THE DISCRETE LOGARITHM PROBLEM IN NON-REPRESENTABLE RINGS

MATAN BANIN AND BOAZ TSABAN

Abstract. Bergman's Ring E_p, parameterized by a prime number p, is a ring with p^5 elements that cannot be embedded in a ring of matrices over any commutative ring. This ring was discovered in 1974. In 2011, Climent, Navarro and Tortosa described an efficient implementation of E_p using simple modular arithmetic, and suggested that this ring may be a useful source for intractable cryptographic problems.

We present a deterministic polynomial time reduction of the Discrete Logarithm Problem in E_p to the classical Discrete Logarithm Problem in $\mathbb{Z}_p$, the p-element field. In particular, the Discrete Logarithm Problem in E_p can be solved, by conventional computers, in sub-exponential time.

Along the way, we collect a number of useful basic reductions for the toolbox of discrete logarithm solvers.

1. Introduction

For Discrete Logarithm based cryptography, it is desirable to find efficiently implementable groups for which sub-exponential algorithms for the Discrete Logarithm Problem are not available. Thus far, the only candidates for such groups seem to be (carefully chosen) groups of points on elliptic curves [5, 7]. Groups of invertible matrices over a finite field, proposed in [8], where proved by Menezes and Wu [6] inadequate for this purpose. Consequently, any candidate for a platform group for Discrete Logarithm based cryptography must not be efficiently embeddable in a group of matrices.

In 1974, Bergman proved that the ring $\mathrm{End}(\mathbb{Z}_p \times \mathbb{Z}_{p^2})$ of endomorphisms of the group $\mathbb{Z}_p \times \mathbb{Z}_{p^2}$, where p is a prime parameter, admits no embedding in any ring of matrices over a commutative ring. In 2011, Climent, Navarro and Tortosa [3] described an efficient implementation of E_p (reviewed below), proved that uniformly random elements of E_p are invertible with probability greater than $1 - 2/p$, and supplied an efficient way to sample the invertible elements of E_p uniformly at random. Consequently, they proposed this ring as a potential source for intractable cryptographic problems. Climent et al. proposed a Diffie–Hellman type key exchange protocol over on E_p, but it was shown by Kamal and Youssef [4] not to be related to the Discrete Logarithm Problem, and to be susceptible to a polynomial time attack.

We consider the Discrete Logarithm Problem in E_p. In light of the above-mentioned feature of E_p, the Menezes–Wu reduction attack [6] is not directly applicable. We present, however, a deterministic polynomial time reduction of the Discrete Logarithm Problem in E_p to the classical Discrete Logarithm Problem in $\mathbb{Z}_p$, the p-element field. In particular, the Discrete Logarithm Problem in E_p can be solved by conventional computers in subexponential time, and E_p offers no advantage, over $\mathbb{Z}_p$, for cryptography based on the Discrete Logarithm Problem.

2. Computing discrete logarithms in $\mathrm{End}(\mathbb{Z}_p \times \mathbb{Z}_{p^2})$

Climent, Navarro and Tortosa [3] provide the following faithful representation of Bergman's Ring. The elements of E_p are the matrices

$$g = \begin{pmatrix} a & b \\ cp & v + up \end{pmatrix}, \quad a, b, c, u, v \in \{0, ..., p-1\}.$$

Addition (respectively, multiplication) is defined by first taking ordinary addition (respectively, multiplication) over the integers, and then reducing each element of the first row modulo p, and each element of the second row modulo p^2. The ordinary zero and identity integer matrices serve are the additive and multiplicative neutral elements of E_p, respectively. The element g is invertible in E_p if and only if $a, v \neq 0$.

The set of invertible elements in a ring R is denoted R^*.

Definition 1. The *Discrete Logarithm Problem* in a ring R is to find x given an element $g \in R^*$ and its power g^x, where $x \in \{0, 1, \ldots, |g| - 1\}$.

Another version of the Discrete Logarithm Problem asks to find any $\tilde{x}$ such that $g^{\tilde{x}} = g^x$. The reductions given below are applicable, with minor changes, to this version as well, but we note in Appendix B that the two versions are essentially equivalent.

By the standard amplification techniques, one can increase the success probability of any discrete logarithm algorithm with non-negligible success probability to become arbitrarily close to 1. Thus, for simplicity, we may restrict attention to algorithms that never fail. For ease of digestion, we present our solution to the Discrete Logarithm Problem in E_p by starting with the easier cases, and gradually building up. Not all of the easier reductions are needed for the main ones, but they do contain some of the important ingredients of the main ones, and may also be of independent interest to some readers.

2.1. Basic reductions.

Reduction 2. *Computing the order of an element in R^*, using discrete logarithms in R.*

Details. For $g \in R^*$, $g^{-1} = g^{|g|-1}$. Thus, $|g| = \log_g(g^{-1}) + 1$. □

Reduction 3. *Computing discrete logarithms in a product of rings using discrete logarithms in each ring separately.*

Details. For rings R, S, $(R \times S)^* = R^* \times S^*$. Let $(g, h) \in R^* \times S^*$ and $(g, h)^x = (g^x, h^x)$, where $x \in \{1, \ldots, |(g, h)|\}$, be given. Compute

$$\begin{aligned} x \bmod |g| &= \log_g(g^x); \\ x \bmod |h| &= \log_h(h^x). \end{aligned}$$

Use Reduction 2 to compute $|g|$ and $|h|$. Compute, using the Chinese Remainder Algorithm,

$$x \bmod \mathrm{lcm}(|g|, |h|) = x \bmod |(g, h)| = x. \quad \square$$

The *Euler isomorphism* is the function

$$\begin{array}{rcl}\Phi_p\colon (\mathbb{Z}_p,+)\times(\mathbb{Z}_p^*,\cdot) & \to & \mathbb{Z}_{p^2}^* \\ (a,b) & \mapsto & (1+ap)\cdot b^p \bmod p^2.\end{array}$$

The function Φ_p is easily seen to be an injective homomorphism between groups of equal cardinality, and thus an isomorphism of groups (cf. Paillier [9] in a slightly more involved context).

Reduction 4. *Inversion of the Euler isomorphism Φ_p, using discrete logarithms in $\mathbb{Z}_p$.*

Details. Given $c\in\mathbb{Z}_{p^2}^*$, let $a\in\mathbb{Z}_p, b\in\mathbb{Z}_p^*$ be such that $c=(1+ap)b^p \bmod p^2$. Then

$$c=(1+ap)\cdot b^p = 1\cdot b^p = b \pmod p.$$

Compute $b=c \bmod p$, then $b^p \bmod p^2$, then $1+ap = c\cdot(b^p)^{-1} \mod p^2$, where the inverse is in $\mathbb{Z}_{p^2}^*$. Since $1+ap<p^2$, we can subtract 1 and divide by p to get a. □

Reduction 5. *Computing discrete logarithms in $\mathbb{Z}_{p^2}$ using discrete logarithms in $\mathbb{Z}_p$.*

Details. Use Reduction 4 to transform the problem into a computation of a discrete logarithm in $(\mathbb{Z}_p,+)\times(\mathbb{Z}_p^*,\cdot)$. Computing discrete logarithm in $(\mathbb{Z}_p,+)$ is trivial. Apply Reduction 3. □

2.2. Algebraic lemmata.

Definition 6. $\tilde{E}_p$ is the ring of matrices $\begin{pmatrix} a & b\\ pc & v\end{pmatrix}$, where addition and multiplication are carried out over $\mathbb{Z}$, and then entry $(2,1)$ is reduced modulo p^2, and the other three entries are reduced modulo p.

Lemma 7. *The map*

$$\begin{array}{rcl} E_p & \to & \tilde{E}_p; \\ \begin{pmatrix} a & b\\ cp & v+up\end{pmatrix} & \mapsto & \begin{pmatrix} a & b\\ cp & v\end{pmatrix}\end{array}$$

is a ring homomorphism.

Proof. Since addition is component-wise, it remains to verify multiplicativity. Indeed, in E_p,

$$\begin{pmatrix} a_1 & b_1\\ c_1p & v_1+u_1p\end{pmatrix}\begin{pmatrix} a_2 & b_2\\ c_2p & v_2+u_2p\end{pmatrix} = \begin{pmatrix} a_1a_2 & a_1b_2+b_1v_2\\ (c_1a_2+v_1c_2)p & v_1v_2+(c_1b_2+v_1u_2+u_1v_2)p\end{pmatrix},$$

and in $\tilde{E}_p$,

$$\begin{pmatrix} a_1 & b_1\\ c_1p & v_1\end{pmatrix}\begin{pmatrix} a_2 & b_2\\ c_2p & v_2\end{pmatrix} = \begin{pmatrix} a_1a_2 & a_1b_2+b_1v_2\\ (c_1a_2+v_1c_2)p & v_1v_2\end{pmatrix}. \quad \square$$

Lemma 8. *Let $g=\begin{pmatrix} a & b\\ cp & v\end{pmatrix}\in\tilde{E}_p^*$, and let x be a natural number. Define $d_x\in\mathbb{Z}_p$ by*

$$d_x=\begin{cases}\frac{a^x-v^x}{a-v} & a\neq v\\ xa^{x-1} & a=v.\end{cases}$$

Then

$$g^x = \begin{pmatrix} a^x & bd_x \\ cd_x p & v^x \end{pmatrix}.$$

Proof. By induction on x. The statement is immediate when $x = 1$. Induction step: If $a \neq v$, then in $\mathbb{Z}_p$,

$$\begin{aligned} a^x + d_x v &= a^x + \frac{a^x - v^x}{a - v} \cdot v = \frac{a^x(a-v) + (a^x - v^x)v}{a-v} = \frac{a^{x+1} - v^{x+1}}{a-v} = d_{x+1}; \\ ad_x + v^x &= \frac{a(a^x - v^x)}{a-v} + \frac{(a-v)v^x}{a-v} = \frac{a^{x+1} - v^{x+1}}{a-v} = d_{x+1}. \end{aligned}$$

If $a = v$, then

$$\begin{aligned} a^x + d_x v &= a^x + xa^{x-1}v = a^x + xa^{x-1}a = a^x + xa^x = (x+1)a^x = d_{x+1}; \\ ad_x + v^x &= xa^x + a^x = (x+1)a^x = d_{x+1}. \end{aligned}$$

Thus, in either case,

$$g^{x+1} = g^x \cdot g = \begin{pmatrix} a^x & bd_x \\ cd_x p & v^x \end{pmatrix} \cdot \begin{pmatrix} a & b \\ cp & v \end{pmatrix} = \begin{pmatrix} a^{x+1} & b(a^x + d_x v) \\ c(ad_x + v^x)p & v^{x+1} \end{pmatrix} = \begin{pmatrix} a^{x+1} & bd_{x+1} \\ cd_{x+1}p & v^{x+1} \end{pmatrix}.$$

□

Lemma 9. *Let* $g = \begin{pmatrix} a & b \\ cp & v \end{pmatrix} \in \tilde{E}_p^*$.

1. *If* $a = v$ *and at least one of* b, c *is nonzero, then* $|g| = p \cdot |a|$.
2. *In all other cases* ($a \neq v$ *or* $b = c = 0$), $|g| = \operatorname{lcm}(|a|, |v|)$.

Proof. Define d_x as in Lemma 8. By Lemma 8,

$$\begin{pmatrix} a^{|g|} & * \\ * & v^{|g|} \end{pmatrix} = g^{|g|} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Thus, $|a|$ and $|v|$ divide $|g|$, and therefore so does $\operatorname{lcm}(|a|, |v|)$.

We consider all possible cases.

If $b = c = 0$, then

$$g^x = \begin{pmatrix} a^x & 0 \\ 0 & v^x \end{pmatrix}$$

for all x, and thus $|g| = \operatorname{lcm}(|a|, |v|)$, as claimed in (2).

Assume, henceforth, that at least one of b, c is nonzero, and let

$$l = \operatorname{lcm}(|a|, |v|).$$

If $a \neq v$, then

$$d_l = \frac{a^l - v^l}{a - v} = \frac{1 - 1}{a - v} = 0 \mod p,$$

and thus, by Lemma 8, $g^l = I$. Thus, $|g|$ divides l, which we have seen to divide $|g|$. It follows that $|g| = l$, as claimed in (2).

Assume, henceforth, that $a = v$.

Since $d_p = pa^{p-1} = 0 \mod p$, we have by Lemma 8 that

$$g^p = \begin{pmatrix} a^p & 0 \\ 0 & a^p \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}.$$

It follows that $g^{p\cdot|a|} = I$. Therefore, $|g|$ divides $p \cdot |a|$. Recall that $|a|$ divides $|g|$. Now, $d_{|a|} = |a| \cdot a^{|a|-1} \bmod p$. Since $|a| < p$, $d_{|a|} \neq 0$. It follows that

$$g^{|a|} = \begin{pmatrix} a^{|a|} & bd_{|a|} \\ cd_{|a|}p & a^{|a|} \end{pmatrix} \neq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

and thus $|g| = p \cdot |a|$, as claimed in (1). □

2.3. **The main reductions.**

Reduction 10. *Computing discrete logarithms in $\tilde{E}_p$ using discrete logarithms in $\mathbb{Z}_p$.*

Details. Let $g = \begin{pmatrix} a & b \\ cp & v \end{pmatrix} \in \tilde{E}_p^*$, and let $x \in \{1, \dots, |g|\}$. By Lemma 8,

$$g^x = \begin{pmatrix} a^x & bd_x \\ cd_xp & v^x \end{pmatrix}.$$

If $a \neq v$ or $b = c = 0$, then by Lemma 9, $|g| = \mathrm{lcm}(|a|, |v|)$. Compute

$$\begin{aligned} x \mod |a| &= \log_a(a^x); \\ x \mod |v| &= \log_v(v^x). \end{aligned}$$

Since $x < |g|$, we can use the Chinese Remainder Algorithm to compute $x \bmod \mathrm{lcm}(|a|, |v|) = x$.

Thus, assume that $a = v$ and one of b, c is nonzero. By Lemma 9, $|g| = p \cdot |a|$. Compute

$$x_0 := x \mod |a| = \log_a(a^x).$$

Compute

$$g^x \cdot g^{-x_0} = g^{x-x_0} = \begin{pmatrix} a^{x-x_0} & bd_{x-x_0} \\ cd_{x-x_0}p & a^{x-x_0} \end{pmatrix} = \begin{pmatrix} 1 & bd_{x-x_0} \\ cd_{x-x_0}p & 1 \end{pmatrix}.$$

Since b or c is nonzero, we can extract $d_{x-x_0} \bmod p$. Compute

$$d_{x-x_0} \cdot a = (x - x_0)a^{x-x_0} = x - x_0 \mod p.$$

As $x - x_0 \leq x < |g| = p \cdot |a|$, we can use the Chinese Remainder Algorithm to compute

$$x - x_0 \bmod \mathrm{lcm}(p, |a|) = x - x_0 \bmod p \cdot |a| = x - x_0.$$

Add x_0 to obtain x. □

Reduction 11. *Computing discrete logarithms in E_p using discrete logarithms in $\mathbb{Z}_p$.*

Details. Let $g = \begin{pmatrix} a & b \\ cp & v + up \end{pmatrix} \in E_p^*$, and let $x \in \{1, \dots, |g|\}$. Take $\bar{g} = \begin{pmatrix} a & b \\ cp & v \end{pmatrix} \in \tilde{E}_p^*$. Use Lemma 9 and Reduction 2 to compute $|\bar{g}|$. By Lemma 7, $|\bar{g}|$ divides $|g|$, and

$$g^{|\bar{g}|} = \begin{pmatrix} 1 & 0 \\ 0 & 1 + sp \end{pmatrix}$$

for some $s \in \{0, \dots, p-1\}$. If $s = 0$ then $|g| = |\bar{g}|$, and thus $x_0 := \log_{\bar{g}}(\bar{g}^x) = \log_g(g^x) = x$, and we are done.

Thus, assume that $s \neq 0$. Using Reduction 10, compute

$$x_0 := \log_{\bar{g}}(\bar{g}^x) = x \bmod |\bar{g}|.$$

Let $q = (x - x_0)/|\bar{g}|$. Since the order of $1 + sp$ in $\mathbb{Z}_{p^2}$ is p (in $\mathbb{Z}_{p^2}$, $(1+sp)^e = 1 + esp$ for all e), the order of $g^{|\bar{g}|}$ is p, and thus $|g| = |\bar{g}| \cdot p$. Thus, $q \leq x/|\bar{g}| < |g|/|\bar{g}| = p$.

Since $x - x_0 \leq x < |g| = |\bar{g}| \cdot p$, $q := (x - x_0)/p < |\bar{g}|$. Compute

$$g^x g^{-x_0} = g^{x-x_0} = (g^{|\bar{g}|})^q = \begin{pmatrix} 1 & 0 \\ 0 & 1+sp \end{pmatrix}^q = \begin{pmatrix} 1 & 0 \\ 0 & (1+sp)^q \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1+sqp \end{pmatrix}.$$

Subtract 1 from $1 + sqp$. Divide by p to get $sq \bmod p$. In $\mathbb{Z}_p$, multiply by s^{-1} to obtain $q \bmod p = q$. Multiply by $|\bar{g}|$ to get $x - x_0$, and add x_0. □

3. Summing up: Code

Following is a self-explanatory code (in Magma [2]) of our main reductions. This code shows, in a concise manner, that the number of computations of discrete logarithms in $\mathbb{Z}_p$ needed to compute discrete logarithms in Bergman's Ring E_p is *at most 2*. For completeness, we provide, in Appendix A, the basic routines.

```
F := GaloisField(p);
Z := IntegerRing();
I :=  ScalarMatrix(2, 1); //identity matrix

function EpTildeOrder(g) //Lemma 9
    a := F!(g[1,1]);
    v := F!(g[2,2]);
    if (a ne v) or (IsZero(g[1,2]) and IsZero(g[2,1])) then
        order := Lcm(Order(a),Order(v));
    else
        order := p*Order(a);
    end if;
    return order;
end function;

function EpTildeLog(g,h) //Reduction 10
    a := F!(g[1,1]);
    b := F!(g[1,2]);
    c := F!(g[2,1] div p);
    v := F!(g[2,2]);
    x0 := Log(a,F!(h[1,1]));
    if (a ne v) or (IsZero(b) and IsZero(c)) then
        xv := Log(v,F!(h[2,2]));
        x := ChineseRemainderTheorem([x0,xv], [Order(a),Order(v)]);
    else
        ginv := EpTildeInverse(g);
```

```
        f := EpTildePower(ginv,x0);
        f := EpTildeProd(h,f);
        if IsZero(c) then
            d := b^-1 * F!(f[1,2]);
        else
            d := c^-1 * F!(f[2,1] div p);
        end if;
        delta := Z!(d*a);
        truedelta := ChineseRemainderTheorem([0,delta],[Order(a),p]);
        x := truedelta+x0;
    end if;
    return x;
end function;

function EpLog(g,h) //Reduction 11
    gbar := Bar(g); hbar := Bar(h);
    gbarorder := EpTildeOrder(gbar);
    x0 := EpTildeLog(gbar,hbar);

    f := EpPower(g,gbarorder);
    s := (f[2,2]-1) div p;

    if IsZero(s) then
        x := x0;
    else
        ginv := EpInverse(g);

        f := EpPower(ginv,x0);
        f := EpProd(h,f);
        n := (f[2,2]-1) div p;
        q := (F!s)^-1*F!n;
        x := gbarorder*(Z!q)+x0;
    end if;
    return x;
end function;
```

We have tested these routines extensively: For random primes of size $4, 8, 16, 32, 64$, and 128 bits, and thousands of random pairs $g, h = g^x$, `EpLog(g,h)` always returned x.

References

[1] G. Bergman, *Examples in PI ring theory*, Israel Journal of Mathematics **18** (1974), 257-277.

[2] W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system, I: The user language*, Journal of Symbolic Computation **24** (1997), 235–265.

[3] J. Climent, P. Navarro, L. Tortosa, *On the arithmetic of the endomorphisms ring* $\mathrm{End}(\mathbb{Z}_p \times \mathbb{Z}_{p^2})$, Applicable Algebra in Engineering, Communication and Computing **22** (2011), 91–108.

[4] A. Kamal, A. Youssef, *Cryptanalysis of a key exchange protocol based on the endomorphisms ring* $\mathrm{End}(\mathbb{Z}_p \times \mathbb{Z}_{p^2})$, Applicable Algebra in Engineering, Communication and Computing, to appear.
[5] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation **48** (1987), 203–209.
[6] A. Menezes, Y. Wu, *The discrete logarithm problem in* $\mathrm{GL}(n, q)$, Ars Combinatoria, 47 (1998), 23–32.
[7] V. Miller, *Uses of elliptic curves in cryptography*, in: Advances in Cryptology–Proceedings of Crypto '85. Lecture Notes in Computer Science **218** (1986), 417–426.
[8] R. Odoni, R. Sanders, V. Varadharajan, *Public key distribution in matrix rings*, Electronic Letters **20** (1984), 386–387.
[9] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, in: J. Stern, ed., Advances in Cryptology – EUROCRYPT'99, Lecture Notes in Computer Science **1592** (1999), 223–238.

Appendix A. Elementary routines

To remove any potential ambiguity, and help readers interested reproducing our experiments, we provide here the basic routines for arithmetic in Bergman's Ring E_p.

```
function EpProd(A, B) //integer matrices
    C := A*B;
    C[1,1] mod:= p;
    C[1,2] mod:= p;
    C[2,1] mod:= p^2;
    C[2,2] mod:= p^2;
    return C;
end function;

function Bar(g)
    h := g;
    h[2,2] mod:= p;
    return h;
end function;

function EpTildeProd(A, B) //integer matrices
    return Bar(EpProd(A,B));
end function;

function EpInvertibleEpMatrix()
    g := ZeroMatrix(Z, 2, 2);
    g[1,1] := Random([1..p-1]);
    g[1,2] := Random([0..p-1]);
    g[2,1] := p*Random([0..p-1]);
    g[2,2] := Random([1..p-1])+p*Random([1..p-1]);
    return g;
end function;

function EpPower(g, n) //square and multiply
     result := I;
     while not IsZero(n) do
```

```
        if ((n mod 2) eq 1) then
            result := EpProd(result, g);
            n -:= 1;
        end if;
        g := EpProd(g, g);
        n div:= 2;
     end while;
     return result;
end function;

function EpTildePower(g, n)
    return Bar(EpPower(g, n));
end function;

function EpInverse(g)
    a := F!(g[1,1]);
    b := F!(g[1,2]);
    c := F!(g[2,1] div p);
    u := F!(g[2,2] div p);
    v := F!(g[2,2]);

    ginv := ZeroMatrix(Z,2,2);
    ginv[1,1] := Z!(a^-1);
    ginv[1,2] := Z!(-a^-1*b*v^-1);
    ginv[2,1] := p*Z!(-v^-1*c*a^-1);
    ginv[2,2] := Z!(v^-1)+
            p*Z!(c*a^-1*b*v^-2-u*v^-2-(F!(Z!v*Z!(v^-1) div p)*v^-1));
    return ginv;
end function;

function EpTildeInverse(g)
    return Bar(EpInverse(g));
end function;
```

Appendix B. Equivalence of Discrete Logarithm Problems

Consider the following two versions of the Discrete Logarithm Problem in a finite group G.

DLP1: Find x, given an element $g \in G$ and its power g^x, where $x \in \{0, 1, \dots, |g| - 1\}$.
DLP2: Given an element $g \in G$ and its power g^x, find $\tilde{x}$ with $g^{\tilde{x}} = g^x$.

DLP1 is harder than DLP2: A DLP1 oracle returns $\tilde{x} := x \bmod |g|$ on input g, g^x. On the other hand, DLP2 is probabilistically harder than DLP2: It suffices to show how $|g|$ can be computed using a DLP2 oracle. Indeed, for a large enough (but polynomial) number of random elements $r \in \{|G|, |G| + 1, \dots, M\}$ where $M \gg |G|$ is fixed, let $\tilde{r}$ be the output of DLP2 on (g, g^r). Then $|g|$ divides all numbers $(r - \tilde{r}) \bmod g$, and the greatest common divisor of these numbers is $|g|$, except for a negligible probability.

Department of Mathematics, Bar-Ilan University, Ramat Gan 52900, Israel
E-mail address: baninmmm@gmail.com, tsaban@math.biu.ac.il
URL: http://www.cs.biu.ac.il/~tsaban