量子鍵配送プロトコルの無条件安全性証明の自動化に向けて

久保田 貴大[1], 角谷 良彦[1], 加藤 豪[2], 河野 泰人[2]

[1]東京大学大学院情報理工学系研究科
[2]NTTコミュニケーション科学基礎研究所

量子鍵配送プロトコル(QKD)

- 量子通信を用いた秘密鍵共有プロトコル
  - 無条件安全性が得られる
    - 攻撃者は量子チャネルに任意の攻撃が可能
    - 攻撃者に漏れる情報量が無視できるほど小さい
      - 攻撃者の計算能力によらない
  - BB84, B92, DPS-QKD, COW-QKD…
- 安全性証明が複雑 [Mayers97]
- 形式手法の有効性を調べたい

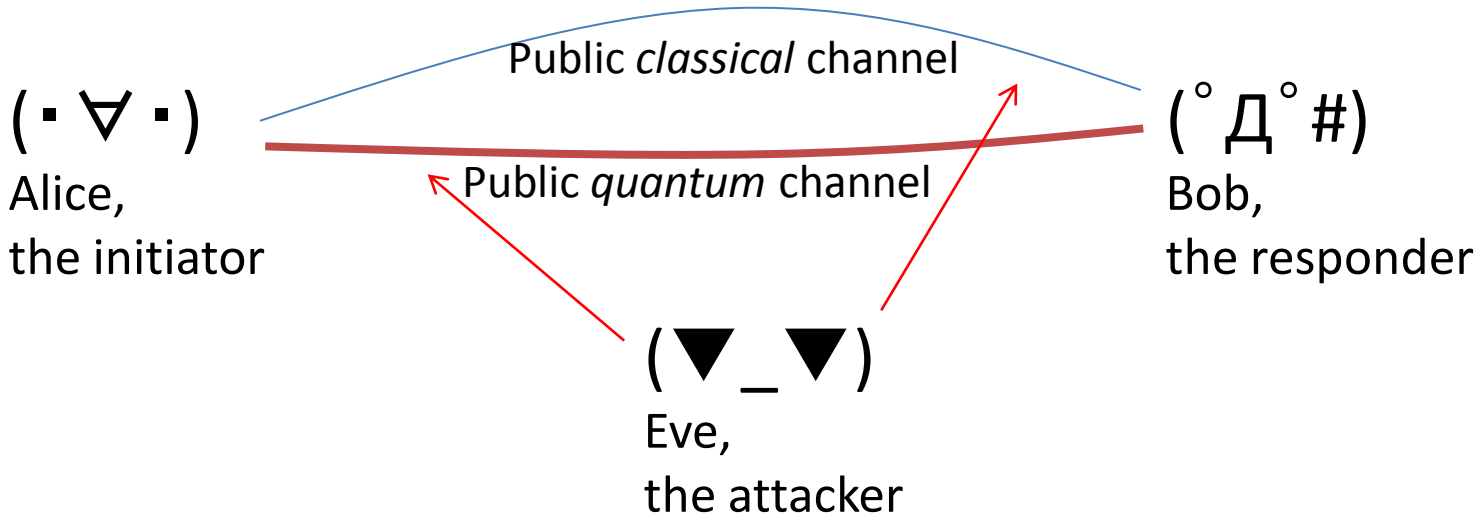
研究の概要

- BB84プロトコルの、Shor-Preskillの証明法[Shor-Preskill00]を形式化した
  - 変形ステップ
    - BB84プロトコルを、Modified Lo-Chauプロトコルに変形するステップ
    - 変形を自動的に行う枠組みを作った
  - 解析ステップ
    - Modified Lo-Chauプロトコルが安全であることを証明するステップ
    - 量子ホーア論理[Kakutani09]で証明を形式化した

発表の流れ

- BB84プロトコルについて
- 変形ステップの自動化について
- 解析ステップの形式化について
- まとめと今後の課題

攻撃者の能力に関する仮定



- 古典チャネルは盗聴可能、改ざん不可
- 量子チャネルには任意の攻撃が可能
- 無限の計算能力を持つ

安全性

- 無条件安全性
  - 任意の攻撃者(イブ)に対して、プロトコルが中断する確率が無視できるほど小さいならば、
    アリスの鍵とイブの推測の相互情報量が無視できるほど小さい。

Preliminaries

- 量子の純粋状態
  - 純粋状態 $|\psi\rangle$ は、C^2の単位ベクトルである

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Z 基底

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

X 基底

観測 (1)

- |0> , |1> が Z 基底で観測された場合
  - 結果は確率1でそれぞれ |0>, |1> である。
- |0> , |1> が X 基底で観測された場合
  - 結果は確率1/2で |+> か |-> のどちらかである。

観測 (2)

- |+> , |-> が X 基底で観測された場合
  - 結果は確率1でそれぞれ |+>, |-> である。
- |+> , |-> が Z 基底で観測された場合
  - 結果は確率1/2で |0> か |1> のどちらかである。

BB84 で用いられる状態と観測

- ４種類の状態を用いる: |0>, |1>, |+>, |->
- ２種類の観測基底を用いる: Z, X
  - |0>, |1> はZ基底で識別可能
  - |+>, |-> はX基底で識別可能
- アリスはボブに送る状態を、
  基底ビットとランダムビットに従って生成する。

基底	0	0	1	1
ランダム	0	1	0	1
状態	|0>	|1>	|+>	|->

BB84 Protocol

Alice

Bob

BB84 Protocol

Alice

Bob

(*4+δ*)*n* ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

BB84 Protocol

Alice

Bob

(*4+δ*)*n* ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

qb := |0>, |1>, |+>, |->, |0>, |0>, |->, |->…

BB84 Protocol

Alice

(*4+δ*)*n* ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

qb := |0>, |1>, |+>, |->, |0>, |0>, |->, |->…

Bob

(*4+δ*)*n* ビットの乱数列を生成する

bb := 01110100 … (観測基底用)
ZXXXZXZZ

BB84 Protocol

Alice

$(4+\delta)n$ ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

qb := |0>, |1>, |+>, |->, |0>, |0>, |->, |->…

Bob

$(4+\delta)n$ ビットの乱数列を生成する

bb := 01110100 … (観測基底用)
ZXXXZXZZ

BB84 Protocol

Alice

$(4+\delta)n$ ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

qb := |0>, |1>, |+>, |->, |0>, |0>, |->, |->…

qb →

Bob

$(4+\delta)n$ ビットの乱数列を生成する

bb := 01110100 … (観測基底用)
ZXXXZXZZ

db := 0?010??? …

BB84 Protocol

Alice

Bob

$(4+\delta)n$ ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

qb := |0>, |1>, |+>, |->, |0>, |0>, |->, |->…

qb

bb := 01110100 … (観測基底用)
ZXXXZXZZ

db := 0?010??? …

ba

bb

$ba_i \neq bb_i$ なる da_i を捨てる

da := 0 010 …

$ba_i \neq bb_i$ なる db_i を捨てる

db := 0 010 …

BB84 Protocol

Alice

Bob

$(4+\delta)n$ ビットの乱数列をふたつ生成する

ba := 00110011 … (基底用)
ZZXXZZXX

da := 01010011 …

qb := |0>, |1>, |+>, |->, |0>, |0>, |->, |->…

qb

bb := 01110100 … (観測基底用)
ZXXXZXZZ

db := 0?010??? …

ba

bb

$ba_i \neq bb_i$ なる da_i を捨てる

da := 0 010 …

$ba_i \neq bb_i$ なる db_i を捨てる

db := 0 010 …

δ が十分大きければ、
高い確率で$2n$ビット以上が残る
(そうでなければ、プロトコルを中断する)

BB84 Protocol

Alice

Bob

da := 001010110… (*2n*)

db := 001010110… (*2n*)

BB84 Protocol

Alice

da := 001010110… (*2n*)

*da*からランダムに半分選び、
チェック用ビット列にする

Bob

db := 001010110… (*2n*)

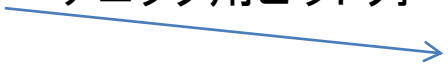
BB84 Protocol

Alice

Bob

da := 001010110…　(*2n*)

*da*からランダムに半分選び、
チェック用ビット列にする



db := 001010110…　(*2n*)

ビットが何個反転しているか数え、
エラーレートを計算する

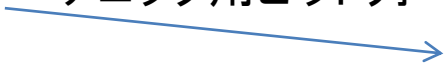
BB84 Protocol

Alice

Bob

da := 001010110… (*2n*)

*da*からランダムに半分選び、
チェック用ビット列にする



db := 001010110… (*2n*)

ビットが何個反転しているか数え、
エラーレートを計算する

エラーレートが閾値より
大きければプロトコルを中断

BB84 Protocol

Alice

Bob

da := 001010110…　(*2n*)

*da*からランダムに半分選び、
チェック用ビット列にする

チェック用ビット列

db := 001010110…　(*2n*)

ビットが何個反転しているか数え、
エラーレートを計算する

エラーレート

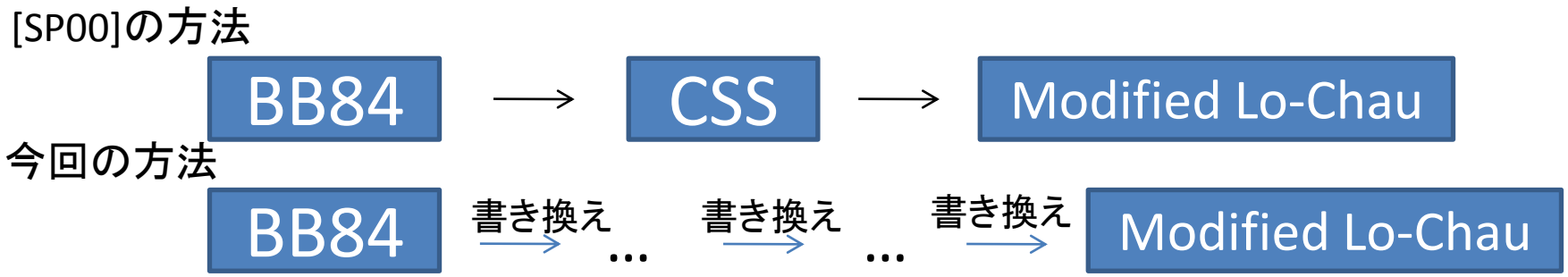
エラーレートが閾値より
大きければプロトコルを中断

n ビットが残る

・エラー訂正
・プライバシー増幅

変形ステップ [Shor-Preskill00]

- BB84プロトコルをModified Lo-Chauプロトコルに変形するステップ
  - Modified Lo-Chauプロトコルは、直接解析しやすい
- 変形ステップを自動化する枠組みを作った
  - BB84をプログラミング言語で形式化し、
  - 変形のための書き換え規則を作り、
  - 自動化に有用な性質を考察した



自動化の枠組み

- 構文論
  - BB84を形式化するのに十分な表現力のプログラミング言語 [Selinger04]

$$
\begin{aligned}
P &::= \texttt{skip}; \mid \texttt{bit}\, b; \mid \texttt{discard}\, b; \mid \texttt{qbit}\, q; \mid \texttt{discard}\, q; \\
&\quad \mid C \mid \texttt{if}\, k\, \texttt{then}\, C \mid P\, P \\
C &::= b := k; \mid q := k; \\
&\quad \mid q, ..., q * = U; \mid b := \texttt{measure}\, q; \\
&\quad \mid f(b, ..., b, q, ..., q | b, ..., b); \\
k &::= 0 \mid 1 \mid b \mid k + k
\end{aligned}
$$

b : program variable for a classical bit
q : program variable for a quantum bit
f : procedure name
U : unitary matrix

  - 攻撃者の存在のもとでのBB84の実行を、プログラムとして形式化した。

BB84

```
      : bit ka[1..'m]; bit kb[1..'m];
      : qbit qb[1..(4+'d)'n]; bit ba[1..(4+'d)'n];
      : bit bb[1..(4+'d)'n]; bit c[1..2'n];
      : bit da[1..(4+'d)'n]; bit db[1..(4+'d)'n];
      : bit e[1..7]; bit ua[1..'n];
      : bit ub[1..'n]; bit x[1..'n];
      : bit sx[1..'n]; bit ke[1..'l];
      : qbit qe[1..'l];
Alice : Rnd(da[1..(4+'d)'n]|);
Alice : for i := 1 to (4+'d)'n do
Alice :  qb[i] := da[i]; end
Alice : Rnd(ba[1..(4+'d)'n]|);
Alice : for i := 1 to (4+'d)'n do
Alice :  if ba[i] then qb[i] *= H;
Alice : end
Eve   : Eve_Attack(ke[],qe[],qb[]|);
Bob   : Rnd(bb[1..(4+'d)'n]|);
Bob   : for i := 1 to (4+'d)'n do
Bob   :  if bb[i] then qb[i] *= H;
Bob   :  db[i] := measure qb[i];
Bob   : end
Bob   : for i := 1 to (4+'d)'n do
Bob   :  if ba[i]==bb[i] then g[i]:= 0
Bob   :                  else g[i]:= 1
Bob   : end
```

```
Alice : Unshuffle(da[1..(4+'d)'n]|g[]);
Bob   : Unshuffle(db[1..(4+'d)'n]|g[]);
Alice : Half_Rnd(c[1..2'n]|);
Alice : Unshuffle(da[1..2'n]|c[]);
Bob   : Unshuffle(db[1..2'n]|c[]);
Alice : e[] := 0;
Alice : for i := 101 to 2'n do
Alice :  unless da[i]==db[i]
Alice :   then Add_One(e[]|);
Alice : end
Alice : Abort(|e[]);
Alice : Rnd_in_C1(ua[1..'n]|);
Alice : for i := 1 to 'n do
Alice :  x[i] := da[i] + ua[i]; end
Bob   : for i := 1 to 'n do
Bob   :  ub[i] := db[i] + x[i]; end
Bob   : Syndrome(sx[]|ub[1..'n]);
Bob   : Correct(ub[1..'n]|sx[]);
Alice : Key(ka[1..'m]|ua[1..'n]);
Bob   : Key(kb[1..'m]|ub[1..'n]);
Eve   : Eve_Guess(ke[],qe[]|
Eve   :           ba[],bb[],c[],e[],x[]);
```

意味論

- プログラム変数の量子状態は密度行列で表現される
- プログラムは、実行前の密度行列を実行後の密度行列に写すオペレータとして解釈される[Selinger04]

書き換え規則

- 健全性
  - 書き換え後のプログラムが安全ならば
    書き換え前のプログラムが安全である
- 書き換え規則の例

意味が同じ

```
b := 0;
if b then P;
-------------
b := 0;
```

```
q[0..'n] *= U1;
q[0..'n] *= U2;
---------------
q[0..'n] *= U1*U2;
```

攻撃者のプロジージャの扱い

```
P;
Eve_Attack(ke[],qe[],qb[]);
Q;
-------------------------------#
Eve_Attack(ke[],qe[],qb[]);
```

上側が、ke[], qe[], qb[]以外の変数を含まない

プロシージャの性質

```
Shuffle(q[1..100]|c[]);
Unshuffle(q[1..100]|c[]);
-------------------------
skip;
```

```
Unshuffle(q[1..100]|c[]);
Shuffle(q[1..100]|c[]);
-----------------------
skip;
```

書き換え系の性質

適当な制約のもとで、
以下が成立することを示した

- 強正規化性(SN)
  - 任意のプログラムの書き換えが停止する
- 合流性(CR)
  - 正規形があれば、書き換えの戦略によらず一意

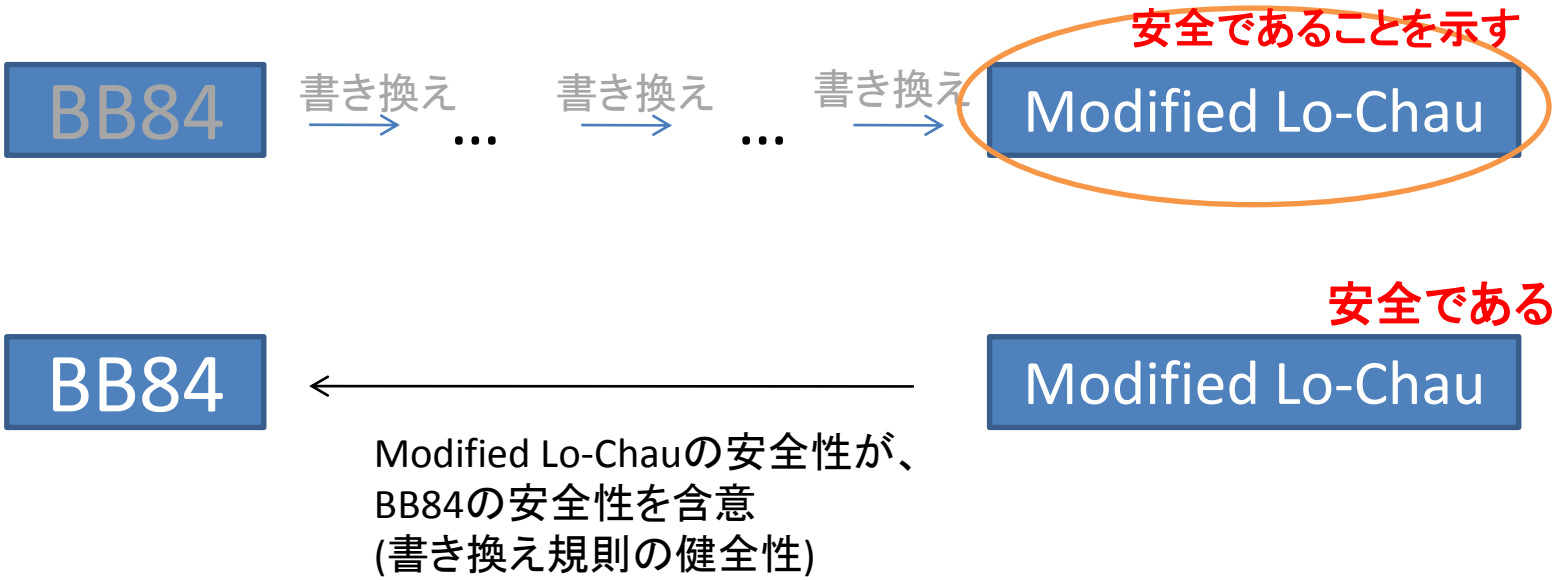
⇒　BB84は必ず Modified Lo-Chau プロトコルに書き換えられて、書き換えが停止する

解析ステップ

- Modified Lo-Chauプロトコルが安全であることを示すステップ
- 証明を量子ホーア論理を用いて形式化した

解析ステップ

- Modified Lo-Chauプロトコルが安全であることを示すステップ
- 証明を量子ホーア論理を用いて形式化した



量子ホーア論理

- 量子プログラムの性質を、ホーアトリプルで表明する枠組み[Kakutani09]
  $\{\Phi\}\ P\ \{\Psi\}$　Φが成り立つときにPを実行するとΨが成り立つ
- プログラムの構文と意味は変形ステップで用いたものと同じ

証明したいこと

$\{\ \}$

```
bit ka[1..40]; bit kb[1..40];
qbit qa[1..200]; qbit qb[1..200];
bit b[1..200]; bit c[1..200];
bit da[101..200]; bit db[1..200];
bit e[1..7]; bit ke[1..65536];
qbit qe[1..65536];
for i := 1 to 200 do
 EPR(qa[i], qb[i]|);
end
Rnd(b[1..200]|);
for i := 1 to 200 do
 if b[i] then qb[i] *= H;
end
```

...

$$\{\mathrm{pr}(\mathrm{ka} = \mathrm{kb}) \geq 2^{-(\gamma-\beta)n} \rightarrow \mathrm{I}(\mathrm{ka}; \mathrm{ke}|\mathrm{e} \leq cn) \leq 2^{-\alpha n}\}$$

証明したいこと

```
{ }
 bit ka[1..40]; bit kb[1..40];
 qbit qa[1..200]; qbit qb[1..200];
 bit b[1..200]; bit c[1..200];
 bit da[101..200]; bit db[1..200];
 bit e[1..7]; bit ke[1..65536];
 qbit qe[1..65536];
 for i := 1 to 200 do
  EPR(qa[i], qb[i]|);
 end
 Rnd(b[1..200]|);
 for i := 1 to 200 do
  if b[i] then qb[i] *= H;
 end
```

Modified Lo-Chau プロトコル

...

プロトコルの実行後に
成り立つべき性質

$$\{\mathrm{pr}(\mathrm{ka} = \mathrm{kb}) \geq 2^{-(\gamma-\beta)n} \rightarrow \mathrm{I}(\mathrm{ka}; \mathrm{ke}|\mathrm{e} \leq cn) \leq 2^{-\alpha n}\}$$

まとめ

- Shor-PreskillのBB84安全性証明を形式化した
  - 変形ステップ
    - BB84をプログラムとして形式化し、
    - 意味論に対して健全な書き換え規則を作り、
    - 書き換え系の性質
  - 解析ステップ
    - 量子ホーア論理で形式化した

今後の課題

- より多くの量子鍵配送プロトコル(QKD)が扱えるよう、枠組みを拡張する
  - 現在、BB84とB92の変形が可能
  - 六状態プロトコル[GB98]、SARGプロトコル[SARG04]など、多数のQKDが扱えることを示す
- 別の検証要件
- 並列実行が形式化しやすい枠組みを検討する
  - マルチセッションを想定した安全性
  - マルチパーティ・プロトコル