1B1-03

ElGamal暗号を用いたSecure DisCSPアルゴリズムの実装と評価

Implemention and evaluation of Secure DisCSP algorithm using ElGamal Encryption

黒田直樹[*1] 横尾真[*2] 岩崎敦[*2]
Naoki Kuroda Makoto Yokoo Atsushi iwasaki

[*1]九州大学大学院システム情報科学府 [*2]九州大学大学院システム情報科学院
Faculty of ISSE, Kyushu University Graduate School of ISEE, Kyushu University

In this paper, we implement a secure distributed Constraint Satisfaction (DisCSP) algorithm on real machines and evaluate its performance experimentally. In a DisCSP, variables and constraints are distributed among multiple agents. Although one major motivation for solving a DisCSP without gathering all information in one server is the concern about privacy/security, traditional DisCSP algorithms leak some information during the search process and privacy/security issues are not dealt with formally. Recently, a Secure DisCSP algorithm that utilizes a public key encryption scheme has been developed to circumvent such information leakage.

In this paper, we measure the execution time and the number of communications between the search-controller and the decryptor. Our experimental results showed that the execution time critically depends on the number of communications. Also, we show that the performance can be significantly improved by aggregating multiple cryptogram/plain texts into a single message.

1. はじめに

制約充足問題 (Constraint Satisfaction Problem, CSP) とは，有限で離散的な領域から値を取る複数の変数に対して，制約を満足する値の割当を発見する問題であり，人工知能の様々な問題が CSP により定式化できることが知られている．さらに分散制約充足問題（Distributed Constraint Satisfaction Problem, DisCSP）は，分散したエージェント達がそれぞれ変数と制約をもつ CSP であり，この DisCSP によってマルチエージェントシステムの様々な応用問題を定式化可能にする枠組みとして注目されており，様々な DisCSP アルゴリズムが提案されている．

分散環境において CSP を解く意義の 1 つは各エージェントがもつ個人情報のプライバシやセキュリティを確保することにある．実際，DisCSP では全ての個人情報を 1 つのサーバに集積させずに問題を解く点においてセキュリティが高いと言える．しかし，問題を解く為には個人情報を通信する必要があるため，サーバや他のエージェントにある程度個人情報を送る必要がある．そのため，この点においては従来の DisCSP アルゴリズムはセキュリティ上の問題をもつと言える．

一方で，横尾らは Secure DisCSP アルゴリズムを提案している [Yokoo 02]．これは ElGamal 暗号を用いて制約情報の暗号化や並び替えを行うことで，情報の秘匿性を理論的に保証するアルゴリズムである．

本論文ではこの Secure DisCSP アルゴリズムを実装することで，セキュリティを強化したことによる実用上の問題点を吟味する．具体的には，実行時間や解の探索を行う search-controller と暗号文の解読を行う decryptor 間の通信時間を測定し，通信時間が実行時間におけるもっとも大きなボトルネックとなっていることを明らかにした．加えて，複数の暗号文/平文をあらかじめまとめて処理することで，通信回数を減らし，その結果，実行時間の短縮が可能なことを示した．

本論文の構成を以下に示す．まず 2 章では DisCSP の定式化を行い，3 章で ElGamal 暗号について述べ，4 章で Secure DisCSP アルゴリズムを概説する．5 章で実験設定と評価を行い，最後に 6 章で結論を述べる．

連絡先: 黒田直樹，九州大学大学院システム情報科学府，812-8581 福岡県福岡市東区箱崎 6-10-1，(092)642-3882，kuroda@agent.is.kyushu-u.ac.jp

2. 分散制約充足問題（DisCSP）の定式化

本章では，DisCSP（分散制約充足問題）を定式化し，例を挙げながら説明する．

2.1 DisCSP の定式化

本論文で取り扱う DisCSP を以下のように定式化する．

- エージェント 1, 2, . . . , I が存在する．
- あるエージェントに属する変数 $x_1, x_2, \ldots, x_n$ があり，x_i がエージェント a に属するとき，(x_i, a) と表す．
- すべての変数は共通のドメイン $D = \{1, 2, \ldots, m\}$ を持ち，この変数の定義域はエージェント間の共通の知識である．
- 1 変数からなる制約と 2 変数からなる制約がある．
- 制約は *nogood* の集合として表される．1 変数からなる $nogood\{x_i = d_i\}$ は x_iに対する d_i という割当が制約を満たさないことを表す．2 変数からなる $nogood\{x_i = d_i, x_j = d_j\}$ は $x_i = d_i$かつ，$x_j = d_j$の割当が制約を満たさないことを表す．
- これらの制約はエージェントのプライベートな情報である．a に属する変数 x_iに関する 1 変数からなる制約は a だけが知っている．このような制約は $C^a_{x_i}$ と表す．またエージェント a, b に属する変数 x_i, x_jに関する 2 変数からなる制約 について，エージェント a が持つ制約は，$C^a_{x_i,x_j}$，エージェント b が持つ制約は $C^b_{xi,xj}$と表す．
- DisCsp の解 $S = \{x_1 = s_1, x_2 = s_2, \ldots, x_n = s_n\}$ は全ての制約を満たす，全ての変数の割当である．

これらの定式化は，[Yokoo 92] で示されている伝統的な DisCSP の定義を基盤としている．[Yokoo 92] の定義と若干異なるのは，[Yokoo 92] の定義では，変数のドメインは変

数ごとに異なり，プライベートな情報であることである．一方，上記の DisCSP の定式化では，ドメインは全ての変数で同じであり，共通の情報である．しかし，上記の DisCSP の定式化では，エージェントはプライベートな 1 変数からなる制約を持つことができるため，[Yokoo 92] の DisCSP の定義と大きな違いは無い．

2.2 DisCSP の例題

分散 4-queen 問題を例題に挙げる．この問題は，4 行 4 列のマスの中で，各行に 1 対づつ存在するクイーン同士が，お互いを殺しあわないような位置を探す問題である．この問題は以下のように，DisCSP で定式化できる．

- エージェント $1, 2, 3, 4$ があり，各エージェント i は，一つの変数 x_i を持ち，ドメイン $D = \{1, 2, 3, 4\}$ を持つ．
- この問題では，1 変数の制約は無い．
  $i < j$ として，$C^i_{x_i,x_j}$ について，エージェント i が持つ制約は斜めの制約のみとすると，$C^1_{x_1,x_2}$ は $nogood\{x_1 = 1, x_2 = 2\}$ 等を含む制約となり，$C^j_{x_i,x_j}$ について，エージェント j が持つ制約は縦の制約のみとすると，$C^2_{x_1,x_2}$ は $nogood\{x_1 = 1, x_2 = 1\}$ 等を含む制約となる．
- この問題の解は $S = \{x_1 = 3, x_2 = 1, x_3 = 4, x_4 = 2\}$ のようになる．

3. ElGamal 暗号

本章では，ElGamal 暗号 [ElGamal 85] について説明する．ElGamal 暗号は識別不可能性および準同形性の性質を持ち，ランダム化の操作が可能な公開鍵暗号である．

3.1 ElGamal 暗号の概要

ElGamal 暗号は公開鍵暗号の一種である．ElGamal 暗号は離散対数問題の困難さに基づいて作られている．離散対数問題とは，整数 g, p, z が与えられて $z = g^x (mod\ p)$ となる x を求める問題である．g, p, x が与えられて z を求めるのは簡単だが，g, p, z を与えられて x を求めるのは難しい．

公開鍵の生成

まず，非常に大きな素数 p をとる．これが法となる．次に，p を法とする原子根 g をとる．g が p を法とする原子根である時，$g^z (mod\ p) = 1$ は $z = p - 1$ の時にはじめて満たされる．さらに，$1 \leq s \leq p - 1$ となる数 s を決定する．そして，$y = g^s (mod\ p)$ によって，y を計算する．こうして，公開鍵 $= \{p, g, y\}$ と秘密鍵 $= \{s\}$ が出来る．公開鍵は公開され，秘密鍵は暗号受信者が秘密に保持する．

暗号化の手順

暗号送信者は $1 \leq r \leq p - 1$ となるランダムな値 r をとり，平文 (M) $(0 \leq M \leq p - 1)$ に対して，$A = g^r (mod\ p)$，$B = My^r (mod\ p)$ を計算する．そして，$E(M) = (A, B)$ が暗号文となり，これを暗号受信者に送る．

復号化の手順

復号は暗号解読者の保持する秘密鍵 s を使って行われる．$A^{-s}B (mod\ p) = M$ によって平文 (M) は復号される．

3.2 ElGamal 暗号の例

簡単な ElGamal 暗号の例題を上げる．今，公開鍵=$\{p = 23, g = 3, y = 9\}$ および秘密鍵=$\{s = 2\}$ とする．そして，平文が $M = 5$ であるとする．さらに暗号化する際に使用する乱数として $r = 2$ を用いると，$A = g^r (mod\ p) = 3^2 (mod\ 23) = 9$，$B = My^r (mod\ p) = 5 * 9^2 (mod\ 23) = 405 (mod\ 23) = 14$ となるので，暗号文は $E(M) = (9, 14)$ となる．

最後に暗号文 $E(M)$ を秘密鍵を使って解読する．$A^{-s}B (mod\ p) = 14/9^2 (mod\ 23) = 5 = M$．

このように，暗号化および復号化が行われる．

4. Secure DisCSP アルゴリズム

この章では，Secure DisCSP アルゴリズムの詳細について説明する．以下，表記上の簡単化の為に次の条件をつける．

- 各エージェント i はただ一つの変数 x_i を持つ．
- 全ての変数の組に対して 2 変数間制約が存在する．

4.1 基本方針

Secure DisCSP アルゴリズムでは，解の探索を行う search-controller と暗号の解読を行う decrytor の 2 つが計算サーバとして用いられる．各エージェントは decryptor が公開する公開鍵を用いて，制約を暗号化した後，サーバに転送する．制約を受け取ったサーバは解を探索した後，変数の割当を各エージェントに転送する．Secure DisCSP アルゴリズムでは，エージェントおよびサーバに個人情報を漏洩させることなく解を求めなくてはならない．

このような要求を，Secure DisCSP アルゴリズムでは，以下の方法によって満足させる．(1) 各エージェントが制約を ElGamal 暗号によって暗号化する．(2) 各エージェントが，制約行列の並び替えを行う．(変数の値の入れ替えをする．) (3) サーバは直接制約を解読すること無しに，問題を解く．残りの節で，これらの手法の詳細を説明する．

4.2 制約の暗号化

まず，各エージェントは制約を暗号化する．エージェント i は，x_i と x_j 間の 2 変数制約 $C^i_{x_i,x_j}$ を $m \times m$ の制約行列として表す．この行列を $A_{i,x_i,x_j} (i \neq j)$ として表記する．この行列の要素 $(A_{i,x_i,x_j}(k, l))$ は $E(1)$，もしくは $E(z)$ である．$E(1)$ と $E(z)$ は 1 と $z \neq 1$ の暗号文である．$A_{i,x_i,x_j} (i \neq j)$ に入る値は以下のようになる．

- $A_{i,x_i,x_j}(k, l) = E(z)$
  $x_i = k$ かつ，$x_j = l$ が制約を満たさないとき．($C^i_{x_i,x_j}$ 内に $nogood\{x_i = k, x_j = l\}$ があるか，単変数制約 $C^i_{x_i}$ 内に $nogood\{x_i = k\}$ があるとき)
- $A_{i,x_i,x_j}(k, l) = E(1)$
  $x_i = k$ かつ，$x_j = l$ が制約を満たすとき．

図 1 は，2 章 2 節で例を上げた分散 4-queen 問題の x_1 と x_2 間の制約行列を，エージェント 1 と 2 が暗号化したものを示す．エージェント 1 は x_1 と x_2 間の斜めの制約を知っており，エージェント 2 は x_1 と x_2 間の縦の制約を知っているとする．k 行目，l 列目 の要素は，$x_1 = k$ かつ，$x_2 = l$ が $nogood$ ならば，$E(z)$ が入り，そうでないなら，$E(1)$ が入る．識別不可能性の性質により，暗号文を解読しない限り，元の平文の同異がどうであれ，全て異なってみえる．そして，エージェント 2 は，A_{2,x_i,x_j} をエージェント 1 に送信する．エージェント 1 は自分の持つ A_{1,x_i,x_j} と，受信した A_{2,x_i,x_j} の各要素との積を取って，新しい制約行列 A_{x_1,x_2} を得る．準同形の性質によって，暗号文 $(E(M))$ を解読すること無しに，平文 (M) の積をとることができる．

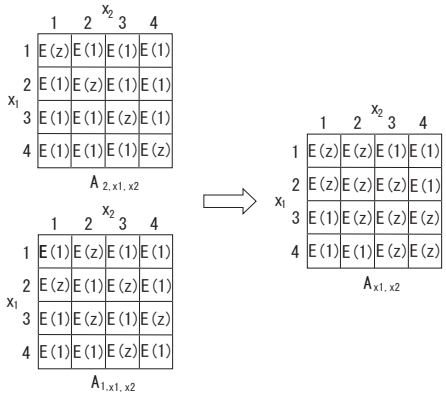

図 1: 制約行列の例

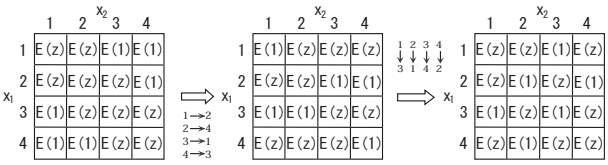

図 2: 制約行列の並び替えの例

4.3 制約行列の並び替え

次にエージェントは各制約行列の並び替えを行う．詳しく言うと，各変数 x_i について，その値 k を $\pi_{x_i}(k)$ によって置き換える．π_{x_i} は $1, 2, \ldots, m$ から $1, 2, \ldots, m$ に m 要素を全単射する関数である．変数の値を変えることによって，各エージェントが元の問題を新しい問題へと変えることになり，サーバへの情報漏洩を防ぐことができる．その結果をエージェント i に送信する．図 2 は前節で作られた，分散 4-queen 問題の制約行列 $A_{1,2}$ に対して，エージェント 1 がまず，$\pi_{x_1}(1) = 2, \pi_{x_1}(2) = 4, \pi_{x_1}(3) = 1, \pi_{x_1}(4) = 3$ によって並び替えを行い，次にエージェント 2 が，$\pi_{x_2}(1) = 3, \pi_{x_2}(2) = 1, \pi_{x_2}(3) = 4, \pi_{x_2}(4) = 2$ によって $A_{1,2}$ を並び替える様子を示している．$A_{i,j}$ の並び替えの終了後，エージェント j は各要素に $E(1)$ を掛けるランダム化の操作を加えた後，$A_{i,j}$ を search-controller に送信する．

4.4 解の探索

本節では，search-controller と decryptor によって行われる探索の手法について示す．search-controller は探索を担当し，decryptor は解読を行う．

search-controller の動作

search-controller は同期バックトラックと同等の動きをする．暗号化されている制約の解読をするために，decryptor に暗号文を送信し，平文を受信しながら解を探索していく．部分解はそれに現れる変数の値の集合で表される．変数 x_1 から x_n まで順番に部分解との制約を満たすかどうか調べ，制約が満たされれば部分解に加え，次の変数について調べる．変数の値が部分解との制約を満たさないときはその変数の次の値を調べる．部分解に対して，次に加える変数の値が存在しなかったときは，最も新しく部分解に加えられた変数を部分解から外し，その変数の値を変えて，部分解に加えられるかどうか調べる．部分解のサイズが，変数の数 m と等しくなったとき，それが解となる．

decryptor の動作

search-controller から，暗号文 $E(M)$ が送られてきたら，decryptor は保持している秘密鍵を使って $E(M)$ を解読し，平文 (M) を search-controller に返す．

解の探索終了後

解の探索が終了したら，search-controller はその割当を各エージェントに送信する．エージェント i は，search-controller から送られてくる x_i の値 d_i を受信した後，$\pi^{-1}_{x_i}(d_i)$ によって最終的な割当を得ることができる．例えば，search-controller からエージェント 1 に送られてきた x_1 の値が 1 であったとすると，$\pi_{x_1}(3) = 1$ なので，$x_1 = 3$ が正しい割当である．

4.5 Secure DisCSP アルゴリズムの特徴

安全性

制約が暗号化されている為，エージェントは制約の積を取る段階や，並び替えの段階で，他のエージェントの制約について何ら情報を得ることはできない．また，エージェントは解の探索課程に参加できないため，エージェントが得られる情報は，そのエージェントに属する変数の割当をオラクルとして受け取るのと同じである．以上のような理由から，エージェントへの情報漏洩に対する安全性は確保される．また，各エージェントが制約行列が並び替えているため，search-controller は，真の解については知ることができない．また，search-controller は自力で暗号を解読することはできないので，制約行列を自由に解読し，変数間の制約に関する付加的な情報を自由に手に入れることはできない．decryptor は，機械的に暗号文を解読するだけであり，何ら問題に対しての知識を得ることはできない．以上のような理由から，計算に用いるサーバへの情報漏洩に対する安全性も確保される．

通信コスト

Secure DisCSP アルゴリズムの解探索方法は，同期バックトラックと同じである．したがって，search-controller と decryptor 間の暗号文/平文の交換回数は最悪の場合 $O(m^n)$ となる．m は変数のドメインサイズ，n はエージェント数である．

5. 実験と評価

本章では，Secure DisCSP アルゴリズムの実験と評価を行う．まず，実験設定について説明し，次に実験結果を示す．そして実行時間の高速化について述べ，最後に考察を述べる．

5.1 実験設定

PC を１０台用いて実験を行った．エージェント数 $I = 8$．ドメインサイズ $m = 8$ を用いた．実験の中で変化させるパラメーターとして以下の p1 および p2 を用いた $(0.0 \leq p1, p2 \leq 1.0)$．p1 は，制約関係のある変数の数を決定するパラメータである．p1=1.0 の時は全ての変数は他の全ての変数との間に制約を持つ．p2 は，制約関係のある変数との制約の強さを決定するパラメータである．p2=1.0 の時は全ての変数の値の組み合わせが制約となる．尚，本論文では，p1 については，p1=1.0 の値のみを使った．すなわち，全ての変数間どうしに制約関係がある．実験で採取する全てのデータはパラメータ p1，p2 の値によって，ランダムに１５回生成される制約についてプログラムを実行して，データを取得し，その平均値を用いる．全ての処理を行うプログラムを JAVA で記述した．ElGamal 暗号の暗号化および復号化には，R. Meyer 作成の Java Package[Mayers 03] を使用した．実験によって取得したデータは次の 2 つである．1 つ目は search-controller と decryptor 間の暗号文/平文の交換回数（以下，通信回数と表記），2 つ目は search-controller が制約情報を全て受信した時から，解を発見するまでの時間（以下，実行時間と表記）である．また本実験前に，この 2 つ

のデータ以外に，Search-Controller が decryptor に暗号文を送信してから，平文を受信するまでに要する時間 (以下，通信時間と表記）と，decryptor が暗号解読に要する時間のデータを取得した．この 2 つのデータはパラメータに依存しない，ほぼ固定値なので，本実験前に充分な回数のデータを取得し，その平均値を取って算出した．

5.2 実験結果

あらかじめ行った実験によって，通信時間は 0.324 秒，decryptor が暗号解読に要した時間は 0.001 秒という結果を得た．この結果より，通信時間はほとんど，暗号文/平文のデータの送受信に要した時間だと言える．実験結果を図 3 の“ 改良前の実験結果 ”に示す．図 3 より，実行時間は p2=0.4375 の時の実行時間約 800 秒を頂点とする急峻な山型に分布していることが分かる．また，通信回数に通信時間を掛けると，ほとんど実行時間と一致する．このことによって実行時間は，ほぼ通信時間が占めていると言える．

5.3 実行時間の高速化

本章 2 節の実験結果より，Secure DisCSP アルゴリズムの実行時間を計測できた．しかし，この実行時間は非常に大きいと言える．例えば，Decryptor との通信が必要無いように，Search-Controller に平文の制約行列が与えられた状態で，同期バックトラックによって解の探索を行うと，同じ問題条件では，ほとんど一瞬にして解の探索は終了する．Secure DisCSP アルゴリズムで，大規模な DisCSP を解くためにも，Secure DisCSP アルゴリズムをより実用的なアルゴリズムにするためにも，高速化の必要性は大きい．本章 2 節の実験結果より，Secure DisCSP アルゴリズムでは search-controller と decryptor 間の通信時間が実行時間のほとんどを占めたことが分かった．したがって，暗号文/平文の交換回数を減らすことで，実行時間の短縮を図ることができると考えられる．そこで，あらかじめバックトラック法において，次に調べる可能性のある複数の制約情報の暗号文/平文を search-controller，decryptor 間で，まとめて送受信するようプログラムに改良を加え，再度実験を行った．具体的には以下のような改良を加えた．

改良点

ある部分解がある時，次に部分解に加えようとする変数 x_i について，部分解との制約関係が不明な全ての x_i の値との制約情報をまとめて送信する．最高でドメインサイズ分（=8）の暗号文/平文をまとめて送受信することになる．

実験結果

図 3 の“ 改良後の実行時間 ”に実験結果を示す．実行時間の大きいところで比較すると，実行時間がおよそ 4 倍程度高速化できた．

5.4 考察

実験によって，Secure DisCSP アルゴリズムは，search-controller と decryptor 間の通信時間が実行時間のボトルネックとなり，実行時間が大きくなることが分かった．また，本章 3 節で示したように，複数の暗号文/平文をあらかじめまとめて処理することにより，実行時間の高速化が可能なことを示した．バックトラック法において，次に調べる可能性の高い制約情報をあらかじめ調べることで，高速化を実現できる．バックトラック法では，制約が弱い時は次々部分解のサイズが拡大される一方，制約が強い時はなかなか部分解のサイズが増えず，バックトラックが頻繁に起こる．したがって，このような性質を踏まえて，まとめて処理する暗号文/平文の組み合わせをさらに工夫することで，さらなる実行時間の高速化を実現できると考えられる．一方，一度にまとめて処理する暗号文/平文の数については，基本的に数を大きくすればするほど，実行時間は単調減少すると考えられるが，大きくし過ぎると search-controller が得る制約に関する情報が増えて，情報漏洩の観点から望ましくないので一定の限度を考えておく必要がある．

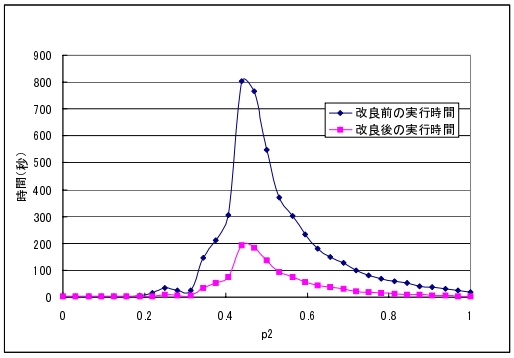

図 3: 本章 2 節と 3 節の実験の実行時間の比較

6. 結論

本論文では ElGamal 暗号を用いた Secure DisCSP アルゴリズムを実装し，評価することで，セキュリティを強化したことによる実用上の問題点を示した．具体的には search-controller と decryptor 間の通信時間や実行時間を測定し，通信時間が実行時間におけるもっとも大きなボトルネックとなっていることを明らかにした．加えて，複数の暗号文/平文をあらかじめまとめて処理することにより，実行時間が短縮されることを示した．今後の課題として，一度にまとめて処理する暗号文/平文の数をある一定の数に制限した上で，処理する暗号文/平文の選別を工夫することで，実行時間のさらなる高速化を実現したい．

参考文献

[ElGamal 85] ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transactions on Information Theory*, Vol. IT-31, No. 4, pp. 469–472 (1985)

[Mayers 03] Mayers, R. and Frank, C.: Implementing Your Own Cryptographic Provider Using the Java Cryptography Extension, in *Proceeding of the First Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pp. 51–60 (2003)

[Yokoo 92] Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K.: Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving, in *Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, pp. 614–621 (1992)

[Yokoo 02] Yokoo, M., Suzuki, K., and Hirayama, K.: Secure Distributed Constraint Satisfaction: Reaching Agreement without Revealing Private Information, in *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming (CP-2002)*, pp. 387–401 (2002)