Projekt pri predmetu Kriptografija in teorija kodiranja 2

Kriptosistem XTR z javnimi ključi

4. letnik IŠRM

Nejc Trdin (63070088)
Mentor: prof. dr. Aleksandar Jurišić

10. julij 2011

Povzetek

V projektu bom predstavil kriptosistem XTR z javnimi ključi. Sistem je podprt z novo metodo predstavitve elementov podgrupe v multiplikativni grupi znotraj končnega obsega. Z uporabo sistema v aplikativnih rešitvah bistveno povečamo hitrost izračunov in komunikacije med klienti, ne da bi zmanjšali nivo varnosti.

1 Uvod

Diffie-Hellman (DH) protokol o določanju skupnega ključa je bil prvi praktični protokol, preko katerega sta se lahko dva klienta, ki se nikoli prej nista srečala, dogovorila o skupnem ključu preko javnega kanala. S predpostavko, da oba klienta poznata praštevilo p in generator g v multiplikativni grupi $\mathrm{GF}(p)^*$, vsak izmed klientov pošlje drugemu približno $\log_2(p)$ bitov.

Pred iznajdbo XTR kriptosistema, je bila predlagana varianta osnovne DH sheme[3], kjer je g generiral majhno podgrupo v $\mathrm{GF}(p)^*$ reda q. S tem so bistveno pridobili na ceni izračunov v DH shemi, vendar ni bilo nobene spremembe v količini izmenjanih bitov. Kasneje so pokazali[4], kako uporabiti končna razširjena polja v povezavi s podgrupami, da se število prenesenih bitov zmanjša za faktor 3. Pokazano je bilo, da lahko elemente praštevilskega reda q podgrupe v $\mathrm{GF}(p^6)^*$ predstavimo z $2\log_2(p)$ bitov, če q deli $p^2 - p + 1$. Ampak kljub zmanjšanemu številu prenesenih bitov je metoda za izračun elementov uporabljala počasne operacije.

V projektni nalogi bom predstavil sistem, ki ima iste komunikacijske doprinose kot v [4], vendar z bistveno hitrejšim izračunavanjem. Metoda ima ime *XTR - Efficient and Compact Subgroup Trace Representation*, katero sta razvila Arjen K. Lenstra in Eric R. Verheul in jo opisala v [1].

Kriptosistem XTR lahko uporabimo v povezavi s poljubnim kriptografskim protokolom, ki temelji na podgupah, in z njim pridobimo veliko pri komunikaciji med klienti in pri hitrosi izračunov. Poleg tega bomo v poglavju 6 pokazali, da z uporabo XTR sistema ne zmanjšamo varnosti celotnega kriptosistema.

Varnost kriptosistema XTR, pri izbranih praštevilih p in q velikosti $1024/6 \approx 170$bitov, je ekvivalentna tradicionalnim sistemom, ki temeljijo na podgrupah s 170bitnimi praštevili in 1024 bitnimi končnimi obsegi. Ampak elementi v XTR podgrupi so predstavljeni z zgolj $2 \cdot 170$ biti, kar je bistveno manj kot 1024 bitov za tradicionalno predstavitev.

Sistem XTR je kot alternativa RSA in eliptičnim krivuljam uporaben v aplikacijah kot so SSL/TLS, pametne kartica, WAP in IPSEC.

V poglavju 2 bomo predstavili računanje operacij med elementi v sistemu XTR, ki jih zahtevajo moderni kriptosistemi. Poglavje 3 je namenjeno algoritmom za pravilno in hitro izbiro parametrov sistema XTR. Uporabo sistema XTR v povezavi z Diffie-Hellman-ovo izmenjavo ključa in z ElGamalovim šifriranjem si bomo pogledali v poglavju 4. Poglavji 5 in 6, sta pa zaporedoma namenjeni primerjanju sistema XTR z ECC in RSA, ter varnosti XTR.

2 Predstavitev podgrup in računanje v njih

XTR je prva metoda, ki uporablja aritmetiko v $\mathrm{GF}(p^2)$, da dosega varnost $\mathrm{GF}(p^6)$, ne da bi zahtevala eksplicitno gradnjo $\mathrm{GF}(p^6)$. Naj bo g element reda $q > 6$, ki deli $p^2 - p + 1$. Ker $p^2 - p + 1$ deli red grupe $\mathrm{GF}(p^6)^*$, ki je $p^6 - 1$, bo g generiral podgrupo reda q. Pokazali bomo, da lahko poljubno potenco g predstavimo z enim elementom iz $\mathrm{GF}(p^2)$ in da lahko take potence izračunamo hitro samo z aritmetiko v $\mathrm{GF}(p^2)$.

2.1 Osnove

Naj bo $p \equiv 2 \mod 3$ praštevilo, tako da bo šesti ciklotomičen polinom evaluiran v p imel praštevilski faktor $q > 6$, torej $\phi_6(p) = p^2 - p + 1$, oz. $q | p^2 - p + 1$. V poglavju 3 bomo pokazali, kako hitro izberemo ustrezna p in q. V tem poglavju bomo pokazali, da ne potrebujemo dejanske izgradnje obsega $\mathrm{GF}(p^6)$, po drugi strani pa potrebujemo reprezentacijo $\mathrm{GF}(p^2)$.

Iz $p \equiv 2 \mod 3$ sledi, da $p \mod 3$ generira $\mathrm{GF}(3)^*$, tako da ničli α in α^p polinoma $(X^3 - 1)/(X - 1) = X^2 + X + 1$ tvorita optimalno normalno bazo za $\mathrm{GF}(p^2)$ nad $\mathrm{GF}(p)$. Ker pa $\alpha^i = \alpha^{i \mod 3}$, lahko element $x \in \mathrm{GF}(p^2)$ predstavimo kot $x_1\alpha + x_2\alpha^p = x_1\alpha + x_2\alpha^2$ za $x_1, x_2 \in \mathrm{GF}(p)$. Aritmetične operacije potekajo takole.

Za $x = x_1\alpha + x_2\alpha^2 \in \mathrm{GF}(p^2)$ imamo $x^p = x_1^p\alpha^p + x_2^p\alpha^{2p} = x_2\alpha + x_1\alpha^2$. Od tod sledi, da potenciranje na p ne zahteva nobenih računskih operacij. Kvadriranje x lahko uresničimo z dvema množenjema v $\mathrm{GF}(p)$. Prepričamo

se, da velja $(x_1\alpha+x_2\alpha^p)^2 = x_2(x_2-2x_1)\alpha+x_1(x_1-2x_2)\alpha^2$ od koder vidimo, da potrebujemo samo dve množenji. Kot ponavadi v GF(p) seštevanj in odštevanj ne računamo v analizi. Za množenje dveh elementov najprej izračunamo $x_1 \cdot y_1$, $x_2 \cdot y_2$ in $(x_1+x_2)\cdot(y_1+y_2)$ od koder dobimo $x_1 \cdot y_2 + x_2 \cdot y_1$ s pomočjo dveh odštevanj. Tako lahko množenje realiziramo s tremi množenji. Za izračun $x \cdot z - y \cdot z^p = (z_1(y_1 - x_2 - y_2) + z_2(x_2 - x_1 + y_2))\alpha + (z_1(x_1 - x_2 + y_1) + z_2(y_2 - x_1 - y_1))\alpha^2$ porabimo le štiri množenja. Zgornje rezultate povzamemo v naslednji lemi.

Lema 1. *Naj bodo $x, y, z \in GF(p^2)$ z $p \equiv 2 \mod 3$. Tedaj veljajo naslednje trditve:*

- *x^p lahko izračunamo brez uporabe kakršnihkoli operacij.*
- *x^2 izračunamo z dvema množenjema v $GF(p)$.*
- *$x \cdot y$ izračunamo s tremi množenji v $GF(p)$.*
- *$x \cdot z - y \cdot z^p$ izračunamo s štirimi množenji v $GF(p)$.*

Kot primerjavo bomo pregledali sledeče znane rezultate.

Lema 2. *Naj bodo $x, y \in GF(p^6)$, kjer je $p \equiv 2 \mod 3$ in naj bosta $a, b \in \mathbb{Z}$, $0 < a, b < p$. Predpostavimo še, da za kvadriranje v $GF(p)$ potrebujemo 80% časa množenja v $GF(p)$.*

- *x^2 lahko izračunamo s 14.4 množenji v $GF(p)$.*
- *$x \cdot y$ lahko izračunamo z 18 množenji v $GF(p)$.*
- *x^a lahko izračunamo s pričakovano $23.4 \log_2(a)$ množenji v $GF(p)$.*
- *$x^a \cdot y^b$ lahko izračunamo s pričakovano $27.9 \log_2(\max(a, b))$ množenji v $GF(p)$.*

2.2 Sledi

Konjugiranji elementi $h \in \mathrm{GF}(p^6)$ nad $\mathrm{GF}(p^2)$ so h, h^{p^2} in h^{p^4}. *Sled* $Tr(h)$ nad $\mathrm{GF}(p^2)$ za h je vsota konjugiranih elementov nad $\mathrm{GF}(p^2)$, $Tr(h) = h + h^{p^2} + h^{p^4}$. Ker red elementa h deli $p^6 - 1$, sledi da $Tr(h)^{p^2} = Tr(h)$, od koder sledi $Tr(h) \in \mathrm{GF}(p^2)$. Sled je tudi linearen operator nad $\mathrm{GF}(p^2)$, saj za poljuben $c \in \mathrm{GF}(p^2)$ velja $Tr(h_1 + h_2) = Tr(h_1) + Tr(h_2)$ in $Tr(c \cdot h) = c \cdot Tr(h)$.

Konjugirani elementi za g so g, g^{p-1} in g^{-p}, saj velja da $p^2 \equiv p - 1 \mod p^2 - p + 1$ in $p^4 \equiv -p \mod p^2 - p + 1$.

Lema 3. *Koreni polinoma $X^3 - Tr(g)X^2 + Tr(g)^p X - 1$ so ravno konjugirani elementi za g.*

Dokaz. Primerjajmo koeficiente polinoma $X^3 - Tr(g)X^2 + Tr(g)^p X - 1$ in polinoma $(X-g)(X-g^{p-1})(X-g^{-p})$. Koeficienti pri X^2 so:

$$g + g^{p-1} + g^{-p} = Tr(g).$$

Hitro lahko izračunamo tudi konstanten koeficient:

$$(-g) \cdot (-g)^{p-1} \cdot (-g)^{-p} = -g^{1+p-1-p} = -1.$$

Koeficient pri X je enak:

$$g{\cdot}g^{p-1}{+}g{\cdot}g^{-p}{+}g^{p-1}{\cdot}g^{-p} = g^p{+}g^{1-p}{+}g^{-1} = g^p{+}g^{-p^2}{+}g^{p^2-p} = (g{+}g^{-p}{+}g^{p-1})^p = Tr(g)^p,$$

saj je $1 - p \equiv -p^2 \mod p^2 - p + 1$ in $-1 \equiv p^2 - p \mod p^2 - p + 1$. □

Podobno lahko pokažemo, da so koreni polinoma $X^3 - Tr(g^n)X^2 + Tr(g^n)X - 1$ ravno konjugirani elementi za g^n. Tako so konjugirani elementi za g^n popolnoma določeni s tem polinomom, oziroma z $Tr(g^n)$. Ker pa je $Tr(g^n) \in \mathrm{GF}(p^2)$, sledi, da lahko kompaktno predstavimo konjugirane elemente g^n. Če želimo uporabljati to reprezentacijo, moramo znati hitro izračunati $Tr(g^n)$ iz $Tr(g)$. Takšno metodo bomo opisali kasneje v poglavju 3, saj jo bomo potrebovali v bolj splošne namene. Zato bomo pogledali nekaj lastnosti polinomov $X^3 - cX^2 + c^p X - 1$ za splošen $c \in \mathrm{GF}(p^2)$.

2.3 Polinom F(c,X)

Definicija 1. *Za $c \in GF(p^2)$ naj bo $F(c,X)$ polinom $X^3 - cX^2 + c^p X - 1 \in GF(p^2)[X]$ z ne nujno različnimi koreni h_0, h_1, h_2 v $GF(p^6)$ in naj bo $\tau(c,n) = h_0^n + h_1^n + h_2^n$ za $n \in \mathbb{Z}$. Uporabljali bomo notacijo $c_n = \tau(c,n)$.*

V tem podpoglavju si bomo pogledali nekaj lastnosti polinoma $F(c,X)$ in njegovih korenov.

Lema 4. *Veljajo naslednje točke:*

1. $c_1 = c$.
2. $h_0 \cdot h_1 \cdot h_2 = 1$.
3. $c_{-n} = h_0^n \cdot h_1^n + h_0^n \cdot h_2^n + h_1^n \cdot h_2^n$.
4. $F(c, h_j^{-p}) = 0$ *za* $j = 0, 1, 2$.
5. $c_{-n} = c_{np} = c_n^p$.
6. *Ali so vsi* h_j *reda deljivega z* $p^2 - p + 1$ *in* $h_j > 3$ *ali so vsi* $h_j \in GF(p^2)$.
7. $c_n \in GF(p^2)$.

Dokaz. Točka 1 in točka 2 sledita direktno iz definicije in Vietovih formul, točka 3 pa sledi iz točke 2. Iz $F(c,h_j) = h_j^3 - ch_j^2 + c^{p^2}h_j^p - 1 = 0$ sledi, da $h_j \neq 0$ in $F(c,h_j)^p = h_j^{3p} - c^p h_j^{2p} + c^{p^2}h_j^p - 1 = 0$. Z $c^{p^2} = c$ in $h_j \neq 0$ sledi, da $-h_j^{3p}(h_j^{-3p} - ch_j^{-2p} + c^p h_j^{-p} - 1) = -h_j^{3p} \cdot F(c, h_j^{-p)} = 0$, kar dokazuje točko 4.

Iz točke 4 sledi, brez izgube splošnosti, da ali $h_j = h_j^{-p}$ za $j = 0,1,2$, ali $h_0 = h_0^{-p}$, $h_1 = h_2^{-p}$ in $h_2 = h_1^{-p}$, ali da $h_j = h_{j+1 \mod 3}^{-p}$ za $j = 0,1,2$. V vsakem primeru sledi točka 5. Nadalje, v prvem primeru imajo vsi h_j red deljiv s $p+1$ in so v GF(p^2). V drugem primeru ima h_0 red deljiv s $p+1$, in ker velja $h_1 = h_1^{p^2}$ in $h_2 = h_2^{p^2}$, imata h_1 in h_2 red deljiv s p^2-1, od koder spet sledi, da so vsi v GF(p^2). V zadnjem primeru pa sledi iz $1 = h_0 \cdot h_1 \cdot h_2$, da $1 = h_0 \cdot h_2^{-p} \cdot h_0^{-p} = h_0 \cdot h_0^{p^2} \cdot h_0^{-p} = h_0^{p^2-p+1}$, tako da ima h_0 red deljiv s $p^2 - p + 1$ in podobno h_1 in h_2. V vsakem od primerov, ima h_0 red največ 3, potem ima h_0 red 1 ali 3, saj je $p^2 - p + 1$ lih. Sledi, da red elementa h_0 deli $p^2 - 1$, in je $h_0 \in$GF(p^2). Ampak potem sta h_1 in h_2 tudi v GF(p^2), ker $h_j = h_{j+1 \mod 3}^{-p}$. Sledi, da v zadnjem primeru ali imajo vsi h_j red deljiv s $p^2 - p + 1$ in > 3 ali pa so vsi v GF(p^2).

Če so vsi $h_j \in$GF(p^2), potem točka 7 direktno sledi, saj sta potenciranje na n in seštevanje zaprti operaciji v GF(p^2). Sicer pa je $F(c,X)$ nerazcepen in so njegovi koreni konjugirani elementi za h_0. Torej $c_n = Tr(h_0^n) \in$GF(p^2). S tem je pokazana še zadnja točka leme. □

Lema 5. *Veljajo naslednje točke:*

- $c_{u+v} = c_u \cdot c_v - c_v^p \cdot c_{u-v} + c_{u-2v}$.
- $F(c_n, h_j^n) = 0$ *za* $j = 0,1,2$.
- $F(c,X)$ *je nerazcepen nad GF*(p^2) *natanko tedajo ko* $c_{p+1} \in GF(p)$.

Dokaz. Dokaz si lahko bralec prebere v [1] na strani 6. □

Iz lem 4 in 5 sledi hiter algoritem za izračun c_n za poljuben $n \in \mathbb{Z}$.

Posledica 1. *Naj bodo podani* c, c_{n-1}, c_n *in* c_{n+1}.

1. *Izračun* $c_{2n} = c_n^2 - 2c_n^p$ *vzame dve množenji v GF*(p).
2. *Izračun* $c_{n+2} = c \cdot c_{n+1} - c^p \cdot c_n + c_{n-1}$ *vzame štiri množenja v GF*(p).
3. *Izračun* $c_{2n-1} = c_{n-1} \cdot c_n - c^p \cdot c_n^p + c_{n+1}^p$ *vzame štiri množenja v GF*(p).
4. *Izračun* $c_{2n+1} = c_{n+1} \cdot c_n - c \cdot c_n^p + c_{n-1}^p$ *vzame štiri množenja v GF*(p).

Dokaz. Identitete sledijo iz lem 4 in 5. Z $u = v = n$ in $c_0 = 3$, za 1. točko, z $u = n+1$ in $v = 1$ za 2., $u = n-1$, $v = n$ za 3. in $u = n+1$, $v = n$ za 4. točko. Analiza števila množenj pa sledi iz leme 1. □

Definicija 2. *Naj bo* $S_n(c) = (c_{n-1}, c_n, c_{n+1}) \in GF(p^2)^3$.

Algoritem 1. *Izračun* $s_n(c)$ *s podanim* c.

1. *Če* $n < 0$, *kličemo ta algoritem z* $-n$ *in uporabimo lemo 4.*
2. *Če* $n = 0$, *potem* $S_0(c) = (c^p, 3, c)$.
3. *Če je* $n = 1$, *potem vrnemo* $S_1(c) = (3, c, c^2 - 2c^p)$.
4. *Če je* $n = 2$ *uporabimo posledico 1 in* $S_1(c)$ *za izračun* c_3 *in s tem* $S_2(n)$.
5. *Sicer za* $n > 2$, *naj bo* $m = n$.
   - *Če je* m *sodo, potem zamenjajmo* m *z* $m - 1$.
   - *Naj bo* $\overline{S}_t(c) = S_{2t+1}(c)$, *za* $t \in \mathbb{Z}$, $k = 1$ *in izračunamo* $\overline{S}_k(c) = S_3(c)$.
   - *Naj bo* $\frac{m-1}{2} = \sum_{j=0}^{r} m_j 2^j$, *z* $m_j \in \{0, 1\}$ *in* $m_r = 1$.
   - *Za* $j = r - 1, \ldots, 0$ *ponovimo:*
     - *Če* $m_j = 0$ *uporabimo* $\overline{S}_k(c) = (c_{2k}, c_{2k+1}, c_{2k+2})$ *za izračun* $\overline{S}_{2k}(c) = (c_{4k}, c_{4k+1}, c_{4k+2})$ *s pomočjo posledice 1, in nadomestimo* k *z* $2k$.
     - *Če* $m_j = 1$ *uporabimo* $\overline{S}_k(c) = (c_{2k}, c_{2k+1}, c_{2k+2})$ *za izračun* $\overline{S}_{2k+1}(c) = (c_{4k+2}, c_{4k+3}, c_{4k+4})$ *s pomočjo posledice 1, in nadomestimo* k *z* $2k + 1$.
6. *Po tej iteraciji velja* $2k + 1 = m$ *in tako* $S_m(c) = \overline{S}_k(c)$.
7. *Če je* n *sod uporabimo* $S_m(c) = (c_{m-1}, c_m, c_{m+1})$ *za izračun* S_{m+1} *s pomočjo posledice 1 in nadomestimo* m *z* $m + 1$.
8. *Kot rezultat imamo* $S_n(c) = S_m(c)$.

Trditev 3. *S podano vsoto* c *korenov polinoma* $F(c, X)$, *za izračun vsote* c_n, *ki so* n*-te potence korenov, potrebujemo* $8 \log_2(n)$ *množenj v* $GF(p)$.

Dokaz. Sledi direktno iz algoritma 1 in posledice 1. □

Opomba 1. *Edina razlika med primeroma, ko je* $m_j = 0$ *ali* $m_j = 1$, *je zgolj v tem, katero točko posledice 1 uporabimo. Oba izračuna sta podobna in porabita isto število operacij. To je dokaj nenavadna lastnost za algoritme potenciranja, kar pa pomeni, da je algoritem razmeroma varen pred okoljskimi napadi, kot je na primer časovni napad.*

2.4 Računanje s sledmi

Iz lem 3 in 5 sledi, da $S_n(Tr(g)) = (Tr(g^{n-1}), Tr(g^n), Tr(g^{n+1}))$. Še več, s podanim $Tr(g)$, lahko s pomočjo algoritma 1 izračunamo $S_n(Tr(g))$, za poljuben $n \in \mathbb{Z}$. Ker pa je red g enak q, za algoritem porabimo $8\log_2(n \mod q)$ množenj v GF(p). Po lemi 2 za izračun g^n s podanim g, potrebujemo pričakovano $23.4\log_2(q)$ množenj v GF(p), kar je trikrat počasneje kot izračun $Tr(g^n)$ s podanim $Tr(g)$. Poleg tega je element $Tr(g^n) \in$GF(p^2), medtem ko je $g^n \in$GF(p^6), s čimer še trikrat zmanjšamo količino zavzetega prostora za shranjevanje $Tr(g^n)$ v primerjavi z g^n.

Splača se nam nadomestiti g^n s $Tr(g^n)$ zaradi prostorske in časovne zahtevnosti, kar lahko uporabimo v mnogih kriptografskih protokolih. Ampak v nekaterih kriptografskih protokolih potrebujemo možnost izračuna produkta dveh potenc števil. Za standardno reprezentacijo je to trivialno, medtem ko je z uporabo *sledi* operacija relativno zahtevna. Naj bosta $Tr(g) \in$GF(p^2) in $S_k(Tr(g)) \in$GF$(p^2)^3$ podana za neko zasebno naravno število k, $0 < k < q$.

Definicija 4. *Naj bosta*

$$A(c) = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -c^p \\ 0 & 1 & c \end{pmatrix},$$

$$M_n(c) = \begin{pmatrix} c_{n-2} & c_{n-1} & c_n \\ c_{n-1} & c_n & c_{n+1} \\ c_n & c_{n+1} & c_{n+2} \end{pmatrix}$$

3×3 *matriki nad GF(p^2) s c in c_n kot v definiciji 1, in naj bo $C(V)$ srednji stolpec v matriki V.*

Lema 6. $S_n(c) = S_m(c) \cdot A(c)^{n-m}$ *in* $M_n(c) = M_m(c) \cdot A(c)^{n-m}$ *za* $n, m \in \mathbb{Z}$.

Dokaz. Za $n - m = 1$ je lema zgolj prepisana posledica 1. To nam ustvari bazo indukcije. Dokaz sledi z indukcijo po $n - m$. □

Posledica 2. $c_n = S_m(c) \cdot C(A(c)^{n-m})$.

Lema 7. *Determinanta $M_o(c)$ je enaka $D = c^{2p+2} + 18c^{p+1} - 4(c^{3p} + c^3) - 27 \in GF(p)$. Če $D \neq 0$, potem velja*

$$M_0(c)^{-1} = \frac{1}{D} \begin{pmatrix} 2c^2 - 6c^p & 2c^{2p} + 3c - c^{p+2} & c^{p+1} - p \\ 2c^{2p} + 3c - c^{p+2} & (c^2 - 2c^p)^{p+1} - 9 & (2c^{2p} + 3c - c^{p+2})^p \\ c^{p+1} - 9 & (2c^{2p} + 3c - c^{p+2})^p & (2c^2 - 6c^p)^p \end{pmatrix}$$

Dokaz. Sledi iz preprostega izračuna s pomočjo leme 4 in posledice 1. Upoštevati moramo še, da za vsak $c \in$ GF(p) velja $c^p = c$. □

Lema 8. *$det(M_0(Tr(g))) = (Tr(g^{p+1})^p - Tr(g^{p+1}))^2 \neq 0$*

Dokaz. Dokaz si lahko bralec prebere v [1] na strani 8. □

Lema 9. *$A(Tr(g))^n = M_0(Tr(g))^{-1} \cdot M_n(Tr(g))$ lahko izračunamo z majhnim konstantnim številom operacij v GF(p^2) s podanima $Tr(g)$ in $S_n(Tr(g))$.*

Dokaz. Dokaz si lahko bralec prebere v [1] na strani 9. □

Posledica 3. *$C(A(Tr(g))^n) = M_0(Tr(g))^{-1} \cdot (S_n(Tr(g)))^T$*

Algoritem 2. *Izračun $Tr(g^a \cdot g^{bk})$. Naj bodo podani $Tr(g), S_k(Tr(g))$, za neznani k in $a, b \in \mathbb{Z}$, z $0 < a, b < q$.*

1. *Izračunamo $e = ab^{-1} \mod q$.*
2. *Izračunamo $S_e(Tr(g))$ z algoritmom 1.*
3. *Izračunamo $C(A(Tr(g))^e)$ s podanimi $Tr(g)$ in $S_e(Tr(g))$ z uporabo posledice 3.*
4. *Izračunamo $Tr(g^{e+k})$ z algoritmom 1, in vrnemo $Tr(g^{(e+k)b}) = Tr(g^a \cdot g^{bk})$.*

Trditev 5. *S podanimi $M_0(Tr(g))^{-1}, Tr(g)$ in $S_k(Tr(g))$, lahko sled $Tr(g^a \cdot g^{kb})$ izračunamo z $8\log_2(ab^{-1} \mod q) + 8\log_2(b) + 34$ množenji v GF(p).*

Dokaz. Časovna analiza sledi z direktno analizo potrebnih vektorskih operacij v matrikah in s trditvijo 3. □

S predpostavko, da izračunamo $M_0(Tr(g))^{-1}$ samo enkrat za vse operacije vnaprej, ugotovimo, da lahko $Tr(g^a \cdot g^{bk})$ izračunamo z zgolj $16\log_2(b)$ množenji v GF(p). S tradicionalnim načinom bi ta operacija zahtevala pričakovano $27.9\log_2(q)$ množenj v GF(p). Torej z novo reprezentacijo dobimo pohitritveni faktor 1.75 za operacijo množenja dveh potenc.

3 Pravilna in hitra izbira parametrov

3.1 Izbira končnega obsega in velikosti podgrupe

Opisali bomo hitre in praktične metode za izbiro karakteristike p in velikosti podgrupe q, tako da q deli $p^2 - p + 1$. Naj bosta P in Q velikosti praštevil p in q v bitih. Za doseg varnosti ekvivalentne 1024 bitnemu RSA, mora biti $6P \approx 1024$, npr. $P \approx 160$ in velikost q lahko nastavimo na $Q \approx 160$. Glede na trenutne kriptoanalitične metode, ni priporočljivo nastaviti Q bistveno manjši kot P. Če želimo doseči varnost RSA kriptosistema, ki bo predvidoma varen do vsaj do leta 2030 - torej z 3072 biti, moramo P in Q nastaviti na približno 512 bitov.

Algoritem 3. *Izbira q in 'lepega' p.*

Poiščemo $r \in \mathbb{Z}$, tako da je $q = r^2 - r + 1$ Q bitno praštevilo in nato poiščemo $k \in \mathbb{Z}$, tako da je $p = r + k \cdot q$ P bitno praštevilo in da velja $p = 2 \mod 3$.

Algoritem je dokaj hiter in uporabimo ga lahko, da najdemo praštevilo p, ki ima zahtevane lastnosti. Tak p nam zagotavlja hitre aritmetične operacije v GF(p). Z implementacijo algoritma v sistemu Mathematica, smo pri naključnem izbiranju števil za $r \in [4 \cdot 10^{25}, 10^{26}]$ in testiranje praštevilskosti za $r^2 - r + 1$ potrebovali povprečno 50.857 naključnih izbir za r. Nato smo pa za pravilni k morali preizkusiti 64.69 naključno izbranih elementov iz intervala $[2^{10}, 2^{15}]$. Test smo ponovili 1000 krat. Za vsak test smo v povprečju potrebovali $0.015ms$.

Algoritem 4. *Izbira p in q.*

Izberemo Q bitno praštevilo $q \equiv 7 \mod 12$. Nato najdemo korena r_1 in r_2 polinoma $X^2 - X + 1 \mod q$. Iz dejstva $q \equiv 1 \mod 3$ in kvadratične recipročnosti sledi, da r_1 in r_2 obstajata. Ker pa je $q \equiv 3 \mod 4$, ju lahko najdemo z enim $(q+1)/4$ potenciranjem po modulu q. Nato poiščemo še $k \in \mathbb{Z}$, tako da za $i = 1$ ali $i = 2$ je $p = r_i + k \cdot q$ P bitno praštevilo.

Test smo ponovno ponovili 1000 krat, pri čemer je vsaka ponovitev povprečno trajala $0.041ms$. Pri tem smo izbirali q naključno enakomerno iz intervala $[2^{160}, 2^{165}]$, pri čemer je bilo povprečno potrebnih 466.455 izbir. Za vsak q smo pa nato izbirali k naključno enakomerno iz intervala $[2^{10}, 2^{15}]$. Povprečno smo morali testirati 62.2 vrednosti za k. Implementacija uporablja samo vgrajene funkcije sistema Mathematica.

V obeh primerih smo ciljali na q velikostnega reda vsaj 160 bitov in p velikostnega reda vsaj 170 bitov. Vse teste smo izvajali na Linux operacijskem sistemu s procesorjem Intel Core2Duo $1.87GHz$. Časovna kompleksnost obeh algoritmov je dominirana s časom iskanja praštevil p in q.

3.2 Izbira podgrupe

Problem je najti pravilen $Tr(g)$ za nek element $g \in$GF(p^6) reda $q > 3$, ki deli $p^2 - p + 1$. Poudariti je treba, da ni potrebno iskati g, ampak zadostuje najti samo $Tr(g)$. Za podani $Tr(g)$, lahko izračunamo generator podgrupe, če najdemo koren polinoma $F(Tr(g), X)$ v GF(p^6). Ta generator bomo označili z g in red podgrupe $\langle g \rangle$ s q. Podgrupi $\langle g \rangle$ pravimo tudi XTR-grupa. Opazimo, da vsi koreni polinoma $F(Tr(g), X)$ vodijo do iste XTR-grupe.

Direkten način za iskanje $Tr(g)$ bi bil, da najdemo nerazcepen polinom stopnje 3 nad GF(p^2), ki ga uporabimo za reprezentacijo GF(p^6) in nato izbiramo elemente $h \in$GF(p^6) dokler $h^{\frac{p^6-1}{q}} \neq 1$. Nato bi vzeli $g = h^{\frac{p^6-1}{q}}$ in s tem izračunali $Tr(g)$. Takšen način je konceptualno lažji, vendar je težji

iz implementacijskega stališča. Lažji način implementacije sledi iz sledeče leme.

Lema 10. *Za enakomerno naključno izbrani $c \in GF(p^2)$ je verjetnost, da je $F(c, X) \in GF(p^2)[X]$ nerazcepen, približno $\frac{1}{3}$.*

Dokaz. Dokaz bomo izpeljali z enostavnim načinom preštevanja. Približno $p^2 - p$ elementov podgrupe reda $p^2 - p + 1$ nad $\mathrm{GF}(p^6)^*$ je korenov moničnih nerazcepnih polinomov oblike $F(c, X)$. Ker ima vsak izmed teh polinomov tri različne korene, mora biti $\frac{p^2-p}{3}$ različnih vrednosti za $c \in \mathrm{GF}(p^2) \setminus \mathrm{GF}(p)$, takih, da je $F(c, X)$ nerazcepen. □

Dovolj je, da izbiramo $c \in \mathrm{GF}(p^2)$, dokler polinom $F(c, X)$ ni nerazcepen in $c_{\frac{p^2-p+1}{3}} \neq 3$, nato določimo $Tr(g) = c_{\frac{p^2-p+1}{3}}$. Izbrani $Tr(g)$ je sled nekega g reda q, vendar eksplicitna določitev g ni bila potrebna.

Algoritem 5. *Izračun $Tr(g)$*

1. *Izberemo enakomerno naključno $c \in GF(p^2) \setminus GF(p)$ in izračunamo c_{p+1} z algoritmom 1.*
2. *Če je $c_{p+1} \in GF(p)$, potem se vrnemo na korak 1.*
3. *Izračunamo $c_{\frac{p^2-p+1}{q}}$ z algoritmom 1.*
4. *Če je $c_{\frac{p^2-p+1}{q}} = 3$, potem se vrnemo na korak 1.*
5. *Vrnemo $Tr(g) = c_{\frac{p^2-p+1}{q}}$.*

Trditev 6. *Algoritem 5 izračuna element $Tr(g)$ za nek $g \in GF(p^6)$ reda q. Pričakovano zahteva $\frac{3q}{q-1}$ klicev algoritma 1 z $n = p + 1$ in $\frac{q}{q-1}$ klicev z $n = \frac{p^2-p+1}{q}$.*

Dokaz. Pravilnost algoritma sledi iz dejstva, da je $F(c, X)$ nerazcepen, če $c_{p+1} \notin \mathrm{GF}(p)$. Časovna zahtevnost pa sledi iz leme 10 in iz dejstva, da je $F(c, X)$ nerazcepen, če $c_{p+1} \notin \mathrm{GF}(p)$. □

3.3 Velikost ključa

XTR javni podatki vsebujejo dve praštevili p in q in še sled $Tr(g)$ generatorja XTR-grupe. V osnovi lahko podatke $p, q, Tr(g)$ delimo med poljubnim številom klientov. Poleg tega lahko javni ključ vsebuje še $Tr(g^k)$ za nek privatni k. Za nekatere uporabe, potrebujemo še $Tr(g^{k-1})$, ter $Tr(g^{k+1})$. V tem razdelku si bomo pogledali približne prostorske zahtevnosti za predstavitev XTR javnega ključa v certifikatu, povrhu uporabniškega ID in ostalih bitih povezanih s certifikati.

$(Tr(g), p, q)$ zasedajo prostor v certifikatu, samo če niso deljeni. V tem primeru, lahko predpostavimo, da $(Tr(g), p, q)$ pripadajo določenemu uporabniku ali skupini uporabnikov. V vsakem primeru je to lahko neka funkcija uporabniškega ID in majhnega števila dodatnih bitov, ki jo izračunamo ob inicializaciji. Ob izbiri P in Q lahko število dodatnih bitov ob znanem uporabniškem ID omejimo na 48, s ceno enega klica algoritma 1 z $n = \frac{p^2-p+1}{q}$.

Za predstavitev javnega ključa uporabnika $Tr(g^k)$ v certifikatu uporabimo $2P$ bitov. Ostali dve vrednosti $Tr(g^{k-1}), Tr(g^{k+1})$ lahko predstavimo z bistveno manj biti kot $4P$, za ceno enkratnega izračuna prejemnika javnega ključa. Tako lahko javni ključ opišemo zgolj z $(Tr(g), p, q, Tr(g^k))$ in z enkratnim izračunom določimo še $Tr(g^{k-1})$ in $Tr(g^{k+1})$.

4 Uporabe XTR kriptositema

XTR lahko uporabimo v vsakem kriptosistemu, katerega varnost temelji na problemu diskretnega logaritma. V tem poglavju bomo opisali, kako lahko XTR vključiti v Diffie-Hellman-ovo izmenjavo ključa in kako XTR vključimo v ElGamal-ovo šifriranje.

4.1 XTR-DH

Recimo, da se želita Alenka in Bojan, ki imata oba dostop do javnih podatkov XTR kriptosistema p, q, $Tr(g)$, dogovoriti za skupni simetrični ključ K. To lahko dosežeta s sledečo shemo:

1. Alenka si izbere naključen $a \in \mathbb{N}$, $1 < a < q-2$, in uporabi algoritem 1, da izračuna $S_a(Tr(g)) = (Tr(g^{a-1}), Tr(g^a), Tr(g^{a+1})) \in \mathrm{GF}(p^2)^3$. Nato pošlje $Tr(g^a)$ Bojanu.

2. Bojan sprejme $Tr(g^a)$ od Alenke, izbere naključen $b \in \mathbb{N}$, $1 < b < q-2$, uporabi algoritem 1 za izračun $S_b(Tr(g))$, in pošlje $Tr(g^b)$ Alenki.

3. Alenka sprejme $Tr(g^b)$ od Bojana in uporabi algoritem 1 za izračun $S_a(Tr(g^b)) = (Tr(g^{(a-1)b}), Tr(g^{ab}), Tr(g^{(a+1)b}))$ in določi simetrični ključ K na podlagi $Tr(g^{ab})$.

4. Bojan uporabi algoritem 1 za izračun $S_b(Tr(g^a))$ in določi simetrični ključ K na podlagi $Tr(g^{ab})$.

S takšno implementacijo Diffie-Hellman-ove izmenjave ključa zmanjšamo ceno izračunov in ceno prenosov za tretjino v primerjavi s tradicionalno Diffie-Hellman shemo, da ohranimo isti nivo varnosti.

4.2 XTR-ElGamal šifriranje

Recimo, da je Alenka lastnica javnega ključa p, q, $Tr(g)$ in da je izbrala privatno naravno število k, izračunala $S_k(Tr(g))$ in dodala $Tr(g^k)$ k javnemu ključu. S podanim javnim ključem $(p, q, Tr(g), Tr(g^k))$, lahko Bojan šifrira sporočilo M, ki je namenjeno samo Alenki, s spodnjo verzijo ElGamal-ovega kriptosistema, ki temelji na XTR.

1. Bojan izbere naključen $b \in \mathbb{N}$, $1 < b < q - 2$, in uporabi algoritem 1 za izračun $S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})) \in \mathrm{GF}(p^2)^3$.

2. Bojan uporabi algoritem 1 za izračun $S_b(Tr(g^k)) = (Tr(g^{k(b-1)}), Tr(g^{kb}), Tr(g^{k(b+1)})) \in \mathrm{GF}(p^2)^3$.

3. Bojan določi simetrični šifrirni ključ K na podlagi $Tr(g^{kb})$ in s ključem šifrira sporočilo M, s čimer pridela tajnopis E.

4. Bojan pošlje E k Alenki.

Ko Alenka sprejme $(Tr(g^b), E)$ lahko dešifrira sporočilo s spodnjim postopkom.

1. Alenka uporabi algoritem 1 za izračun $S_k(Tr(g^b))$, od koder s pomočjo $Tr(g^{bk})$ določi simetrični ključ K.

2. Alenka dešifrira sporočilo E s pomočjo ključa K in dobi čistopis M.

Tudi ElGamal-ov kriptosistem, ki bazira na XTR shemi ima za tretjino manjše število izračunov in prenaša za tretjino manj bitov med uporabniki, v primerjavi s tradicionalnim ElGamal-ovim kriptosistemom, ki bazira na podgrupah multiplikativnih grup v končnih obsegih in ki dosegajo isti nivo varnosti.

5 Primerjave kriptosistema

Kriptosistem XTR primerjamo z RSA in eliptičnimi krivuljami. Rezultati so povzeti po [1]. Za primerjavo z RSA so podali čase izvajanje za 1020 bitni RSA in 170 bitni XTR v uporabi z generično programsko opremo. Za eliptične krivulje predvidevajo naključne krivulje nad praštevilskimi obsegi s približno 170 biti, v podgrupi s 170 bitnim redom in nato primerjajo število množenj v $\mathrm{GF}(p)$ za eliptične krivulje in XTR kriptosistem. Uporabljena je teoretična primerjava, saj nimajo dostopa do programske opreme za eliptične krivulje.

Za RSA in za XTR so zgenerirali 100 naključnih ključev (za eliptične krivulje je generiranje parametrov bistveno težje in počasnejše, zato ni podano v tabeli 2). Za RSA so uporabili naključno izbrano liho število s 510

Tabela 1: Velikosti ključev za RSA, XTR in eliptične krivulje

	Velikost deljenega ključa	Velikost ključa na podlagi ID	Velikost ključa brez ID
1020 bitni RSA		510 bitov	1050 bitov
170 bitni XTR	340 bitov	388 bitov	680 bitov
170 bitni ECC	171 bitov	304 bitov	766 bitov

Tabela 2: Hitrost izvajanja za RSA in XTR

	Izbira ključa	Šifriranje	Dešifriranje
1020 bitni RSA	1224 ms	5 ms	40 (123) ms
170 bitni XTR	73 ms	23 ms	11 ms

Tabela 3: 170 bitni ECC, XTR - primerjava števila množenj v GF(p)

	Šifriranje	Dešifriranje	Podpisovanje	Preverjanje podpisa	Hitrost DH	Velikost DH
ECC	3400	1921(1700)	1700	2575	3842(3400)	171(340) bitov
XTR	2720	1360	1360	2754	2720	340 bitov

biti, nato so dodajali 2, dokler ni bilo praštevilo. Za XTR so uporabili algoritem 4 s $Q = 170$ in $P \geq 170$ in hitro izbiro $Tr(g)$. Za vsak RSA ključ so naredili 10 šifriranj in dešifriranj na naključnih 1020 bitnih sporočilih s pomočjo Kitajskega Izreka o Ostankih (in brez v oklepaju v tabeli 2). Za vsak XTR ključ so naredili 10 enojnih in dvojnih potenciranj (aplikacija algoritma 1 in algoritma 2) za naključne potence $< q$. Za RSA je šifriranje in dešifriranje podobno kot preverjanje podpisa in generiranje podpisa. Za XTR je enojno potenciranje kot dešifriranje in generiranje podpisa, dvojno potenciranje pa kot generairanje podpisa in približno šifriranje.

Povprečni časi izvajanj so v milisekundah na 450MHz Pentium II procesorju. Številke za eliptične krivulje v tabeli 5 so prepisane iz [5]. S podanimi y-koordinatami so podatki zapisani v oklepajih. Vsi časi izvajanj XTR kriptosistema, so doseženi z osnovnimi algoritmi, ki smo jih izpeljali in dokazali v tej seminarski nalogi. V kasnješem razvoju, so algoritme še izpopolnili. Novi algoritmi enojnega in dvojnega potenciranja so v povprečju za več kot 50% hitrejši, kar še pospeši že tako hitro podpisovanje XTR kriptosistema, ob tem moramo vnaprej izračunati zelo majhen delež podatkov. Dodaten doprinos dvojnega potenciranja je, da ni več potrebna uporaba matrik, kar naredi implementacijo bistveno lažjo. Za več podatkov in informacij, bralca usmerjamo k [6].

Rezultatov testiranj na novejših platformah žal nismo našli. V članku [1] in [6] še žal niso podani rezultati za generiranje eliptičnih krivulj in štetje točk na njih, saj za ta problem že obstajajo algoritmi, ki temeljijo na aritemtično-geometrični sredini in potrebujejo primerljiv čas izvajanja.

6 Varnost XTR kriptosistema

6.1 Diskretni logaritem v GF(p^t)

Naj bo $\langle\gamma\rangle$ multiplikativna grupa reda ω. Varnost Diffie-Hellman-ovega protokola temelji na *Diffie-Hellman* problemu (DH), ki je problem izračuna γ^{xy} s podanima γ^x in γ^y. Pišemo $\mathrm{DH}(\gamma^x, \gamma^y) = \gamma^{xy}$. Dva druga problema sta tudi povezana z DH problemom. Prvi je *Diffie-Hellman Decision* problem (DHD): za podane $a, b, c \in \langle\gamma\rangle$, določi ali je $\mathrm{DH}(a, b) = c$. DH problem je vsaj tako težek kot DHD. Drugi pa je problem *Diskretnega Logaritma* (DL): za podan $a = \gamma^x \in \langle\gamma\rangle$ z $0 \leq x < \omega$, najdi $x = \mathrm{DL}(a)$. Problem DL je vsaj tako težek kot DH. S podano faktorizacijo za ω lahko problem DL reduciramo na vse podgrupe praštevilskega reda v $\langle\gamma\rangle$, po Pohlig-Hellmanovem algoritmu. Torej za problem DL predpostavimo, da je ω praštevilo.

S poznanimi napadi, je problem DL v XTR grupi splošno poznan kot težji, v primerjavi s faktorizacijo $6\log_2(p)$ bitnega RSA modula, če je velikost praštevila q dovolj velika. S primerjavo računskih potreb za omenjena algoritma, se izkaže da če sta p in q velikostnega reda 170 bitov, potem je problem DL v XTR grupi težji kot faktorizacija RSA modula velikosti $6 \cdot 170 = 1020$ bitov.

6.2 Varnost XTR

Kriptografski protokoli, ki bazirajo na diskretnem logaritmu lahko uporabljajo različne tipe podgrup, kot so multiplikativne grupe končnih obsegov, podgrup kot je XTR grupa, grupe točke eliptičnih krivulj nad končnimi obsegi itn. Varnost verzije, ki bazira na XTR grupi ne moremo gledati z osnovnimi DH, DHD in DL problemi, ampak na XTR verzijah teh problemov. Definirajmo XTR-DH problem kot problem izračuna $Tr(g^{xy})$ s podanima $Tr(g^x)$ in $Tr(g^y)$, in pišemo $\mathrm{XDH}(g^x, g^y) = g^{xy}$. Problem XTR-DHD je problem določanja ali velja $\mathrm{XDH}(a, b) = c$, za $a, b, c \in Tr(\langle g\rangle)$. Za podan $a \in Tr(\langle g\rangle)$, je XTR-DL problem iskanja $x = \mathrm{XDL}(a)$, za $0 \leq x < q$, tako da $a = Tr(g^x)$. Naj povemo še, da če $x = \mathrm{DL}(a)$, potem so rešitve tudi $x \cdot p^2 \mod q$ in $x \cdot p^4 \mod q$.

Pravimo, da je problem $\mathcal{A}$ (a, b)-ekvivalenten problemu $\mathcal{B}$, če za vsako instanco problema v $\mathcal{A}$ potrebujemo največ a klicev algoritmu, ki rešuje problem $\mathcal{B}$, in če za vsako instanco problema v $\mathcal{B}$ potrebujemo največ b klicev algoritmu, ki rešuje problem $\mathcal{A}$.

Trditev 7. *Držijo sledeče ekvivalence:*

(i.) XTR-DL problem je $(1, 1)$*-ekvivalenten s problemom DL v* $\langle g\rangle$*.*

(ii.) XTR-DH problem je $(1, 2)$*-ekvivalenten z DH problemom v* $\langle g\rangle$*.*

(iii.) XTR-DHD problem je $(3, 2)$*-ekvivalenten z DHD problemom v* $\langle g\rangle$*.*

Dokaz. Za $a \in \mathrm{GF}(p^2)$ naj bo $r(a)$ koren polinoma $F(a, X)$. Za izračun $\mathrm{DL}(y)$, naj bo $x = \mathrm{XDL}(Tr(y))$, potem je $\mathrm{DL}(y) = x \cdot p^{2j} \mod q$ za $j = 0$, $j = 1$ ali $j = 2$. Obratno je $\mathrm{XDL}(a) = \mathrm{DL}(r(a))$. To dokazuje točko (i).

Za izračun $\mathrm{DH}(x, y)$, izračunamo $d_i = \mathrm{XDH}(Tr(x \cdot g^i), Tr(y))$ za $i = 0, 1$, potem $r(d_i) \in \{(\mathrm{DH}(x, y) \cdot y^i)^{p^{2j}} : j = 0, 1, 2\}$, od koder sledi $\mathrm{DH}(x, y)$. Obratno je $\mathrm{XDH}(a, b) = Tr(\mathrm{DH}(r(a), r(b)))$. S tem smo pokazali točko (ii).

Za dokaz iii, sledi da $\mathrm{DHD}(x, y) = z$ natanko tedaj ko $\mathrm{XDHD}(Tr(x), Tr(y)) = Tr(z)$ in $\mathrm{XDHD}(Tr(x \cdot g), Tr(y)) = Tr(z \cdot y)$. Obratno, $\mathrm{XDHD}(a, b) = c$ natanko tedaj ko $\mathrm{DH}(r(a), r(b)) = r(c)^{p^{2j}}$ za $j = 0$, $j = 1$ ali $j = 2$. S tem smo pokazali še točko (iii). □

7 Nadaljne delo

V seminarski nalogi smo zapisali osnovne izreke in algoritme, ki jih potrebuje XTR kriptosistem za delovanje. Vsekakor je še dosti prostora za nadaljne delo, ki se lahko usmeri v pohitritev že tako hitrih algoritmov [6]. Pravtako je še dosti prostora za napredek na področju zmanjšanja uporabljenega prostora za predstavitev ključa in javnih podatkov. Sistem se lahko posploši, da ne deluje samo v $\mathrm{GF}(p^6)$ ampak tudi v $\mathrm{GF}(p^c)$, za $c > 6$. V tem primeru se lahko pokažejo še dodatne optimizacije algoritmov in doprinosi k prostoru. Nazadnje je pa potrebna še primerjava sistema XTR prot eliptičnim krivuljam in sistemu RSA na sodobnejših računalnikih in platformah, v mislu števila množenj in časa.

Literatura

[1] The XTR public key system, A. Lenstra; E. Verheul, 2000.

[2] An overview of the XTR public key system, A. Lenstra; E. Verheul, Public-Key Cryptography and Computational Number Theory, Verlages Walter de Gruyter, 2000.

[3] C. P. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology, 4 (1991), 161-174.

[4] A. E. Brouwer, R. Pellikaan, E. R. Verheul, Doing more with fewer bits, Proceedings Asiacrypt99, LNCS 1716, Springer-Verlag 1999, 321-332.

[5] H. Cohen, A. Miyaji, T. Ono, Efficient elliptic curve exponentation usinh mixed coordinates, Proceedings Asiacrypt'98, LNCS 1514, Springer-Verlag 1998, 51-65.

[6] Martijn Stam, Arjen K. Lenstra, Speeding up XTR. Advances in Cryptology - ASIACRYPT 2001 Proceedings, December 9-13, 2001, 125–143.