben egyeznek, akkor folytatódik, ellenkező esetben hibajelzéssel kilép a program.

7. A vizsgabiztos feljegyzi a vizsgázónak és a javítást végző személynek az azonosítóját, valamint a kapott érdemjegyet. Ezt követően eltávolítja a tanár azonosítóját és digitális aláírását, majd ő is aláírja a dolgozatot és elküldi a vizsgázónak.

8. A vizsgázó ellenőrzi a saját, majd a vizsgabiztos aláírását is, és ellenőrzi a kapott érdemjegyet, majd elküldi azonosítóját és véletlenszámát a vizsgabiztosnak.

9. A vizsgabiztos felfedi a vizsgázó személyazonosságát.

5.4. Eltérések az elméleti modelltől

A program írása folyamán felbukkantak kisebb problémák, melyek miatt el kellett térni kisebb mértékben az elméleti modelltől. A program írásának kezdetén az 1024 bites kulcsméret kiválasztás nagyon jónak bizonyult, mert a program futási ideje igen kedvező volt, viszont ez igaz még 2048 bites kulcs esetén is.

A következő eltérés az azonosító hosszának megválasztása. Itt az a probléma merült fel, hogy ha az azonosító hosszát a tervezett 128 karakter hosszúra választjuk és a kulcsméretünk 1024 bit hosszú, akkor a vizsgázó azonosítója nem mindig lesz hiteles. Ennek oka az volt, hogy az azonosító számmá való átkonvertálása esetén előfordult, hogy ez a szám nagyobb volt, mint a vizsgabiztos modulusa, így azok különbségét adta az ellenőrző művelet során. Célszerű a következő összefüggés alapján megválszatni a kulcsot és az azonosító hosszát:

$$\frac{\text{Kulcsméret}}{\text{Azonosító hossza}} > 8.$$

A programban jól megfigyelhető az, hogy több ellenőrzés van, mint amennyit megkövetel az elméleti modell. Ennek oka csak biztonsági és hitelességi jellegű, továbbá az, hogy a program futása időben megszakadjon, mielőtt egy felbukkanó probléma nagyobb „kárt” okozna.

A vizsgázó személyazonosságának felfedésekor a vizsgázó nemcsak a véletlenszámot, hanem az azonosítóját is elküldi. Ennek az az oka, hogy a vizsgabiztos még ezen a ponton is meggyőződhessen arról, hogy a ténylegesen arra jogosult személy kapja a megfelelő érdemjegyet. Ezt a letárolt azonosítóval való összevetéssel éri el.

Függelék

1. Egy ötlet a vizsgázó azonosítójának előállítására

Itt megadunk egy nagyon egyszerű ötletet, ami megadja, hogy hogyan lehetne egy azonosítót előállítani egy vizsgázó esetén. Természetesen ez az eljárás alkalmas mind a javító tanár, mind a vizsgabiztos esetén a számukra készítendő azonosító előállításánál is.

Itt talán kissé furcsának tűnhet a vizsga ideje, ami azt jelenti, hogy a vizsga 2009. május 27.-én, 11 órakor kezdődik és 120 perc hosszan tart.

Itt szeretném megjegyezni, hogy a különböző hosszú azonosítók konstruálásának elkerülése érdekében korlátozni kell a teljes azonosító hosszát, valamint a fel nem használt karakterek helyét kitölteni valamilyen „kitöltő karakterrel”.

Feladat: állítsunk elő ezekből az adatokból azonosítót!

Vizsga kódja	M01
Vizsga neve	Matematika 1.
Vizsga ideje	2009052711
Vizsga hossza	120
Vizsgázó személyi száma	1198204213581
Vizsgázó neve	Kiss Péter
Vizsgázó születési helye	Debrecen
Vizsgázó születési ideje	19810523
Vizsgázó édesanyjának neve	Nagy Viola
Vizsgázó állampolgársága	Magyar

1. táblázat. A vizsgázó azonosítójának elkészítéséhez szükséges adatok

Az azonosító elkészítéséhez vegyük a vizsga kódját, a vizsgázó nevét, a vizsga idejét és időtartamát, valamint a vizsgázó személyi számát. Ezt írjuk egymás mellé, konkatenációként véve például a # karaktert. Látható, hogy nem minden adatot használtunk fel az azonosító elkészítéséhez, de tetszőlegesen hozzávehetünk még.

Ezt követően a mezőhosszban fennmaradó üres helyet töltsük ki egy véletlenszámmal. Az így kapott karaktersorozat lesz a természetes azonosító, mely tartalmaz a vizsgázó számára minden szükséges adatot.

2. Egy ötlet a javító tanár azonosítójának előállítására

A javító tanár azonosítójának előállítása gyakorlatilag egyezik a vizsgázó azonosítójának előállításakor elmondottakkal. Azonban, itt néhány plusz információra is szükség van, amelyek közlik számunkra, hogy melyik vizsgáztató mit javíthat, továbbá mennyi az azonosítójának **érvényességi ideje**.

Vizsga kódja	M01
Vizsga neve	Matematika 1.
Az azonosító érvényességi ideje	20091012
Vizsgáztató személyi száma	1198204213581
Vizsgáztató neve	Nagy Péter
Vizsgáztató születési helye	Budapest
Vizsgáztató születési ideje	19510224
Vizsgáztató édesanyjának neve	Kiss Viola
Vizsgáztató állampolgársága	Magyar

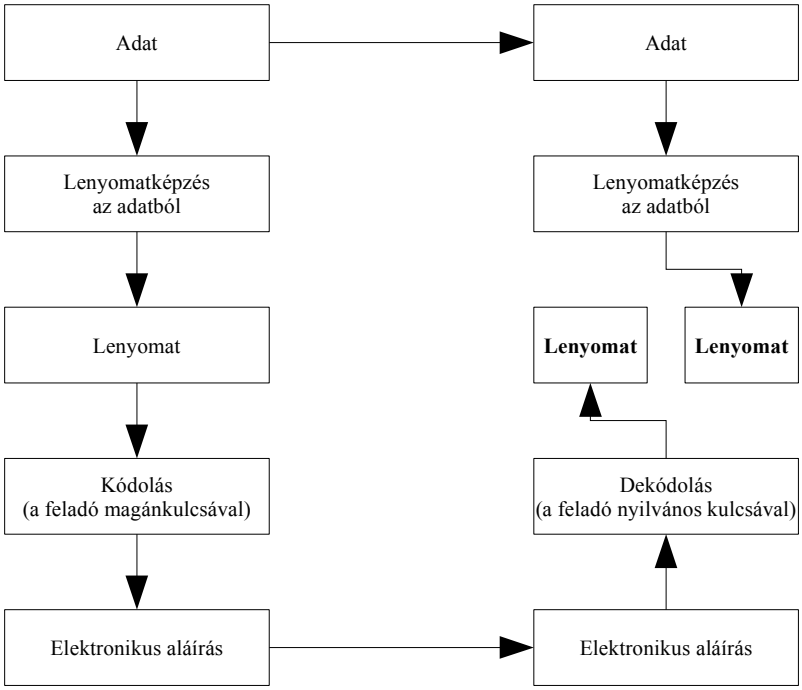
2. táblázat. A vizsgáztató azonosítójának elkészítéséhez szükséges adatok

3. A digitális aláírásról röviden

A digitális aláírás sémája és működése az 1. ábrán látható. A működése RSA alapjain nyugszik. Az ábra bal oldalán láthatjuk az aláírás elkészítését, a jobb olbalán pedig annak az ellenőrzésére használt eljárást.

Az elektronikus aláírást arra lehet használni, hogy egy aláírandó dokumentumot, vagy (nagy méretű dokumentum esetén) annak egy jellemzőkivonatát kódolja, így az aláírás végső tartalma függ az aláírt dokumentumtól is. Mondhatjuk azt is, hogy az aláírandó dokumentum valamilyen formában az aláírás része, és ha a dokumentum megváltozott, akkor az aláírás ellenőzése ezt ki fogja mutatni.

Csak abban az esetben hamisítható, ha valaki a titkosításra használt kulcsot megszerzi, így csak mi tudunk aláírni vele. Következésképp az aláíró nem tagadhatja le aláírását, és ez nem is hamisítható.



1. ábra. A digitális aláírás sémája és annak gyakorlati alkalmazása

Amennyiben az aláírás ellenőrzésekor a mi nyilvános kulcsunkkal sikeresen elolvasható egy aláírás, akkor azt biztosan a mi titkos kulcsunkkal írták alá, melyből következik, hogy az aláírás tőlünk származik (feltéve, hogy titkos kulcsunkat nem lopták el, ugyanis ebben az esetben ezt nekünk kell bizonyítani).

Most lássuk a digitális aláírást és annak működését. Ezt úgy kezdjük, hogy az aláírandó dokumentumból egy egyedi kivonatot (Message Digest – MD) veszünk[1], majd azt aláírjuk titkos kulcsunk segítségével, majd az üzenetet és a hozzá tartozó aláírást elküldjük a címzettnek. Most lássuk ezt formalizálva. Legyen az elküldendő üzenet m, titkos kulcs (d, n), nyilvános kulcs pedig (e, n).

[1]Ez a gyakorlatban egy előre definiált függvény.

Ekkor a következőképpen küldünk üzenetet:

1. Kiszámítjuk: $s = (MD(m))^d \bmod n$;
2. Elküldjük: $\{m, s\}$.

Itt az *MD* függvény vesz mintát a teljes sorozatból, azt kódolja a dokumentum tulajdonosának titkos kulcsával, majd ezt követően a dokumentumot és a rá jellemző aláírást elküldjük a célszemély részére.

Az aláírás ellenőrzéséhez a következőket kell tennünk:

1. $m_1 = s^e \bmod n$, $m_2 = MD(m)$;
2. Amennyiben
   (a) $m_1 = m_2$, akkor biztosak lehetünk benne, hogy a dokumentumunk nem változott, mióta a feladója elküldte;
   (b) egyébként az üzenet vagy a lenyomata megváltozott. Ezt az 1. ábrán a két lenyomat egyenlősége jelzi.

Röviden összefoglalva a digitális aláírás gyakorlatilag nem is „igazán” aláírás, hanem egy „kísérőlevél postai küldeményekhez az elektronikus utakon”. Mindezen tulajdonságok mellett az *MD* bitsorozathoz hozzáfűzhetjük az aláírás idejét, helyét, valamint egyéb fontosnak tartott jellemzőket is.

Irodalomjegyzék

[1] *Zuzana Rjašková*:
Electronic Voting Schemes.
Diplomává práca, April 2002.

[2] *Adi Shamir*:
How to share a secret.
Communications of the ACM, 22:612-613, 1979.

[3] *Virasztó Tamás*:
Titkosítás és adatrejtés
Biztonságos kommunikáció és algoritmikus adatvédelem,
NetAcademia Kft., 2004.

[4] *David L. Chaum*:
Untraceable electronic mail, return address, and digital pseudonym.
Communication of ACM 24, February 1981.

[5] *Kerckhoffs*:
”la Cryptographie Militaire,” Journel des Sciences militaires, 9th series. IX, January and February, 1883, Librarie Militaire de L. Baudoin & Co., Paris.
English trans. by Warren T., McCready of the University of Toronto, 1964.

[6] *Rózsahegyi Zsolt*:
Elektronikus aláírás, időbélyegzés, elektronikus hitelesítés című előadása a Debreceni Egyetemen, 2008.

[7] A program a `http://www.hoozi.com/Articles/SHA1_Source.htm` helyről származó **sha1** algoritmust megvalósító programot használja a lenyomatok elkészítéséhez.

[8] A program a kulcsok előállításához a `http://www.rajorshi.net/old/code_rsa.htm` helyről származó **RSA** algoritmust megvalósító program részeit használja.

[9] A program a nagy számok kezeléséhez a `http://gmplib.org` helyről származó **GNU MP Bignum Library** programcsomagot használja.

Köszönetnyilvánítás

Szeretném köszönetem kifejezni a következő embereknek, a diplomamunkám megírása során nyújtott segítségükért:

- *Prof. Dr. Pethő Attila* témavezetőmnek, a diplomamunka megírásához nyújtott segítségéért és támogatásáért;
- *Folláth Jánosnak*, a programozásban nyújtott segítségéért;
- *Oláh Gyula* barátomnak, a diplomamunka megírásához nyújtott hasznos tanácsaiért és támogatásáért;
- Unokatestvéremnek *Rácz Erikának* és férjének *Balogh Lászlónak*, a technikai akadályok leküzdésében való segítségükért és támogatásukért;
- *Édesanyámnak*, a felsőfokú tanulmányok elvégzését lehetővé tévő erőfeszítéseiért és támogatásáért.

Forráskód

1. A programot alkotó állományok

1.1. diplomamunka.cpp

```
using namespace std;

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstdio>
#include <gmp.h>
#include <gmpxx.h>
#include "sha1.h"
#include "globalis.h"
#include "rsa.h"
#include "Tablazat.h"
#include "VizsgaBiztos.h"
#include "Vizsgazo.h"
#include "JavitoTanar.h"
#include "Felugyelo.h"

int main(void)
{
    //A vizsgabiztos objektum letrehozasa
    printf("\nA VIZSGABIZTOS LETREHOZASA...\n");
    VizsgaBiztos vizsgabiztos;

    //A felugyelo objektum letrehozasa
    printf("\nA FELUGYELO LETREHOZASA...\n");
    Felugyelo felugyelo;

    //A javito tanar objektum letrehozasa
    printf("\nA JAVITOTANAR LETREHOZASA...\n");
    JavitoTanar javitotanar;

    printf("\nA FELUGYELO BEALLITJA A JAVITO TANAR AZONOSITOJAT...\n");

    //A felugyelo beallitja a T_{a} es T_{av} azonositot, es visszaadja
    //a javito tanar azonositojanak eloallitasahoz hasznalt veletlenszamot
```

```
mpz_t * javito_tanar_veletlenszam=felugyelo.Tanari_Azonosito_Elkeszites             \
("Nagy_Peter.dat",&javitotanar,&vizsgabiztos);
if(Globalis::MEGJELENIT) gmp_printf("\n[ A felugyelo altal generalt                 \
veletlenszam a javito tanar reszere : ]\n\n%Zd\n",javito_tanar_veletlenszam);

//A vizsgabiztos eltarolja a veletlenszamot
vizsgabiztos.Beallit_JavitoTanar_VeletlenSzam(javito_tanar_veletlenszam);

//Hiteles-e a javito tanarnak a vizsgabiztos altal hitelesitett azonositoja
if(!Globalis::Hiteles(javitotanar.Atad_Azonosito(),vizsgabiztos._Nyilvanos_Kulcs,   \
vizsgabiztos._Modulus))
{
    if(Globalis::MEGJELENIT) printf("\n\t[ A javito tanar azonositoja ervenyes! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t[ A javito tanar azonositoja ervenytelen,  \
    a program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgazo objektum letrehozasa
printf("\nA VIZSGAZO LETREHOZASA...\n");
Vizsgazo vizsgazo("Kiss_Peter.dat",vizsgabiztos._Nyilvanos_Kulcs,                  \
vizsgabiztos._Modulus);

//A vizsgazo azonositojanak es veletlenszamanak hitelesitese
if(Globalis::MEGJELENIT) printf("\t[ A vizsgazo vakitott azonositojanak atadasa     \
hitelesitesre... ]\n");
mpz_t * vizsgazo_azonosito=vizsgazo.Atad_Azonosito_Hitelesitesre();
vizsgazo_azonosito=vizsgabiztos.Hitelesit(vizsgazo_azonosito);

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo a vizsgabiztos altal hitelesitett \
azonositojanak beallitasa... ]\n");
vizsgazo.Hitelesitett_Azonosito_Beallitas(vizsgabiztos._Modulus);

//Hiteles-e a vizsgazonak a vizsgabiztos altal hitelesitett
//azonositoja es veletlenszama?
if(!Globalis::Hiteles(vizsgazo.Atad_Azonosito(),vizsgabiztos._Nyilvanos_Kulcs,     \
vizsgabiztos._Modulus))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo azonositoja ervenyes! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo azonositoja ervenytelen,   \
    a program futasa megszakad! ]\n");
    exit(-1);
}

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo vakitott veletlenszamanak atadasa \
hitelesitesre... ]\n");
mpz_t * vizsgazo_veletlenszam=vizsgazo.Atad_VeletlenSzam_Hitelesitesre();
vizsgazo_veletlenszam=vizsgabiztos.Hitelesit(vizsgazo_veletlenszam);
```

```
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo a vizsgabiztos altal hitelesitett \
veletlenszamanak beallitasa... ]\n");
vizsgazo.Hitelesitett_VeletlenSzam_Beallitas(vizsgabiztos._Modulus);

if(!Globalis::Hiteles(vizsgazo.Atad_VeletlenSzam(),vizsgabiztos._Nyilvanos_Kulcs,   \
vizsgabiztos._Modulus))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo                              \
    veletlenszama ervenyes! ]\n\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo veletlenszama ervenytelen,  \
    a program futasa megszakad! ]\n\n");
    exit(-1);
}

printf("\nA DOLGOZAT ATADASA A VIZSGAZO RESZERE...\n");
if(Globalis::MEGJELENIT) printf("\n\t[ A dolgozat megirasa... ]\n");

//A vizsgabiztos atadja a megirando dolgozatot a vizsgazo reszere
char * vizsgazo_dolgozat=vizsgabiztos.Atad_Dolgozat("Dolgozat.dat");

vizsgazo_dolgozat=vizsgazo.Megir_Dolgozat(vizsgazo_dolgozat);
unsigned char * lenyomat1=vizsgazo.Digitalisan_Alair(vizsgazo_dolgozat);

if(Globalis::MEGJELENIT) printf("\n\t[ A dolgozatnak a vizsgazo altal               \
alairt lenyomata : ] \n\n%s\n\n",lenyomat1);

//A dolgozat elkuldese a javito tanarnak javitas-ertekelesre
printf("\nA MEGIRT ES ALAIRT DOLGOZAT ATADASA A JAVITO TANAR RESZERE...\n");

if(Globalis::MEGJELENIT) printf("\n\t[ A javito tanar ellenorzi a dolgozat          \
digitalis alairasat... ]\n");

//A vizsgazo alairasanak ellenorzese
if(!Globalis::Ellenoriz_Alairas(vizsgazo_dolgozat,vizsgazo._Nyilvanos_Kulcs,        \
vizsgazo._Modulus,3))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas rendben! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas korrupt!                     \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A javito tanar osztalyozza a dolgozatot
if(Globalis::MEGJELENIT) printf("\n\t[ A dolgozat osztalyzasa... ]\n");
vizsgazo_dolgozat=javitotanar.Leosztalyoz_Dolgozat(vizsgazo_dolgozat);
```

```
if(Globalis::MEGJELENIT) printf("\n\t[ A javito tanar digitalisan alairja          \
a dolgozatot... ]\n");

//A javito tanar digitalisan alarja a dolgozatot
unsigned char * lenyomat2=javitotanar.Digitalisan_Alair(vizsgazo_dolgozat);
if(Globalis::MEGJELENIT) printf("\n\t[ A dolgozatnak a javito tanar altal          \
alairt lenyomata : ] \n\n%s\n\n",lenyomat2);

printf("\nA JAVITOTT ES ALAIRT DOLGOZAT ATADASA A VIZSGABIZTOS RESZERE...\n");

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi a dolgozat vizsgazo \
altal vegzett digitalis alairasat... ]\n");

//A vizsgabiztos ellenorzi a vizsgazo digitalis alairasat
if(!Globalis::Ellenoriz_Alairas(vizsgazo_dolgozat,vizsgazo._Nyilvanos_Kulcs,       \
vizsgazo._Modulus,3))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas rendben! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas korrupt!                   \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgabiztos ellenorzi a javito tanar digitalis alairasat
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi a dolgozat         \
javito tanar altal vegzett digitalis alairasat... ]\n");

if(!Globalis::Ellenoriz_Alairas(vizsgazo_dolgozat,javitotanar._Nyilvanos_Kulcs,    \
javitotanar._Modulus,7))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas rendben! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas korrupt!                   \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgabiztos ellenorzi a vizsgazo azonositojanak ervenyesseget
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi a vizsgazo         \
azonositojanak ervenyesseget... ]\n");

if(!Globalis::Altalanos_Hiteles(vizsgazo_dolgozat,vizsgabiztos._Nyilvanos_Kulcs,   \
vizsgabiztos._Modulus,2))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az azonosito rendben! ]\n");
}
```

```
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az azonosito ervenytelen!              \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgabiztos ellenorzi a javito tanar azonositojanak ervenyesseget
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi a javito tanar      \
azonositojanak ervenyesseget... ]\n");

if(!Globalis::Altalanos_Hiteles(vizsgazo_dolgozat,vizsgabiztos._Nyilvanos_Kulcs,    \
vizsgabiztos._Modulus,6))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az azonosito rendben! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az azonosito ervenytelen!              \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgabiztos ellenorzi, hogy a megfelelo dolgozat kerul-e ellenorzesre
unsigned char * v_azon=Globalis::Milyen_Tantargy(vizsgazo_dolgozat,2);

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi, hogy a megfelelo   \
dolgozatot adta be a vizsgazo... ]\n");

if(Globalis::Letezik_Tantargy(v_azon))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo ervenyes dolgozatot         \
    adott be! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo ervenytelen dolgozatot      \
    adott be! A program futasa megszakad ]\n");
    exit(-1);
}

//A vizsgabiztos ellenorzi, hogy a megfelelo dolgozat kerult-e javitasra
unsigned char * v_javt=Globalis::Milyen_Tantargy(vizsgazo_dolgozat,6);

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi,                   \
hogy a javito tanar jo dolgozatot javitott-e... ]\n");

if(Globalis::Letezik_Tantargy(v_javt))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A javito tanar ervenyes dolgozatot     \
    javitott! ]\n");
}
```

```
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A javito tanar ervenytelen dolgozatot  \
    javitott! A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgabiztos ellenorzi, hogy a ket dolgozat egyezik-e
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos ellenorzi,                    \
hogy a ket dolgozat egyezik-e... ]\n");

if(!strcmp((char*)v_azon,(char*)v_javt))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A ket dolgozat egyezik! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A ket dolgozat nem egyezik!           \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgabiztos feljegyzi a szukseges informaciokat
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos feljegyzi a vizsgazo es      \
a javito tanar azonositojat, es a jegyet... ]\n");
vizsgabiztos.Feljegyez_Eredmeny(vizsgazo_dolgozat);

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos eltavolitja a javito tanar   \
alairasat es azonositojat... ]\n");

//A javito tanar azonositojanak es alairasanak eltavolitasa
vizsgazo_dolgozat=vizsgabiztos.Eltavolit_Tanar_Azonosito(vizsgazo_dolgozat);

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos digitalisan alairja          \
a dolgozatot... ]\n");

//A vizsgabiztos digitalisan alairja a dolgozatot
unsigned char * vizsgabiztos_digitalis_alairasa=                                  \
vizsgabiztos.Digitalisan_Alair(vizsgazo_dolgozat);

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgabiztos digitalis alairasa...]       \
\n\n%s\n",vizsgabiztos_digitalis_alairasa);

//A vizsgabiztos elkuldi a dolgozatot a vizsgazonak
printf("\nA VIZSGABIZTOS ELKULDI A DOLGOZATOT A VIZSGAZO RESZERE...\n");

//A vizsgazo ellenorzi a sajat alairasanak valtozatlansagat
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo ellenorzi alairasanak            \
valtozatlansagat... ]\n");
```

```
if(!Globalis::Ellenoriz_Alairas(vizsgazo_dolgozat,vizsgazo._Nyilvanos_Kulcs,        \
vizsgazo._Modulus,3))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas rendben! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas korrupt!                    \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgazo ellenorzi a vizsgabiztos alairasat
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo ellenorzi                        \
a vizsgabiztos alairasat... ]\n");

if(!Globalis::Ellenoriz_Alairas(vizsgazo_dolgozat,vizsgabiztos._Nyilvanos_Kulcs,    \
vizsgabiztos._Modulus,6))
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas rendben! ]\n");
}
else
{
    if(Globalis::MEGJELENIT) printf("\n\t\t[ Az alairas korrupt!                    \
    A program futasa megszakad! ]\n");
    exit(-1);
}

//A vizsgazo ellenorzi a kapott erdemjegyet
if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo ellenorzi a kapott               \
erdemjegyet...]\n");

if(vizsgazo.Ellenoriz_Erdemjegy(vizsgazo_dolgozat))
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo elegedett                   \
    a javitassal... ]\n");
else
    if(Globalis::MEGJELENIT) printf("\n\t\t[ A vizsgazo nem elegedett              \
    a javitassal... ]\n");

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo atadja a veletlenszamat es       \
azonositojat a vizsgabiztosnak... ]\n");

mpz_t * atadott_veletlenszam=vizsgazo.Atad_VeletlenSzam();
mpz_t * atadott_azonosito=vizsgazo.Atad_Azonosito();

//A vizsgabiztos felfedi a vizsgazo szemelyazonossagat
printf("\nA VIZSGABIZTOS FELFEDI A VIZSGAZO SZEMELYET...\n");

if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo szemelye... ]\n");

unsigned char * vizsgazo_szemelye=                                              \
vizsgabiztos.Felfed_Szemelyazonossag(atadott_azonosito,atadott_veletlenszam);
```

```
    if(vizsgazo_szemelye)
    {
        if(Globalis::MEGJELENIT) printf("\n%s\n",Globalis::ELVALASZTO);

        if(Globalis::MEGJELENIT) printf("%s\n",vizsgazo_szemelye);

        if(Globalis::MEGJELENIT) printf("%s\n",Globalis::ELVALASZTO);
    }
    else
    {
        if(Globalis::MEGJELENIT) printf("\n\t[ A vizsgazo azonositoja ervenytelen!      \
        A program futasa megszakad! ]\n");
        exit(-1);
    }

    if(Globalis::MEGJELENIT) printf("\n\t[ A program futasa szabalyosan                  \
    veget ert. ]\n\n");

    return 0;
}
```

1.2. sha1.h

```
#define rotateleft(x,n) ((x<<n) | (x>>(32-n)))
#define rotateright(x,n) ((x>>n) | (x<<(32-n)))

class SHA1
{
    private :

        //A lenyomatot tarolo attributum
        unsigned char * sha1_hash;

    unsigned char * SHA1_Hash(unsigned char * str1)
    {
        unsigned long int h0,h1,h2,h3,h4,a,b,c,d,e,f,k,temp;
        int i,j;

            h0 = 0x67452301;
            h1 = 0xEFCDAB89;
            h2 = 0x98BADCFE;
            h3 = 0x10325476;
            h4 = 0xC3D2E1F0;

        unsigned char * str;
        str = (unsigned char *)malloc(strlen((const char *)str1)+100);
        strcpy((char *)str,(const char *)str1);

        int current_length = strlen((const char *)str);
        int original_length = current_length;
        str[current_length] = 0x80;
        str[current_length + 1] = '\0';

        current_length++;
```

```
int ib = current_length % 64;

if(ib<56)
    ib = 56-ib;
else
    ib = 120 - ib;

for(i=0;i < ib;i++)
{
    str[current_length]=0x00;
    current_length++;
}

str[current_length + 1]='\0';

for(i=0;i<6;i++)
{
    str[current_length]=0x0;
    current_length++;
}

str[current_length] = (original_length * 8) / 0x100 ;
current_length++;
str[current_length] = (original_length * 8) % 0x100;
current_length++;
str[current_length+i]='\0';

int number_of_chunks = current_length/64;
unsigned long int word[80];

for(i=0;i<number_of_chunks;i++)
{
    for(j=0;j<16;j++)

        word[j] = str[i*64 + j*4 + 0] * 0x1000000 + str[i*64 + j*4 + 1] *       \
        0x10000 + str[i*64 + j*4 + 2] * 0x100 + str[i*64 + j*4 + 3];

    for(j=16;j<80;j++)

        word[j] = rotateleft((word[j-3] ^ word[j-8] ^ word[j-14] ^              \
        word[j-16]),1);

    a = h0;
    b = h1;
    c = h2;
    d = h3;
    e = h4;
```

```
        for(int m=0;m<80;m++)
        {
            if(m<=19)
            {
                f = (b & c) | ((~b) & d);
                k = 0x5A827999;
            }
            else if(m<=39)
            {
                f = b ^ c ^ d;
                k = 0x6ED9EBA1;
            }
            else if(m<=59)
            {
                f = (b & c) | (b & d) | (c & d);
                k = 0x8F1BBCDC;
            }
            else
            {
                f = b ^ c ^ d;
                k = 0xCA62C1D6;
            }

        temp = (rotateleft(a,5) + f + e + k + word[m]) & 0xFFFFFFFF;
        e = d;
        d = c;
        c = rotateleft(b,30);
        b = a;
        a = temp;
    }

        h0 = h0 + a;
        h1 = h1 + b;
        h2 = h2 + c;
        h3 = h3 + d;
        h4 = h4 + e;
    }

    unsigned char * Vissza=new unsigned char[46];

    sprintf((char*)Vissza,"%lX%lX%lX%lX%lX",h0, h1, h2, h3, h4);

    return Vissza;
}

public :

SHA1(unsigned char * Uzenet)
{
    sha1_hash=SHA1_Hash(Uzenet);
}
```

```
    unsigned char * sha1_alairas(void)
    {
        return sha1_hash;
    }
};
```

1.3. globalis.h

```
class Globalis
{
    public :

        static const int KULCSMERET;
        static const int AZONOSITOHOSSZ;
        static const int MEGJELENIT;
        static const int KERDESEKSZAMA;
        static const int TANTARGYDARABSZAM;
        static const char * ELVALASZTO;
        static const char * HELYESVALASZOK;
        static const char * ERDEMJEGYEK[];
        static const char * TANTARGYKODOK[];

        //A veletlenszamok generalasahoz beallitja
        //az srand() fuggveny kezdoerteket
        static void VeletlenSzam_Generator_Indit(void)
        {
            sleep(1);
            srand((unsigned)time(NULL));
        }

        //Ez a metodus meghatarozza, hogy hany tantargy
        //van akkreditalva a rendszeren belul
        static int Meghataroz_Tantargy_Darabszam(const char ** Tomb)
        {
            int Vissza=0;

                for(int i=0;Tomb[i];i++)

                    if(Tomb[i]!=NULL) Vissza++;

            return Vissza;
        }

        //Ez a fuggveny egy AZONOSITOHOSSZ hosszusagu
        //karaktersorozatbol eloallitja a kettes
        //szamrendszerbeli alakjat
        static void Atalakit_Szo_Kettes(unsigned char * Szo, unsigned char * Vissza)
        {
            int Hossz=Globalis::AZONOSITOHOSSZ,i=0;
            for(i=0;i<Hossz;i++)
            {
                Vissza[0+(i*8)]=((Szo[i] & 0x80)>>7)+'0';
                Vissza[1+(i*8)]=((Szo[i] & 0x40)>>6)+'0';
                Vissza[2+(i*8)]=((Szo[i] & 0x20)>>5)+'0';
```

```
        Vissza[3+(i*8)]=((Szo[i] & 0x10)>>4)+'0';
        Vissza[4+(i*8)]=((Szo[i] & 0x08)>>3)+'0';
        Vissza[5+(i*8)]=((Szo[i] & 0x04)>>2)+'0';
        Vissza[6+(i*8)]=((Szo[i] & 0x02)>>1)+'0';
        Vissza[7+(i*8)]=((Szo[i] & 0x01)>>0)+'0';
    }
    Vissza[i*8]='\0';
}

//Ez a fuggveny egy kettes szamrendszerbeli alakbol sztringet allit elo
static void Atalakit_Kettes_Szo(unsigned char * Szo, unsigned char * Vissza)
{
    int Hossz=strlen((char*)Szo)/8,i=0,j=0,ketto=0;

    for(i=0,Vissza[i]=0;i<Hossz;i++,Vissza[i]=0)

        for(ketto=128;ketto>0;j++,ketto/=2)

            Vissza[i]+=(ketto*(Szo[j]-'0'));

    Vissza[i]='\0';
}

//Ez a metodus megvizsgalja, hogy letezo-e a tantargy,
//melybol vizsgat szeretnek tenni
static int Letezik_Tantargy(unsigned char * Nev)
{
    for(int i=0;i<Globalis::TANTARGYDARABSZAM;i++)

        if(!strcmp((char*)Nev,Globalis::TANTARGYKODOK[i])) return 1;

    return 0;
}

//A fajlbol karakterenkent kiolvassa vizsgazo adatait,
//es szuksegkeppen kitolti a fennmarado helyeket
static void Adatok_Kiolvasasa_Kitoltessel(char * FajlNev,                    \
unsigned char * Buffer)
{
    int i=0,szamlal=0;
    char Karakter;
    ifstream Fajl(FajlNev,ios::in);

    while(szamlal<Globalis::AZONOSITOHOSSZ && Fajl.get(Karakter))

        Buffer[i++]=(unsigned char)Karakter; szamlal++;

    for(;szamlal<Globalis::AZONOSITOHOSSZ && i<Globalis::AZONOSITOHOSSZ;i++)
    {
        Buffer[i]=(unsigned char)((((double)10.0*rand())/RAND_MAX)+'0');
        if(Buffer[i]=='0') i--;
    }
    Buffer[i]='\0';
```

```
    Fajl.close();
}

//Egy veletlenszamokat tartalmazo tombot allit elo a Vernam-kodolashoz
static void VeletlenSzam_Eloallitas(unsigned char * VeletlenSzam)
{
    int i=0;

    VeletlenSzam[0]=VeletlenSzam[1]=VeletlenSzam[2]='0';

    for(i=3;i<Globalis::AZONOSITOHOSSZ;i++)
    {
        VeletlenSzam[i]=(unsigned char)((((double)10.0*rand())/RAND_MAX)+'0');

        //Erre azert van szukseg, hogy a tenyleges veletlenszam
        //ne kezdodhessen nulla karakterrel
        if(VeletlenSzam[i]=='0') i--;
    }
    VeletlenSzam[i]='\0';
}

//Vernam-kodolast vegzo metodus
static void Vernam_Kodolas(unsigned char * VernamBuffer, unsigned char * Buffer,\
unsigned char * VeletlenSzam)
{
    int i=0;

    for(i=0;i<Globalis::AZONOSITOHOSSZ;i++)

        VernamBuffer[i]=(VeletlenSzam[i] ^ Buffer[i]);

    VernamBuffer[i]='\0';
}

//Az azonosito es a veletlenszam vakitasat vegzi el
static void Vakitas(mpz_t * Mit, mpz_t * Mibe, mpz_t * _VeletlenSzam,          \
const mpz_t * vNyilvanos_Kulcs, const mpz_t * vModulus)
{
    mpz_t temp;
    mpz_init(temp);

        mpz_powm(temp,*_VeletlenSzam,*vNyilvanos_Kulcs,*vModulus);
        mpz_mul(temp,temp,*Mit);
        mpz_powm_ui(*Mibe,temp,(unsigned long)1,*vModulus);

    mpz_clear(temp);
}

//Ellenorzi, hogy az azonositok hitelesek-e
static int Hiteles(mpz_t * Azonosito, const mpz_t * vNyilvanos_Kulcs,          \
const mpz_t * vModulus)
{
    mpz_t temp;
    mpz_init(temp);
```

```
    mpz_powm(temp,Azonosito[1],*vNyilvanos_Kulcs,*vModulus);

    int Vissza=mpz_cmp(Azonosito[0],temp);
    mpz_clear(temp);

    return Vissza;
}

//A vizsgazo vagy javito tanar digitalis alairasanak ellenorzeset elvegzo metodus
static int Ellenoriz_Alairas(char * FajlNev, const mpz_t * v_Nyilvanos,         \
const mpz_t * v_Modulus, int SorSzam)
{
    unsigned char Dolgozat[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    unsigned char Alairas[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];

    //A vizsgazo vagy a javito tanar beolvassa az egesz dolgozatot egy tombbe,
    //majd abbol kesziti el a megfelelo lenyomatot
    int i=0,szamlal=0;
    char Karakter;

    ifstream Fajl(FajlNev,ios::in);

    //A dolgozat beolvasasa, a megfelelo pozicioig
    while(Fajl.get(Karakter))
    {
        if(szamlal==(int)strlen(Globalis::ELVALASZTO)*SorSzam) break;
        Dolgozat[i++]=(unsigned char)Karakter;
        if(Karakter=='*') szamlal++;
    }
    Dolgozat[i++]=(unsigned char)Karakter;
    Dolgozat[i]='\0';

    //A vizsgazo vagy javito tanar altal digitalisan alairt
    //dolgozat alairasanak beolvasasa
    i=0;

    while(Fajl.get(Karakter))
    {
        if(Karakter=='\n') break;
        Alairas[i++]=(unsigned char)Karakter;
    }
    Alairas[i]='\0';

Fajl.close();

//Most vegre kell hajtani a dolgozat
//valtozatlansaganak ellenorzeset,vagyis
//lenyomatot kell kesziteni a dolgozatrol
SHA1 dlenyomat(Dolgozat);

//A lenyomat szamma alakitasa
mpz_t lenyomat_dolgozat;
mpz_init_set_str(lenyomat_dolgozat,(char*)dlenyomat.sha1_alairas(),16);
```

```
    //A dolgozathoz hozzafuzott lenyomat
    //beolvasasa es szamma alakitasa
    mpz_t eredeti_lenyomat;
    mpz_init_set_str(eredeti_lenyomat,(char*)Alairas,10);

    mpz_t temp;
    mpz_init(temp);

    mpz_powm(temp,eredeti_lenyomat,*v_Nyilvanos,*v_Modulus);

    //A lenyomatok osszevetese
    int Vissza=mpz_cmp(temp,lenyomat_dolgozat);

    mpz_clear(temp);
    mpz_clear(eredeti_lenyomat);
    mpz_clear(lenyomat_dolgozat);

    return Vissza;
}

//Ez a metodus az allomanyokban valo digitalis alairasok ellenorzeset vegzi
static int Altalanos_Hiteles(char * FajlNev,const mpz_t * vNyilvanos_Kulcs,        \
const mpz_t * vModulus, int SorSzam)
{
    char Buffer[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    int szamlal=0;

    FILE * in=fopen(FajlNev,"r");

    //Pozicionalas a megfelelo pozicioig
    while(fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in))
    {
        Buffer[strlen(Buffer)-1]='\0';
        if(!strcmp(Buffer,Globalis::ELVALASZTO))szamlal++;
        if(szamlal==SorSzam) break;
    }

    //A megfelelo adat kiolvasasa
    fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in);
    Buffer[strlen(Buffer)-1]='\0';
    mpz_t * temp=new mpz_t[2];

    mpz_init_set_str(temp[0],Buffer,10);

    fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in);
    Buffer[strlen(Buffer)-1]='\0';

    mpz_init_set_str(temp[1],Buffer,10);

    //Osszevetes a kivant ertekkel
    int Vissza=Globalis::Hiteles(temp,vNyilvanos_Kulcs,vModulus);

    mpz_clear(temp[0]);
```

```
    mpz_clear(temp[1]);

    fflush(in);
    fclose(in);

    return Vissza;
}

//Ez a metodus ellenorzi, hogy a megfelelo dolgozat volt-e megirva
static unsigned char * Milyen_Tantargy(char * FajlNev, int SorSzam)
{
    unsigned char Buffer[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    int szamlal=0,i;
    FILE * in=fopen(FajlNev,"r");

    //Pozicionalas az allomanyban
    while(fgets((char*)Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in))
    {
        Buffer[strlen((char*)Buffer)-1]='\0';
        if(!strcmp((char*)Buffer,Globalis::ELVALASZTO))szamlal++;
        if(szamlal==SorSzam) break;
    }

    //A megfelelo ertek beolvasasa
    fgets((char*)Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in);
    Buffer[strlen((char*)Buffer)-1]='\0';

    mpz_t temp;
    mpz_init_set_str(temp,(char*)Buffer,10);

    unsigned char * Buffer_Ketto=(unsigned char*)mpz_get_str(NULL,2,temp);
    unsigned char temp_tomb[Globalis::AZONOSITOHOSSZ+1];
    unsigned char Buffer_Vissza[Globalis::AZONOSITOHOSSZ+1];

    //A tantargykod "beolvasasa"
    for(i=0,Buffer_Vissza[0]='0';Buffer_Ketto[i];i++)

        Buffer_Vissza[i+1]=Buffer_Ketto[i];

    Buffer_Vissza[i+1]='\0';

    Atalakit_Kettes_Szo(Buffer_Vissza,temp_tomb);

    unsigned char * Vissza=new unsigned char[4];

    //A tantargykod "eloallitasa"
    for(i=0;i<3;i++)

        Vissza[i]=temp_tomb[i]^'0';

    Vissza[i]='\0';

    mpz_clear(temp);
    delete[] Buffer_Ketto;
```

```
        fflush(in);
        fclose(in);

        return Vissza;
    }
};

        const int Globalis::KULCSMERET=1024;
        const int Globalis::AZONOSITOHOSSZ=127;
        const int Globalis::MEGJELENIT=1;
        const int Globalis::KERDESEKSZAMA=strlen((char*)HELYESVALASZOK);
        const int Globalis::TANTARGYDARABSZAM=                                    \
        Meghataroz_Tantargy_Darabszam(Globalis::TANTARGYKODOK);
        const char * Globalis::HELYESVALASZOK="dab";
        const char * Globalis::TANTARGYKODOK[]={"M01",NULL};
        const char * Globalis::ELVALASZTO="********************";
        const char * Globalis::ERDEMJEGYEK[]=                                     \
        {NULL,"ELEGTELEN","ELEGSEGES","KOZEPES","JO","JELES"};
```

1.4. rsa.h

```
class RSA_Kulcs_Generator
{
    private :

        mpz_t _Modulus;
        mpz_t _Nyilvanos_Kulcs;
        mpz_t _Titkos_Kulcs;
        int   _KulcsMeret;

    public :

    RSA_Kulcs_Generator(int KulcsMeret)
    {
        _KulcsMeret=KulcsMeret;
        RSA_Kulcs_Generalas();
    }

    ~RSA_Kulcs_Generator()
    {
        mpz_clear(_Modulus);
        mpz_clear(_Nyilvanos_Kulcs);
        mpz_clear(_Titkos_Kulcs);
    }

    //RSA eljaras kulcsok generalasahoz
    void RSA_Kulcs_Generalas(void)
    {
        int PrimMeret = _KulcsMeret/2;

        //A veletlenszam generator inicializalasa
        Globalis::VeletlenSzam_Generator_Indit();
```

```
mpz_t p,q,n,e,d,x;
mpz_init(p);
mpz_init(q);
mpz_init(n);
mpz_init(e);
mpz_init(d);
mpz_init(x);

//A veletleszamok tarolasahoz szukseges tombok
char* p_str = new char[PrimMeret+1];
char* q_str = new char[PrimMeret+1];

p_str[0] = '1';
q_str[0] = '1';

//Veletlenszamok generalasa
for(int i=1;i<PrimMeret;i++)
    p_str[i] = (int)(2.0*rand()/(RAND_MAX+1.0)) + 48;

for(int i=1;i<PrimMeret;i++)
    q_str[i] = (int)(2.0*rand()/(RAND_MAX+1.0)) + 48;

//A veletlenszamok beallitasa ugy,
//hogy kellokeppen messze legyenek egymastol
if(p_str[1]=='0') q_str[1]='1';
else if(p_str[1]=='1') q_str[1]='0';

p_str[PrimMeret] = '\0';
q_str[PrimMeret] = '\0';

//A p es q ertekenek beallitasa az
//elozoekben generalt sztringekbol
mpz_set_str(p,p_str,2);
mpz_set_str(q,q_str,2);

//A p-nel es a q-nal legkozelebbi es nagyobb
//primszamok beallitasa
mpz_nextprime(p,p);
mpz_nextprime(q,q);

if(Globalis::MEGJELENIT) gmp_printf("\n[ A generalt p: ]\n\n%Zd\n\n",p);
if(Globalis::MEGJELENIT) gmp_printf("[ A generalt q: ]\n\n%Zd\n",q);

//Az n = p*q ertek kiszamitasa
mpz_mul(n,p,q);

//Az x = (p-1)*(q-1)  kiszamitasa
mpz_t p_minus_1,q_minus_1;

mpz_init(p_minus_1);
mpz_init(q_minus_1);

mpz_sub_ui(p_minus_1,p,(unsigned long int)1);
mpz_sub_ui(q_minus_1,q,(unsigned long int)1);
```

```
    mpz_mul(x,p_minus_1,q_minus_1);

    //A legnagyobb kozos oszto kiszamitasa: gcd(e,x)=1
    mpz_t gcd;
    mpz_init(gcd);

    unsigned long int e_int = 65537;

    while(true)
    {
        mpz_gcd_ui(gcd,x,e_int);
        if(mpz_cmp_ui(gcd,(unsigned long int)1)==0) break;
        e_int += 2;
    }

    mpz_init_set_ui(e,e_int);

    //Az e*d = 1 mod x megoldasa d-re
    if(mpz_invert(d,e,x)==0)
    {
        printf("Multiplikativ inverz nem talalhato!\n");
        RSA_Kulcs_Generalas();
    }

    //A megfelelo tagok ertekenek beallitasa
    mpz_init_set(_Modulus,n);
    mpz_init_set(_Nyilvanos_Kulcs,e);
    mpz_init_set(_Titkos_Kulcs,d);

    //A lefoglalt helyek felszabaditasa
    mpz_clear(p);
    mpz_clear(q);
    mpz_clear(x);
    mpz_clear(n);
    mpz_clear(e);
    mpz_clear(d);
    mpz_clear(p_minus_1);
    mpz_clear(q_minus_1);
    mpz_clear(gcd);
    delete  []p_str;
    delete  []q_str;
}

//A modulus exportalasa
mpz_t * Exportal_Modulus(void)
{
    mpz_t * Export=new mpz_t[1];
        mpz_init(Export[0]);
        mpz_set(Export[0],_Modulus);
    return Export;
}
```

```
    //A nyilvanos kulcs exportalasa
    mpz_t * Exportal_Nyilvanos_Kulcs(void)
    {
        mpz_t * Export=new mpz_t[1];
            mpz_init(Export[0]);
            mpz_set(Export[0],_Nyilvanos_Kulcs);
        return Export;
    }

    //A titkos kulcs exporatlasa
    mpz_t * Exportal_Titkos_Kulcs(void)
    {
        mpz_t * Export=new mpz_t[1];
            mpz_init(Export[0]);
            mpz_set(Export[0],_Titkos_Kulcs);
        return Export;
    }

};
```

1.5. Tablazat.h

```
class Tablazat
{
    public :

        unsigned char * _Tanar_Azonosito;
        unsigned char * _Vizsgazo_Azonosito;
        unsigned char * _Erdemjegy;

    Tablazat(unsigned char * tAzon, unsigned char * vAzon, unsigned char * vJegy)
    {
        _Tanar_Azonosito=new unsigned char[strlen((char*)tAzon)+1];
        strcpy((char*)_Tanar_Azonosito,(char*)tAzon);

        _Vizsgazo_Azonosito=new unsigned char[strlen((char*)vAzon)+1];
        strcpy((char*)_Vizsgazo_Azonosito,(char*)tAzon);

        _Erdemjegy=new unsigned char[strlen((char*)vJegy)+1];
        strcpy((char*)_Erdemjegy,(char*)vJegy);
    }
};
```

1.6. VizsgaBiztos.h

```
class VizsgaBiztos
{
    private :

        const mpz_t * _Titkos_Kulcs;

        //A vizsgabiztos tarolja a javito tanar
        //szemelyazonossagat biztosito veletlenszamot
```

```
    mpz_t * _JavitoTanar_VeletlenSzam;

    //A vizsabiztos itt tarolja az erdemjegyet,
    //a javito es a vizsgazo szemelyet
    Tablazat * _Eredmenyek;

public :

    const mpz_t * _Nyilvanos_Kulcs;
    const mpz_t * _Modulus;

//A vizsgabiztost letrehozo konstruktor
VizsgaBiztos()
{
    RSA_Kulcs_Generator Generator(Globalis::KULCSMERET);

    _Modulus=Generator.Exportal_Modulus();
    if(Globalis::MEGJELENIT) gmp_printf("\n[ A vizsgabiztos                          \
    modulusa : ]\n\n%Zd\n\n",_Modulus);
    _Nyilvanos_Kulcs=Generator.Exportal_Nyilvanos_Kulcs();
    if(Globalis::MEGJELENIT) gmp_printf("[ A vizsgabiztos                          \
    nyilvanos kulcsa : ]\n\n%Zd\n\n",_Nyilvanos_Kulcs);
    _Titkos_Kulcs=Generator.Exportal_Titkos_Kulcs();
    if(Globalis::MEGJELENIT) gmp_printf("[ A vizsgabiztos                          \
    titkos kulcsa : ]\n\n%Zd\n",_Titkos_Kulcs);
}

~VizsgaBiztos(){}

//A felugyelo altal generalt veletlenszam eltarolasa
void Beallit_JavitoTanar_VeletlenSzam(mpz_t * Szam)
{
    _JavitoTanar_VeletlenSzam=new mpz_t[1];
    mpz_init_set(_JavitoTanar_VeletlenSzam[0],*Szam);
}

//Az azonosito es a veletlenszam
//hitelesiteset vegzo eljaras
mpz_t * Hitelesit(mpz_t * Hitelesitendo)
{
    mpz_powm(*Hitelesitendo,*Hitelesitendo,*_Titkos_Kulcs,*_Modulus);
    return Hitelesitendo;
}

//Az alapfajlbol masolatot keszit, es
//atadja a vizsgazonak
char * Atad_Dolgozat(char * Alap_Fajl)
{
    int i,Hossz=((((Globalis::AZONOSITOHOSSZ/8)+1) >= 1) ?                        \
    ((Globalis::AZONOSITOHOSSZ/8)+1) : 8);
    char * FajlNev=new char[Hossz];

    //A dolgozat neve egy veletlenszam lesz
    for(i=0;i<Hossz;i++)
```

```
        FajlNev[i]=(char)(((double)rand()/RAND_MAX)*10)+'0';

    FajlNev[i]='\0';

    FILE * be=fopen(Alap_Fajl,"r");
    FILE * ki=fopen(FajlNev,"w");

    //Az adatok atmasolasa
    while((i=fgetc(be))!=EOF) fputc(i,ki);

    fclose(be);
    fflush(ki);
    fclose(ki);

    //Az allomany neve lesz az atadando dolgozat
    return FajlNev;
}

//A vizsgabiztos pozicional a megfelelo helyekre es tarolja a megfelelo informaciokat
void Feljegyez_Eredmeny(char * FajlNev)
{
    char Buffer1[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    char Buffer2[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    char Buffer3[Globalis::KULCSMERET];

    FILE * in=fopen(FajlNev,"r");
    int ok=0,szamlal=0;

    while(fgets(Buffer1,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in))
    {
        Buffer1[strlen(Buffer1)-1]='\0';

        //A vizsgazo szemelyazonossaga (Sa)
        if(szamlal==4) strcpy(Buffer2,Buffer1);

        //A kapott erdemjegy
        if(szamlal==11) strcpy(Buffer3,Buffer1);

        //A javito tanar "azonositoja" es az eredmenyek "rogzitese"
        if(szamlal==13)

         _Eredmenyek=new Tablazat((unsigned char*)Buffer2,(unsigned char *)Buffer1, \
         (unsigned char*)Buffer3);

        if(!strcmp(Buffer1,"VALASZOK:")) ok=1;
        if(ok) szamlal++;
    }

    fflush(in);
    fclose(in);
}

//Ez a metodus eltavolitja a tanar "kezjegyet" es azonositojat a dolgozatrol
```

```
char * Eltavolit_Tanar_Azonosito(char * FajlNev)
{
    FILE * in=fopen(FajlNev,"r+");
    FILE * out=tmpfile();
    int szamlal=0;
    char Buffer[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];

        //A vizsgazo adatainak keresese
        while(fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in))
        {
            if(szamlal==6) break;
            Buffer[strlen(Buffer)-1]='\0';
            if(!strcmp(Buffer,Globalis::ELVALASZTO)) szamlal++;
            fprintf(out,"%s\n",Buffer);
        }

    fflush(in);
    fclose(in);

    remove(FajlNev);

    in=fopen(FajlNev,"w");
    fseek(out,0,0);

        //A vizsgazo adatainak visszairasa az eredeti allomanyba
        while(fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,out))
        {
            Buffer[strlen(Buffer)-1]='\0';
            fprintf(in,"%s\n",Buffer);
        }

    fflush(in);
    fclose(in);

    return FajlNev;
}

//A vizsgazo szemelyazonossaganak felfedeset vegzo metodus
unsigned char * Felfed_Szemelyazonossag(mpz_t * _Azonosito, mpz_t * _VeletlenSzam)
{
    unsigned char * VeletlenSzam=(unsigned char*)mpz_get_str(NULL,10,*_VeletlenSzam);
    unsigned char * Azonosito=(unsigned char*)mpz_get_str(NULL,2,*_Azonosito);
    unsigned char * Azonosito_10=(unsigned char *)mpz_get_str(NULL,10,*_Azonosito);
    unsigned char * Vissza=new unsigned char[Globalis::AZONOSITOHOSSZ+1];
    unsigned char temp[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    int i=0;

    //A vizsgabiztos osszeveti az adatbazisban tarolt
    //vizsgazo azonositojat a kapott azonositoval,
    //az esetleges visszaelesek megakadalyozasa erdekeben
    if(strcmp((char*)(*_Eredmenyek)._Vizsgazo_Azonosito,(char*)Azonosito_10))
    {
        return NULL;
    }
```

```
    //A vizsgazo adatainak "korrekcioja"
    for(i=0,temp[0]='0';Azonosito[i];i++)

        temp[i+1]=Azonosito[i];

        //A vizsgazo adatainak konverzioja
        Globalis::Atalakit_Kettes_Szo(temp,Vissza);

    //A vizsgazo adatainak kinyerese
    for(i=0;i<Globalis::AZONOSITOHOSSZ;i++)
    {
        if(i==0 || i==1 || i==2)
        {
            Vissza[i]=Vissza[i]^'0';
        }
        else Vissza[i]=Vissza[i]^VeletlenSzam[i-3];
    }

    Vissza[i]='\0';

    return Vissza;
}

//A vizsgabiztos digitalis alairasat elvegzo metodus
unsigned char * Digitalisan_Alair(char * FajlNev)
{
    //Teruletfogalalas a dolgozat szamara
    unsigned char Dolgozat[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];

    //A vizsgabiztos beolvassa az egesz dolgozatat egy tombbe,
    //majd abbol kesziti el a megfelelo lenyomatot
    int i=0;
    char Karakter;

    ifstream Fajl(FajlNev,ios::in);

        //A dolgozat beolvasasa
        while(Fajl.get(Karakter))

            Dolgozat[i++]=(unsigned char)Karakter;

        Dolgozat[i]='\0';

    //A vizsgabiztos lenyomatot keszit
    SHA1 Lenyomat(Dolgozat);

    FILE * out=fopen(FajlNev,"r+");

    while(fgets((char*)Dolgozat,Globalis::AZONOSITOHOSSZ,out));

    mpz_t _lenyomat;
    mpz_t _temp;
```

```
            mpz_init(_temp);
            mpz_init(_lenyomat);

        //A lenyomat titkositasa
        mpz_set_str(_lenyomat,(char*)Lenyomat.sha1_alairas(),16);
        mpz_powm(_temp,_lenyomat,*_Titkos_Kulcs,*_Modulus);

        //A lenyomat tarolasa a kesobbi kiirathatosaghoz
        unsigned char * Vissza=(unsigned char*)mpz_get_str(NULL,10,_temp);

        //A vizsgazo hozzafuzi a lenyomatat a dolgozathoz
        fprintf(out, "%s\n", Vissza);
        fprintf(out, "%s\n", Globalis::ELVALASZTO);

        mpz_clear(_lenyomat);
        mpz_clear(_temp);

        fflush(out);
        fclose(out);

        return Vissza;
    }
};
```

1.7. Vizsgazo.h

```
class Vizsgazo
{
    private :

        //A vizsgazo Vernam-kodolassal kodolt szemelyazonossagat tarolo attribuitum
        mpz_t * _Azonosito;

        //A vizsgazo Vernam-kodolasnal felhasznalt veletlenszamat tarolo attributum
        mpz_t * _VeletlenSzam;
        const mpz_t * _Titkos_Kulcs;

    //Ez az eljaras egy fajlbol eloallitja a szukseges
    //Sa es vsz1 adatokat a hitelesites elokeszitesehez
    void Azonosito_Elkeszites(char * FajlNev, const mpz_t * vNyilvanos_Kulcs,          \
    const mpz_t * vModulus)
    {

        //A vizsgazo termeszetes azonositojanak eloallitasa
        unsigned char Buffer[Globalis::AZONOSITOHOSSZ+1];
        Globalis::Adatok_Kiolvasasa_Kitoltessel(FajlNev,Buffer);

        //Veletlenszam eloallitasa
        unsigned char VeletlenSzam[Globalis::AZONOSITOHOSSZ+1];
        Globalis::VeletlenSzam_Eloallitas(VeletlenSzam);

        //A Buffer es a VeletlenSzam Vernam-kodolasa
        unsigned char Vernam_Buffer[Globalis::AZONOSITOHOSSZ+1];
        Globalis::Vernam_Kodolas(Vernam_Buffer,Buffer,VeletlenSzam);
```

```
    //Az Azonosito es VeletlenSzam tombok inicializalasa
    _Azonosito=new mpz_t[2];
    _VeletlenSzam=new mpz_t[2];

    mpz_init(_Azonosito[0]); mpz_init(_Azonosito[1]);
    mpz_init(_VeletlenSzam[0]); mpz_init(_VeletlenSzam[1]);

    //Az Azonosito es VeletlenSzamok beallitasa
    unsigned char Vernam_Buffer_Kettes[(Globalis::AZONOSITOHOSSZ*8)+1];
    Globalis::Atalakit_Szo_Kettes(Vernam_Buffer,Vernam_Buffer_Kettes);

    mpz_set_str(_Azonosito[0],(char*)Vernam_Buffer_Kettes,2);

    if(Globalis::MEGJELENIT) gmp_printf("\n[ A vizsgazo altal                          \
    generalt azonosito : ]\n\n%Zd\n\n",_Azonosito[0]);

    mpz_set_str(_VeletlenSzam[0],(char*)VeletlenSzam,10);

    if(Globalis::MEGJELENIT) gmp_printf("[ A vizsgazo altal                          \
    generalt veletlenszam : ]\n\n%Zd\n\n",_VeletlenSzam[0]);

    //Az Sa azonosito vakitasa a hitelesiteshez
    Globalis::Vakitas(&_Azonosito[0],&_Azonosito[1], &_VeletlenSzam[0],             \
    vNyilvanos_Kulcs, vModulus);

    if(Globalis::MEGJELENIT) gmp_printf("[ A vizsgazo altal                          \
    vakitott azonosito : ]\n\n%Zd\n\n",_Azonosito[1]);

    //A vsz_1 elem vakitasa a hitelesiteshez
    Globalis::Vakitas(&_VeletlenSzam[0],&_VeletlenSzam[1],&_VeletlenSzam[0],        \
    vNyilvanos_Kulcs,vModulus);

    if(Globalis::MEGJELENIT) gmp_printf("[ A vizsgazo altal                          \
    vakitott veletlenszam : ]\n\n%Zd\n\n",_VeletlenSzam[1]);
}

public :

    const mpz_t * _Modulus;
    const mpz_t * _Nyilvanos_Kulcs;

//A vizsgazo objektumot letrehozo konstruktor
Vizsgazo(char * Adatok, const mpz_t * vNyilvanos_Kulcs, const mpz_t * vModulus)
{
    //A vizsgazo reszere elkesziti a megfelelo kulcsokat
    RSA_Kulcs_Generator Generator(Globalis::KULCSMERET);

    _Modulus=Generator.Exportal_Modulus();
    if(Globalis::MEGJELENIT) gmp_printf("\n[ A vizsgazo                               \
    modulusa : ]\n\n%Zd\n",_Modulus);
    _Nyilvanos_Kulcs=Generator.Exportal_Nyilvanos_Kulcs();
    if(Globalis::MEGJELENIT) gmp_printf("\n[ A vizsgazo                               \
    nyilvanos kulcsa : ]\n\n%Zd\n",_Nyilvanos_Kulcs);
```

```
    _Titkos_Kulcs=Generator.Exportal_Titkos_Kulcs();
    if(Globalis::MEGJELENIT) gmp_printf("\n[ A vizsgazo                                \
    titkos kulcsa : ]\n\n%Zd\n",_Titkos_Kulcs);

    //Az S_{a} es vsz_{1} eloallitasa
    Azonosito_Elkeszites(Adatok,vNyilvanos_Kulcs,vModulus);
}

~Vizsgazo(){}

mpz_t * Atad_Azonosito_Hitelesitesre(void)
{
    return &_Azonosito[1];
}

mpz_t * Atad_Azonosito(void)
{
    return _Azonosito;
}

mpz_t * Atad_VeletlenSzam_Hitelesitesre(void)
{
    return &_VeletlenSzam[1];
}

mpz_t * Atad_VeletlenSzam(void)
{
    return _VeletlenSzam;
}

//A vizsgazo a vizsgabiztos altal hitelesitett azonositojat beallito metodus
void Hitelesitett_Azonosito_Beallitas(const mpz_t * vModulus)
{
    mpz_t temp;
    mpz_init(temp);

        if(mpz_invert(temp,_VeletlenSzam[0],*vModulus)==0)
        {
            printf("\n\t ! ! ! A hitelesitett azonosito multiplikativ              \
            inverze nem letezik ! ! !\n\n");
            exit(-1);
        }
        mpz_mul(temp,_Azonosito[1],temp);
        mpz_powm_ui(_Azonosito[1],temp,(unsigned long)1,*vModulus);

    mpz_clear(temp);
}

//A vizsgazo a vizsgabiztos altal hitelesitett veletlenszamat beallito metodus
void Hitelesitett_VeletlenSzam_Beallitas(const mpz_t * vModulus)
{
    mpz_t temp;
    mpz_init(temp);
```

```
        if(mpz_invert(temp,_VeletlenSzam[0],*vModulus)==0)
        {
            printf("\n\t! ! ! A hitelesitett veletlenszam multiplikativ            \
            inverze nem letezik ! ! !\n\n");
            exit(-1);
        }
        mpz_mul(temp,_VeletlenSzam[1],temp);
        mpz_powm_ui(_VeletlenSzam[1],temp,(unsigned long)1,*vModulus);

    mpz_clear(temp);
}

//A vizsgazo dolgozatanak "megirasat" elvegzo eljaras
char * Megir_Dolgozat(char * FajlNev)
{
    char * Valaszok="dab";
    char Buffer[Globalis::AZONOSITOHOSSZ+1];
    FILE * Fajl=fopen(FajlNev,"r+");

    //Rairja a dolgozatra a valaszokat
    while(fgets(Buffer,Globalis::AZONOSITOHOSSZ,Fajl))
    {
        Buffer[strlen(Buffer)-1]='\0';

        if(!strcmp(Buffer,"VALASZOK:"))

            fprintf(Fajl,"%s\n%s\n%s\n",Globalis::ELVALASZTO,                    \
            Valaszok,Globalis::ELVALASZTO);
    }

        //Ezt kovetoen hozzafuzi a dolgozatahoz a hitelesitett azonositojat
        fprintf(Fajl,"%s\n%s\n", mpz_get_str(NULL,10,_Azonosito[0]),             \
        mpz_get_str(NULL,10,_Azonosito[1]));
        fprintf(Fajl, "%s\n", Globalis::ELVALASZTO);

    fflush(Fajl);
    fclose(Fajl);

    return FajlNev;
}

//A vizsgazo digitalis alairasat elvegzo metodus
unsigned char * Digitalisan_Alair(char * FajlNev)
{
    //Teruletfogalalas a dolgozat szamara
    unsigned char Dolgozat[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];

    //A vizsgazo beolvassa az egesz dolgozatat egy tombbe,
    //majd abbol kesziti el a megfelelo lenyomatot
    int i=0;
    char Karakter;

    ifstream Fajl(FajlNev,ios::in);
```

```
        //A dolgozat beolvasasa
        while(Fajl.get(Karakter))

            Dolgozat[i++]=(unsigned char)Karakter;

        Dolgozat[i]='\0';

    //A vizsgazo lenyomatot keszit
    SHA1 Lenyomat(Dolgozat);

    FILE * out=fopen(FajlNev,"r+");

    while(fgets((char*)Dolgozat,Globalis::AZONOSITOHOSSZ,out));

    mpz_t _lenyomat;
    mpz_t _temp;

        mpz_init(_temp);
        mpz_init(_lenyomat);

    //A lenyomat titkositasa
    mpz_set_str(_lenyomat,(char*)Lenyomat.sha1_alairas(),16);
    mpz_powm(_temp,_lenyomat,*_Titkos_Kulcs,*_Modulus);

    //A lenyomat tarolasa a kesobbi kiirathatosaghoz
    unsigned char * Vissza=(unsigned char*)mpz_get_str(NULL,10,_temp);

    //A vizsgazo hozzafuzi a lenyomatat a dolgozathoz
    fprintf(out, "%s\n", Vissza);
    fprintf(out, "%s\n", Globalis::ELVALASZTO);

    mpz_clear(_lenyomat);
    mpz_clear(_temp);

    fflush(out);
    fclose(out);

    return Vissza;
}

//A vizsgazo ellenorzi a kapott erdemjegyet
int Ellenoriz_Erdemjegy(char * FajlNev)
{
    FILE * in=fopen(FajlNev,"r");
    char Buffer[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];
    int szamlal=0;

    while(fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in))
    {
        Buffer[strlen(Buffer)-1]='\0';
        if(!strcmp(Buffer,Globalis::ELVALASZTO)) szamlal++;
        if(szamlal==5) break;
    }
```

```
        fgets(Buffer,Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET,in);
        Buffer[strlen(Buffer)-1]='\0';

        //Az elegedettseg ellenorzese
        for(int i=2;i;i++)

            if(!strcmp(Buffer,Globalis::ERDEMJEGYEK[i])) return 1;

        fflush(in);
        fclose(in);

        return 0;
    }
};
```

1.8. JavitoTanar.h

```
class JavitoTanar
{
    private :

        //A javito tanar Vernam-kodolassal elfedett szemelyazonossagat tarolo attributum
        mpz_t * _Azonosito;

        //A javito tanar szemelyazonossagat tarolo attributum
        unsigned char * _T;
        const mpz_t * _Titkos_Kulcs;

    public :

        const mpz_t * _Modulus;
        const mpz_t * _Nyilvanos_Kulcs;

    //A javito tanar objektumot letrehozo konstruktor
    JavitoTanar()
    {
        //A javito tanar reszere elkesziti a megfelelo kulcsokat
        RSA_Kulcs_Generator Generator(Globalis::KULCSMERET);

        _Modulus=Generator.Exportal_Modulus();
        if(Globalis::MEGJELENIT) gmp_printf("\n[ A javito tanar                          \
        modulusa : ]\n\n%Zd\n",_Modulus);
        _Nyilvanos_Kulcs=Generator.Exportal_Nyilvanos_Kulcs();
        if(Globalis::MEGJELENIT) gmp_printf("\n[ A javito tanar                          \
        nyilvanos kulcsa : ]\n\n%Zd\n",_Nyilvanos_Kulcs);
        _Titkos_Kulcs=Generator.Exportal_Titkos_Kulcs();
        if(Globalis::MEGJELENIT) gmp_printf("\n[ A javito tanar                          \
        titkos kulcsa : ]\n\n%Zd\n",_Titkos_Kulcs);
    }

    ~JavitoTanar(){}
```

```
//A javito tanar szemelyazonosagat beallito metodus
void Beallit_T(unsigned char * T)
{
    _T=new unsigned char[Globalis::AZONOSITOHOSSZ+1];
    strcpy((char*)_T,(char*)T);
}

//A javito tanar kodolt es alairt azonositojat beallito metodus
void Beallit_Azonosito(mpz_t * Azonosito)
{
    _Azonosito=new mpz_t[2];
    mpz_init_set(_Azonosito[0],Azonosito[0]);
    mpz_init_set(_Azonosito[1],Azonosito[1]);
}

mpz_t * Atad_Azonosito(void)
{
    return _Azonosito;
}

//A javito tanar azonositojat hitelesito metodus
void Hitelesitett_Azonosito_Beallitas(const mpz_t * vModulus, mpz_t * VeletlenSzam, \
mpz_t * Ezt)
{
    mpz_t temp;
    mpz_init(temp);

        if(mpz_invert(temp,*VeletlenSzam,*vModulus)==0)
        {
            printf("! ! ! A hitelesitett azonosito multiplikativ                    \
            inverze nem letezik! ! !\n");
            exit(-1);
        }
        mpz_mul(temp,*Ezt,temp);
        mpz_powm_ui(*Ezt,temp,(unsigned long)1,*vModulus);

    mpz_clear(temp);
}

//A vizsgazo dolgozatanak osztalyzasat elvegzo metodus
char * Leosztalyoz_Dolgozat(char * FajlNev)
{
    char Buffer[Globalis::AZONOSITOHOSSZ+1];
    const char * Erdemjegy;
    FILE * Fajl=fopen(FajlNev,"r+");
    int i=0,megvan=0,osszpontszam=0;

    //Megkeresi a dolgozat valaszait
    while(fgets(Buffer,Globalis::AZONOSITOHOSSZ,Fajl))
    {
        Buffer[strlen(Buffer)-1]='\0';

        if(!strcmp(Buffer,"VALASZOK:")) megvan=1;
```

```
        if(megvan && !strcmp(Buffer,Globalis::ELVALASZTO))
        {
            fgets(Buffer,Globalis::AZONOSITOHOSSZ,Fajl);
            break;
        }

    }

    //A valaszok ellenorzese
    for(i=0;i<Globalis::KERDESEKSZAMA;i++)

        if(Globalis::HELYESVALASZOK[i]==Buffer[i]) osszpontszam++;

    double szazalek=(double)osszpontszam/Globalis::KERDESEKSZAMA;

    if(szazalek>=0.9) Erdemjegy=Globalis::ERDEMJEGYEK[5];
    else if(szazalek<0.9 && szazalek>=0.8) Erdemjegy=Globalis::ERDEMJEGYEK[4];
    else if(szazalek<0.8 && szazalek>=0.7) Erdemjegy=Globalis::ERDEMJEGYEK[3];
    else if(szazalek<0.7 && szazalek>=0.6) Erdemjegy=Globalis::ERDEMJEGYEK[2];
    else Erdemjegy=Globalis::ERDEMJEGYEK[1];

    //A dolgozat vegere pozicional
    while(fgets(Buffer,Globalis::AZONOSITOHOSSZ,Fajl));

    //Az erdemjegy beirasa
    fprintf(Fajl, "ERDEMJEGY:\n");
    fprintf(Fajl, "%s\n", Globalis::ELVALASZTO);
    fprintf(Fajl, "%s\n", Erdemjegy);
    fprintf(Fajl, "%s\n", Globalis::ELVALASZTO);
    fprintf(Fajl,"%s\n%s\n", mpz_get_str(NULL,10,_Azonosito[0]),                    \
    mpz_get_str(NULL,10,_Azonosito[1]));
    fprintf(Fajl, "%s\n", Globalis::ELVALASZTO);

    fflush(Fajl);
    fclose(Fajl);

    return FajlNev;
}

//A javito tanar digitalis alairasat elvegzo metodus
unsigned char * Digitalisan_Alair(char * FajlNev)
{
    unsigned char Dolgozat[Globalis::AZONOSITOHOSSZ*Globalis::KULCSMERET];

    //A javito tanar beolvassa az egesz dolgozatot egy tombbe,
    //majd abbol kesziti el a megfelelo alairast
    int i=0;
    char Karakter;

    ifstream Fajl(FajlNev,ios::in);

        while(Fajl.get(Karakter))

            Dolgozat[i++]=(unsigned char)Karakter;
```

```
            Dolgozat[i]='\0';

        //A javito tanar elkesziti a lenyomatot
        SHA1 Lenyomat(Dolgozat);

        //A javito tanar elokeszul a dolgozat hozzairasahoz
        FILE * out=fopen(FajlNev,"r+");
        while(fgets((char*)Dolgozat,Globalis::AZONOSITOHOSSZ,out));

        mpz_t _lenyomat;
        mpz_t _temp;

            mpz_init(_temp);
            mpz_init(_lenyomat);

        //A javito tanar kodolja a lenyomatot
        mpz_set_str(_lenyomat,(char*)Lenyomat.sha1_alairas(),16);
        mpz_powm(_temp,_lenyomat,*_Titkos_Kulcs,*_Modulus);

        //A kodolt lenyomat tarolasa, a kesobbi kiiratasahoz
        unsigned char * Vissza=(unsigned char*)mpz_get_str(NULL,10,_temp);

        //Elhelyezi a digitalis alairasat
        fprintf(out, "%s\n", Vissza);
        fprintf(out, "%s\n", Globalis::ELVALASZTO);

        mpz_clear(_lenyomat);
        mpz_clear(_temp);

        fflush(out);
        fclose(out);

        return Vissza;
    }
};
```

1.9. Felugyelo.h

```
class Felugyelo
{
    public :

    //Ez a metodus beallitja a javito tanar reszere
    //a szukseges attributumokat, es visszaadja
    //a kodolashoz hasznalt veletlenszamot
    mpz_t * Tanari_Azonosito_Elkeszites(char * FajlNev, JavitoTanar * Javito_Tanar,     \
    VizsgaBiztos * Vizsga_Biztos)
    {
        mpz_t * Vissza=new mpz_t[1];
        mpz_init(Vissza[0]);

        //A javito tanar termeszetes azonositojanak eloallitasa
        unsigned char Buffer[Globalis::AZONOSITOHOSSZ+1];
```

```
        Globalis::Adatok_Kiolvasasa_Kitoltessel(FajlNev,Buffer);
        (*Javito_Tanar).Beallit_T(Buffer);

        //Veletlenszam eloallitasa
        unsigned char VeletlenSzam[Globalis::AZONOSITOHOSSZ+1];
        Globalis::VeletlenSzam_Eloallitas(VeletlenSzam);
        mpz_init_set_str(Vissza[0],(char*)VeletlenSzam,10);

        //A Buffer Vernam kodolasa a VeletlenSzammal
        unsigned char Vernam_Buffer[Globalis::AZONOSITOHOSSZ+1];
        Globalis::Vernam_Kodolas(Vernam_Buffer,Buffer,VeletlenSzam);

        //A javito tanar azonositojanak inicializalasa
        mpz_t * temp=new mpz_t[2];

        mpz_init(temp[0]);
        mpz_init(temp[1]);

        //Az Azonosito es VeletlenSzamok beallitasa
        unsigned char Vernam_Buffer_Kettes[(Globalis::AZONOSITOHOSSZ*8)+1];
        Globalis::Atalakit_Szo_Kettes(Vernam_Buffer,Vernam_Buffer_Kettes);
        mpz_init_set_str(temp[0],(char*)Vernam_Buffer_Kettes,2);

        //A javito tanar azonositojanak vakitasa
        Globalis::Vakitas(&temp[0],&temp[1],Vissza,(*Vizsga_Biztos)._Nyilvanos_Kulcs,   \
        (*Vizsga_Biztos)._Modulus);

        //A javito tanar azonositojanak hitelesitese
        (*Vizsga_Biztos).Hitelesit(&temp[1]);

        //A javito tanar azonositojanak beallitasa
        (*Javito_Tanar).Hitelesitett_Azonosito_Beallitas((*Vizsga_Biztos)._Modulus,     \
        Vissza,&temp[1]);

        (*Javito_Tanar).Beallit_Azonosito(temp);

        if(Globalis::MEGJELENIT) gmp_printf("\n[ A felugyelo altal generalt            \
        azonosito a javito tanar reszere : ]\n\n%Zd\n",temp);

        mpz_clear(temp[0]);
        mpz_clear(temp[1]);

        return Vissza;
    }
};
```

2. A program futásához szükséges adatállományok

2.1. Kiss_Peter.dat

```
M01
Matematika 1.
2009052711
120
11982042135 81
Kiss Péter
Debrecen
19810523
Nagy Viola
Magyar
```

2.2. Nagy_Peter.dat

```
M01
Matematika 1.
20091012
11982042135 81
Nagy Péter
Budapest
19510224
Kiss Viola
Magyar
```

2.3. Dolgozat.dat

```
Matematika 1.
2009. majus 27., 11 ora

1. Kerdes:
Mely allitasok igazak az 1-re.

    a) Eleme az egesz szamok halmazanak
    b) Eleme a valos szamok halmazanak
    c) Nem eleme egyik halmaznak sem
    d) Mindkettonek eleme

2. Kerdes:
Igaz-e a kovetkezo allitas? Az A es B halmaz egyenlo, ha elemeik megegyeznek?

    a) Igaz
    b) Hamis

3. Kerdes:
Igaz-e, hogy a magyar kartya elemeinek 31! kulonbozo sorrendje letezik?

    a) Igaz
    b) Hamis

VALASZOK:
```

3. A program fordítását végző Makefile

```
CFILES=diplomamunka.cpp

LIBS=-lgmpxx -lgmp

CC= g++ -Wall -O3

diplomamunka:  diplomamunka.cpp
	$(CC) -o $@ $< $(OBJS) $(LIBS)
	@exit 0
clean:
	rm -f *.o [0-9]* *.txt *~ diplomamunka
	@exit 0

all:	clean diplomamunka
	@exit 0
```

4. A program egy lehetséges futási eredménye

```
A VIZSGABIZTOS LETREHOZASA...

[ A generalt p: ]

888279968781159316415616530615931560429173542492720671187142156275233496469490635381847 3\
675416520087510674423496332040745649927668577992481012108967068267

[ A generalt q: ]

121410381354464100291255138067490576451850651710380849547514356771574366054277537106227 7\
245003869970220614862731104536572924379451867605989256691534982367 7

[ A vizsgabiztos modulusa : ]

107846409759252018168477396331797930537817510525607699092556577957939884615821726759270 9\
226049605380377210586277553005017481110027610866817630817519614799723034442539388216272 8\
063866646717408497761632543929893639586645388495612818511005043367373968782663516027189 1\
781874612731878692380885721310970798071957759

[ A vizsgabiztos nyilvanos kulcsa : ]

65537

[ A vizsgabiztos titkos kulcsa : ]

253978895619923958742737710756514527659287952981099108563791175092366781994994053863098 3\
138509484633221213694341784500273098166252063127464381042403182143778399141936810888063 7\
501590698788960360095670670031100014901875406152226515340236565712922441766544891804467 8\
822709291143980656723455632448800621959455 85

A FELUGYELO LETREHOZASA...

A JAVITOTANAR LETREHOZASA...

[ A generalt p: ]

120991386860618579067331850187608693267259210372479332435653525270797376710597065667903 0\
528571364647371954415600301197942847290812569911167689005023939706 1

[ A generalt q: ]

677284446766091631052340857069256476121244078809776473604495340418974712028616933852546 3\
040526207486311116899619250117027519291211695570756919089501751969

[ A javito tanar modulusa : ]

819455845133562224442610986453470188862674290027942588591661671246140197896595957397342 4\
810269113489322438053111521160324209899570719400939403166024873913673930178565924986843 4\
951908657035367726070846061475873610583394593643120045523229152544608393772264784862572 2\
846902648127680562505188998255238257295631 09
```

[A javito tanar nyilvanos kulcsa :]

65537

[A javito tanar titkos kulcsa :]

304352880455254865269353098574300286358947692045259174953226982312316684575141404077219 1\
789474045065285220023052436586408465342714052839438271700935524297523549144483216738369 6\
727342535283256043343840475498569019954433904105129604316882211742549764385392313075348 5\
371647168160318771552307334106188438842789 13

A FELUGYELO BEALLITJA A JAVITO TANAR AZONOSITOJAT...

[A felugyelo altal generalt azonosito a javito tanar reszere :]

342882804409782600393849952096289980641592458618522310463745335500557745826154246806152 2\
475551548300890053427937397304118209935249401084116967775326845738027717412229607520553 2\
769262552826877247027879254282744310931327688576315938665183381090445670143452266678069 9\
581635580523922542768393912057801726036226

[A felugyelo altal generalt veletlenszam a javito tanar reszere :]

817277819934932936134359731558826736639139542489613734622184858421445497486191722294216 6\
464986322822561718841452513169658281

[A javito tanar azonositoja ervenyes!]

A VIZSGAZO LETREHOZASA...

[A generalt p:]

126518974640728175492087461447275512970986049192517335066759185898018866573801979276485 1\
365470403618509379390524897574724324187787950738315708106779095432 1

[A generalt q:]

764364374294724113329660020722067057153094985670599797960505524318475544635568968732568 1\
572292120875867063110797370216029659592607105704337479195705897603

[A vizsgazo modulusa :]

967065968876702593845379274202068239112093124874560302597325158876761634062739791765478 4\
241879278345613040975386858204917332431445996919457689783090534869430099883166604518628 6\
185527165100087916775117507315359462958517971233917452034583548561037864611542691363089 5\
094875849065673417817647747631667552763925 63

[A vizsgazo nyilvanos kulcsa :]

65537

[A vizsgazo titkos kulcsa :]

463265576588554220269095050331475843674934215106532503777149142666523208285391698772864 1\
092418022546966162952565274781320164970707342009649116846058369198499170456575309743024 8\

```
553426399345281207902141125611268421630669470547367398087097035143729987871407814352780\5
113955660342850607948378354486458154846678 73

[ A vizsgazo altal generalt azonosito : ]

342882806533992854016300682969020774470963383559410572398306442816307408715582609334046\7
546813380439531963627902588661378087957203227401777612989605902837895532017889654689791\6
068644171072240020466447885908054779761872178771609439007400433362217638356495991865762\0
334086600176388625159306816554599453556999

[ A vizsgazo altal generalt veletlenszam : ]

574151295366332925637855936523434479977223586361855515759566793423446334525776536527197\7
661946417741733111764427147824525471

[ A vizsgazo altal vakitott azonosito : ]

870145937623256410840130054038073277313808793171446693999039009787279959526712579868866\0
023662374728083429180852701454331802918397428374019244700457640517337661125849468717514\8
461704597775560398078759105193191395930692127188966943269305845102372900872494222178948\7
646351725383006321529205594463925436616 42888

[ A vizsgazo altal vakitott veletlenszam : ]

789179000617922867393796308118865173305991705673616078084092502600862250904982098632860\8
594024407015328243612233816194849377306235018370866361672810587682428102460776635532495\6
235375922289665368648316349850082344085565143434778617922580246705132064375141159539947\2
663952630418827271809440148068266586105330 35

        [ A vizsgazo vakitott azonositojanak atadasa hitelesitesre... ]

        [ A vizsgazo a vizsgabiztos altal hitelesitett azonositojanak beallitasa... ]

                [ A vizsgazo azonositoja ervenyes! ]

        [ A vizsgazo vakitott veletlenszamanak atadasa hitelesitesre... ]

        [ A vizsgazo a vizsgabiztos altal hitelesitett veletlenszamanak beallitasa... ]

                [ A vizsgazo veletlenszama ervenyes! ]

A DOLGOZAT ATADASA A VIZSGAZO RESZERE...

        [ A dolgozat megirasa... ]

        [ A dolgozatnak a vizsgazo altal alairt lenyomata : ]

772942442995933728035528289727314669585978089582798689789569626128209499155351682798663\5
064931551940437741843081093974184288169192338503582286265733824208933595202933215063113\7
053335531659018647743906857981817829136883738611164712395502425635816999085160416420694\5
868041070516510071479667860317046009359181 19
```

```
A MEGIRT ES ALAIRT DOLGOZAT ATADASA A JAVITO TANAR RESZERE...

        [ A javito tanar ellenorzi a dolgozat digitalis alairasat... ]

                [ Az alairas rendben! ]

        [ A dolgozat osztalyzasa... ]

        [ A javito tanar digitalisan alairja a dolgozatot... ]

        [ A dolgozatnak a javito tanar altal alairt lenyomata : ]

467908435795056291031422045743609281591690062508462523839337463592204589217112946520763 7\
118840433780879213885726318537734635932015154328893116691618243587960458116970400247955 4\
853316648413114638449802793896543866840239259577960360308456903040939608797979204641684 8\
536515212192897387189997196650853190458463 89

A JAVITOTT ES ALAIRT DOLGOZAT ATADASA A VIZSGABIZTOS RESZERE...

        [ A vizsgabiztos ellenorzi a dolgozat vizsgazo altal vegzett
          digitalis alairasat... ]

                [ Az alairas rendben! ]

        [ A vizsgabiztos ellenorzi a dolgozat javito tanar altal vegzett
          digitalis alairasat... ]

                [ Az alairas rendben! ]

        [ A vizsgabiztos ellenorzi a vizsgazo azonositojanak ervenyesseget... ]

                [ Az azonosito rendben! ]

        [ A vizsgabiztos ellenorzi a javito tanar azonositojanak ervenyesseget... ]

                [ Az azonosito rendben! ]

        [ A vizsgabiztos ellenorzi, hogy a megfelelo dolgozatot adta be a vizsgazo... ]

                [ A vizsgazo ervenyes dolgozatot adott be! ]

        [ A vizsgabiztos ellenorzi, hogy a javito tanar jo dolgozatot javitott-e... ]

                [ A javito tanar ervenyes dolgozatot javitott! ]

        [ A vizsgabiztos ellenorzi, hogy a ket dolgozat egyezik-e... ]

                [ A ket dolgozat egyezik! ]

        [ A vizsgabiztos feljegyzi a vizsgazo es a javito tanar azonositojat,
          es a jegyet... ]

        [ A vizsgabiztos eltavolitja a javito tanar alairasat es azonositojat... ]
```

```
        [ A vizsgabiztos digitalisan alairja a dolgozatot... ]

        [ A vizsgabiztos digitalis alairasa... ]

4452379242486090563670170873880414893467338975893708012541108462103372992272879729181529\
9452198529966673177924431835806732672383855030128368664730296683399122396409002166769749\
5139043883843157756756991000524184777339464570381369451784252076400200426958539736408985\
6291188060319502182993032681485562591373709

A VIZSGABIZTOS ELKULDI A DOLGOZATOT A VIZSGAZO RESZERE...

        [ A vizsgazo ellenorzi alairasanak valtozatlansagat... ]

                [ Az alairas rendben! ]

        [ A vizsgazo ellenorzi a vizsgabiztos alairasat... ]

                [ Az alairas rendben! ]

        [ A vizsgazo ellenorzi a kapott erdemjegyet...]

                [ A vizsgazo elegedett a javitassal... ]

        [ A vizsgazo atadja a veletlenszamat es azonositojat a vizsgabiztosnak... ]

A VIZSGABIZTOS FELFEDI A VIZSGAZO SZEMELYET...

        [ A vizsgazo szemelye... ]

**********
M01
Matematika 1.
2009052711
120
11982042135 81
Kiss Péter
Debrecen
19810523
Nagy Viola
Magyar
6962585175212341327669271111818 66
**********

        [ A program futasa szabalyosan veget ert. ]
```

5. Egy, a program által előállított dolgozat

```
Matematika 1.
2009. majus 27., 11 ora

1. Kerdes:
Mely allitasok igazak az 1-re.

    a) Eleme az egesz szamok halmazanak
    b) Eleme a valos szamok halmazanak
    c) Nem eleme egyik halmaznak sem
    d) Mindkettonek eleme

2. Kerdes:
Igaz-e a kovetkezo allitas? Az A es B halmaz egyenlo, ha elemeik megegyeznek?

    a) Igaz
    b) Hamis

3. Kerdes:
Igaz-e, hogy a magyar kartya elemeinek 31! kulonbozo sorrendje letezik?

    a) Igaz
    b) Hamis

VALASZOK:
**********
dab
**********
342882806533992854016300682969020774470963383559410572398306442816307408715582609334046\
7
546813380439531963627902588661378087957203227401777612989605902837895532017889654689791\
6
068644171072240020466447885908054779761872178771609439007400433362217638356495991865762\
0
334086600176388625159306816554599453556999
615335681754240679802098368905280385309062287000803791473792549908279253800820101828624\
6
880885619719054637096844926160744780075951325228238862223234566320614186811413296552843\
4
620422794627925472844992898874689007431497406784090677167416785253576629439560033858746\
4
703677836214167434367844416905230317359245\
1
**********
772942442995933728035528289727314669585978089582798689789569626128209499155351682798663\
5
064931551940437741843081093974184288169192338503582286265733824208933595202933215063113\
7
053335531659018647743906857981817829136883738611164712395502425635816999085160416420694\
5
868041070516510071479667860317046009359181\
19
**********
ERDEMJEGY:
**********
JELES
**********
445237924248609056367017087388041489346733897589370801254110846210337299227287972918152\
9
945219852996667317792443183580673267238385503012836866473029668339912239640900216676974\
9
513904388384315775675699100052418477733946457038136945178425207640020042695853973640898\
5
629118806031950218299303268148556259137370\
9
**********
```