# 知識の証明と暗号技術

有田正剛 †

**概 要**

暗号技術/理論の面白さとむずかしさは，システムには攻撃者が存在するという前提から生じる．攻撃者は約束事には従わず，システムを本来とは異なる用途に利用しようとする．それを防ぐには,「知識の証明」が基本手段となる．本解説論文では，ここでいう「知識」とは何か，それを「証明する」とはどういうことかを，離散対数問題を例として説明し，具体的な知識の証明プロトコルとしてシュノアプロトコルを紹介する．さらに，知識の証明によって安全性が強化される事例の代表として公開鍵暗号を取り上げる．公開鍵暗号の安全性がどのように考えられ定義されるかを説明し，離散対数問題の困難性にもとづく，代表的な公開鍵暗号であるエルガマル暗号が，そのままでは安全とは言い難いこと，しかし，それをシュノアプロトコル (やその改良) を用いてより安全な公開鍵暗号へと変換できることをみる．

## 1 はじめに

### 1.1 スフィンクスの問いかけ

あなたはある密命をおびてエジプトのとあるピラミッドを訪ねた（と思ってください）．ピラミッドに入ろうとすると，スフィンクスが問いかける,「二乗して 323 でわると 206 余る，その数を知っているか？」．

それは知っている，事前に教わっていた『知識の数』23 だ．(確かに 23 を二乗すると 529 で，529 を 323 でわると 206 余る．) そう口に出そうとした瞬間,「ただし，その『知識の数』そのものを決して口にしてはならない.」とスフィンクスが遮った．近辺に怪しい気配を察知し知識の数を盗み取られることを警戒しているのか，スフィンクス自身は知識の数を知ることを許されていないのか．

とにかく，知識の数 23 を一切口にすることなく，それを知っていることだけを示せとスフィンクスはいう．

### 1.2 知識の証明と暗号技術

これまでもこれからも，システムには攻撃者が存在する．攻撃者は利己的な目的でシステムを本来とは異なる用途に利用しようとする．それを放置すると，正直な利用者が損害を被ることになる．

†情報セキュリティ研究科 教授

攻撃者の利己的な振る舞いを防ぐために，暗号技術の定石では，利用者に対して「知識の証明」による正当性証明を要求する．正当な利用者であること，より一般には，約束事を守った正当なシステム利用であることを，利用者に証明してもらう．証明が不適切なら相手をしない．

それらの正当性証明は，パスワードや秘密の乱数などの何らかの秘密情報をもっていることを示すことによって行われる．そのため，正当性証明を通じて利用者の重要な秘密情報がシステム側にもれてしまう危険がある．システムは，実は攻撃者がそれとみせかけた偽物であるかもしれない．そのようなことを防ぐために，正当性証明はゼロ知識である必要がある．すなわち，たとえ証明を検証する立場のシステム側がどのような企みをもってふるまおうとも，利用者の秘密情報がシステム側に漏れることはなく，システムが得る情報はただ利用者が正当であるというそのことのみでなければならない．

知識のゼロ知識証明（という気がおかしいのではないかと思える言葉が表すもの）とは，知識をもっていることをその知識自体は一切教えないで証明するためのプロトコルである．知識のゼロ知識証明を用いることで，様々なプロトコルやシステムをより強力で能動的な敵にも耐えうるよう，強化することができる．

### 1.3 使者の応答

途方に暮れたあなたに，スフィンクスが第一問を発した.「なにかある数を選び，二乗して 323 でわった余りをいいなさい.」

では，61 を選ぼう．61 の二乗は 3721．3721 を 323 でわると 168 余る.「168.」

スフィンクスは第二問を続ける.「その数に知識の数をかけ，323 でわった余りをいいなさい.」

知識の数は 23．先に 61 を選んだ．61 に 23 をかけると 1403．それを 323 でわると 111.「111.」

スフィンクスは判断をくだす.「あなたの第一の答え 168 に 206 を掛け 323 でわると 47 余る．あなたの第二の答え 111 を二乗して 323 でわると 47 余る．この 2 つの余りは一致した．よって，あなたを正しい使者と承認する.」

何をもってスフィンクスは使者を正しいと認めたのか？このようにすることで，知識の数「23」はだれにもスフィンクスにさえも知られないとどうして言えるのだろうか？

## 2 知識と整数論

### 2.1 「知識」とは？

スフィンクスと使者は，実は，知識のゼロ知識証明プロトコルを実行している．その詳細を見る前に，まず，ここでいう「知識」が何を意味しているか確認したい．

「知識」とは問題に対する答えである．ただし，その答えは，

- 簡単には求められないこと．
- しかし，いったん答えが与えられたら，それが正しいことは簡単に確認できること．

の 2 条件を満たすようなものであるとする．

簡単な問題であれば，問題自体が情報として答えを含んでいるようなものなので，その答えをわざわざ知識とは呼ばないのは自然である．一方，あまりに問題が難しく，答えを教わってもその正しささえ自力で確認できないようでは，知識の証明には使い難い．

整数論は，このような，問題とその答えとしての知識の宝庫である．たとえば，次にみる平方剰余問題とその答えがそれである．

## 2.2 モジュロ演算

まず，モジュロ演算について説明する．2 整数 $a, b$ と整数 $N$ について，その差 $a - b$ が $N$ の倍数となっていることを

$$a \equiv b \pmod{N}$$

という等式で表す．合同式という．mod の中の $N$ は法と呼ぶ．たとえば，32 と 2 の差 30 は 15 のちょうど 2 倍なので，

$$32 \equiv 2 \pmod{15} \tag{1}$$

となる．32 と 2 は 15 を法として等しい．21 と 6 の差 15 ももちろん 15 の倍数なので，

$$21 \equiv 6 \pmod{15} \tag{2}$$

である．21 と 6 は 15 を法として等しい．

モジュロ演算の記号を使うと，スフィンクスの問いかけ「二乗して 323 でわると 206 余る，その数を知っているか?」は，コンパクトに「方程式 $X^2 \equiv 206 \pmod{323}$ の解を知っているか？」とかける．

モジュロ演算の性質として，

$$a_1 \equiv b_1 \pmod{N},\ a_2 \equiv b_2 \pmod{N}$$

ならば

$$\begin{aligned} a_1 + a_2 &\equiv b_1 + b_2 \pmod{N} \\ a_1 \cdot a_2 &\equiv b_1 \cdot b_2 \pmod{N} \end{aligned}$$

が成り立つ．すなわち，モジュロの意味で等しいもの同士を足したり，かけたりしても等しいまま．たとえば，式 (1) と式 (2) より，

$$\begin{aligned} 32 + 21 &\equiv 2 + 6 \pmod{15} \\ 32 \cdot 21 &\equiv 2 \cdot 6 \pmod{15} \end{aligned}$$

となる．

ただし，割り算については，$c$ と $N$ が互いに素 (それらの間の最大公約数が 1) なときには，

$$ca \equiv cb \pmod{N} \ \Rightarrow\ a \equiv b \pmod{N}$$

となる．法 $N$ を保ったままで両辺を $c$ でわりたいときは，$c$ と $N$ が互いに素であることを確認する必要がある．実際，式 (1) の両辺を 2 でわって

$$16 \equiv 1 \pmod{15}$$

となるが，式 (2) の両辺を 3 でわっても $7 \equiv 2 \pmod{15}$ とはならず，

$$7 \equiv 2 \pmod{5}$$

となる．2 は 15 と互いに素だが，3 は 15 とは互いに素でないから．

## 2.3 平方剰余問題

実数の世界では，2 次方程式 $X^2 = 206$ はたとえば以下のようにして解くことができる．
まず，方程式の右辺 206 は $100 = 10^2$ と $400 = 20^2$ の間にあるので，

$$10 \leq X \leq 20.$$

10 と 20 の中央は $\frac{10+20}{2} = 15$ で，その二乗 $15^2 = 225$ は方程式の右辺 206 より大きいので，解 $X$ は 15 よりも小さい．よって，

$$10 \leq X \leq 15.$$

10 と 15 の中央は $\frac{10+15}{2} = 12.5$ で，その二乗 $12.5^2 = 156.25$ は方程式の右辺 206 より小さいので，解 $X$ は 12.5 よりも大きい．よって，

$$12.5 \leq X \leq 15.$$

以下，同様に繰り返すと繰り返しのたびに $X$ の存在する範囲を半分に狭めることができ，解 $X = 14.35270009\cdots$ に高速に近づいていくことができる．

$N = 323$ を法とする世界ではどうか．方程式 $X^2 \equiv 206 \pmod{323}$ も同じように解くことができるだろうか．先の節の話では，合同式も通常の式と同じように扱えていた．
上でみた平方根の計算方法を改めて観察すると，数の大小関係や数の間の距離（中央）を積極的に用いている．真の解が（解候補の）区間の左半分それとも右半分にあるかを，区間の中央値と真の解との大小関係から判断し，真の解に近づいていった．
ところが，$N = 323$ を法とする世界では，数の大小関係や距離の意味がよくわからなくなってしまう．たとえば，$x = 0$ からはじめて $x$ を 1 ずつ増やしていくと，最初のうちは 1 ずつ増えるが，$N = 323$ に達する瞬間に 0 に戻る．増加の反復が減少になった．実数の場合の計算方法をあてはめるには，モジュロの世界での，数の大小関係や距離の意味を考え直す必要がありそうである．

合成数 $N$ と整数 $a$ について，方程式 $X^2 \equiv a \pmod{N}$ の解 $X$ を求める問題，すなわち法 $N$ のもとでの整数 $a$ の平方根 (平方剰余) を求める問題を平方剰余問題と呼ぶ．スフィンクスは使者に対し平方剰余問題を投げかけていた．

実は，法 $N$ が素数のときには平方剰余問題を解くのは容易である．$N$ が 4 を法として 1 に等しい素数のときには簡単な公式まである．しかし，法 $N$ が合成数のときに平方剰余問題を解くのは難しく，以下の平方剰余仮定が信じられている．(もちろん，$N$ を法とするとき，整数は高々$N$ 個の有限個しかないので，すべてをしらみつぶしにあたれば平方剰余を求めることはできる．しかし，暗号で用いているような大きさの $N$ の場合，その解を生きているうちに目にすることはできない．)

**平方剰余仮定**

$N$ が大きな合成数のとき，平方剰余問題の解を効率的に求めるアルゴリズムは存在しない．

以下，本解説では，平方剰余仮定は正しいとする．(平方剰余仮定が成り立つような大きな合成数 $N$ をとる．)

法 $N$ が素数のときには平方剰余問題を解くのは容易なのだから，平方剰余仮定が成り立つためには $N$ の素因数分解が困難（素因数分解仮定）でなければならない．実際，平方剰余仮定は素因数分解仮定と同じ難しさ，一方が解ければ他方も同程度の時間で解けることが知られている．

## 3 知識のゼロ知識証明

知識のゼロ知識証明とは，難問の種類（言語）ごとに定まる，証明者 $P$ と検証者 $V$ からなるプロトコルである．証明者 $P$ は検証者 $V$ に，自分が難問 $x$ の答えである知識 $w$ を知っていることを，知識 $w$ そのものは明かすことなく証明したい．より正確には，プロトコル $(P,V)$ が（言語 $L$ の）知識のゼロ知識証明であるとは，($L$ に属する) 任意の問題 $x$ とその知識 $w$ について，

**(完全性)** 証明者 $P$ が知識 $w$ をもっているとき，検証者 $V$ は $P$ を必ず承認する．言い換えれば，$P$ は $V$ を必ず説得できる．

**(健全性)** 悪意ある証明者 $P^*$ が知識 $w$ をもっていないにもかかわらず検証者 $V$ を説得しようとしても，$V$ は（殆んどの場合）$P^*$ を拒否する．ここで，悪意ある証明者 $P^*$ はプロトコルに従う保証はなく，好き勝手な方法で $V$ へのメッセージを計算しうることに注意．

**(ゼロ知識)** 悪意ある検証者 $V^*$ は，証明者 $P$ とのやりとりにおいてどのように振る舞っても，問題 $x$ に関する自明な情報 (だれもが効率的に計算できる情報) しか入手できない．ここでも，上と同じように，悪意ある検証者 $V^*$ はプロトコルに従う保証はなく，好き勝手な方法で $P$ へのメッセージを計算しうることに注意．

という 3 つの性質が同時に成り立つことをいう [4].

これら 3 つの性質は互いに相反する傾向にある．健全性を確実に達成するには知識 $w$ そのものを検証者 $V$ に伝えるのが手っ取り早いが，これではゼロ知識ならぬフル知識となる．ゼロ知識を達成するには証明者 $P$ のメッセージになるべく知識 $w$ の情報を入れないようにしたいが，それでは健全性が危うくなる．相反する 3 性質の両立を図ることがポイントである．

### 3.1 平方剰余の知識のゼロ知識証明，フィアット・シャミアプロトコル

平方剰余問題 $X^2 \equiv a \pmod N$ の解である，平方剰余の知識 $X = w$ は，図 1 のフィアット・シャミアプロトコル [3] によってゼロ知識で証明することができる．

証明者 $P$ は入力として平方剰余問題 $X^2 \equiv a \pmod N$ を表す整数ペア $(a, N)$ とその解 $X = w$ を得る．検証者 $V$ は入力として整数ペア $(a, N)$ のみを得る．証明者 $P$ と検証者 $V$ は以下の基本プロトコルを十分な回数繰り返す．それら繰り返しのすべてが承認であれば，検証者 $V$ は承認を出力する．一度でも拒否があれば検証者 $V$ は拒否を出力する．

[**基本プロトコル**]

1. (証明者の第 1 メッセージ，“**コミットメント**”) 証明者 $P$ は 0 以上 $(N-1)$ 以下の範囲の乱数 $r$ を選び，$r$ の法 $N$ での平方 $c = r^2 \bmod N$ を計算する．証明者 $P$ は $c$ を検証者 $V$ に送る．

2. (検証者のメッセージ，“**チャレンジ**”) 検証者 $V$ は $e$ として 0 または 1 をランダムに選択し，証明者 $P$ に送る．

3. (証明者の第 2 メッセージ，“**レスポンス**”) 証明者 $P$ は，受け取った $e$ が 0 ならばコミットメント $c$ に含まれる乱数 $r$ をそのまま $y$ とし $(y = r)$，$e$ が 1 ならば乱数 $r$ に秘密の解 $w$ を法 $N$ で掛けて $y$ とする $(y = rw \bmod N)$．証明者 $P$ は $y$ を検証者 $V$ に送る．

4. (検証者の判断) 検証者 $V$ は証明者のコミットメント $c$，自身で選択したチャレンジ $e$，証明者のレスポンス $y$ の間で以下の等式が成り立つかどうか調べる．

   $$y^2 \equiv ca^e \pmod N.$$

   成り立つならば（基本プロトコルの出力として）承認し，成り立たないなら拒否する．

図 1: 平方剰余の知識のゼロ知識証明，フィアット・シャミアプロトコル

図 1 のフィアット・シャミアプロトコルをみると，$e = 0$ のとき $a^0 \equiv 1$ なので検証式（検証者 $V$ が最後に調べる等式) は $y^2 \equiv c \pmod N$．$y = r$ でコミットメント $c$ は $r$ の平方としてつくられたのでこれはもちろん成立．$e = 1$ のときには検証式は $y^2 \equiv ca \pmod N$ となるが，今度は $y$ には $w$ がかかり $y \equiv rw$ となっていて，$w$ の平方が $a$ なのでやはりこれは成立する．このように $P$ が $a$ の平方剰余である知識 $w$ をもっているとき，$V$ は $P$ を必ず承認する (完全性)．

フィアット・シャミアプロトコルの健全性を調べる．(悪意があるかも知れない) ある証明者 $P^*$ が検証者 $V$ に承認されるものとする．いまフィアット・シャミアプロトコルが，証明者 $P^*$ が第 1 メッセージ $c$ を送ったところまで進んだとする．この $P^*$ が $V$ に承認さ

れるには，次に検証者 $V$ が送ってくるチャレンジ $e$ について，それが $e = 0$ のときにも $e = 1$ のときにも検証式をみたすようなレスポンス $y$ を返せなければならない．(一度でも答えそこなうと $V$ に拒否されてしまうので，$e = 0$ か $e = 1$ かどちらかにヤマを張るのは無理．) これら 2 つのレスポンス $y$ を $y_0, y_1$ とする．これら $y_0, y_1$ を $P^*$ は知っており，

$$\begin{aligned} y_0^2 &\equiv c \pmod N \\ y_1^2 &\equiv ca \pmod N \end{aligned}$$

が成り立つ (2 式が $c$ を共有していることに注意)． 辺々をわると

$$(y_1/y_0)^2 \equiv a \pmod N$$

となる[1]．すなわち， $w = y_1/y_0$ は方程式 $X^2 \equiv a \pmod N$ の解となり，$y_0, y_1$ を知っている $P^*$ は $y_1/y_0$ も知っているので，結局 $P^*$ は平方剰余の知識 $w$ をもっていた，つまり正しい証明者 $P$ だということになる．

ゼロ知識性について検証する．図 1 のフィアット・シャミアプロトコルにおいて証明者 $P$ は知識 $w$ そのものは検証者 $V$ に送らず，乱数 $r$ 倍してからレスポンス $y$ として送っている．この乱数 $r$ の平方をコミットメント $c$ として送っているが，平方剰余仮定のもとでは $c$ からその平方剰余である $r$ を求めることはできず，悪意ある検証者 $V^*$ であっても $y$ から $r$ 経由で $w$ を求めることはできない．このように考えると証明者 $P$ の知識 $w$ が悪意ある検証者 $V^*$ に知られることはなさそうにみえる．しかし，この議論で本当にフィアット・シャミアプロトコルがゼロ知識であるといえたかというとそうとはいえない．$y$ から $r$ 経由で $w$ を求める方法以外にも，$w$ を求める方法はあるだろう．

フィアット・シャミアプロトコルのゼロ知識性は以下のようにして示すができる．どのような悪意ある検証者 $V^*$ も，証明者 $P$ とのやりとりで入手する情報は，問題 $x$ に関する自明な情報 (だれもが効率的に計算できる情報) でしかないことを示したい．次のようなアルゴリズム S(シミュレータと呼ばれる) を考える．

- シミュレータ S:
  1. (入力) 入力として平方剰余問題 $X^2 \equiv a \pmod N$ を表す整数ペア $(a, N)$ を受け取る．
  2. (チャレンジの推測) $e'$ として 0 または 1 をランダムに選択する．
  3. (レスポンスとコミットメントの生成) まずレスポンスとして乱数 $y$ を選び，上で選んだ $e'$ とつじつまがあうようにコミットメント $c$ を $c = y^2/a^{e'} \bmod N$ として生成する．
  4. (推測したチャレンジを確認) S 内部で，$(a, N)$ を入力として検証者 $V^*$ を起動し，$V^*$ にコミットメント $c$ を送り，(本当の) チャレンジ $e$ を受け取る．$e$ が $e'$ と異なるならば最初からやり直す．

[1] こう変形するには，$y_0$(や $c$) が $N$ と互いに素である必要がある．その確認をさぼっているようだが，$y_0$(や $c$) が $N$ と互いに素でないならば，それらの間の最大公約数をユークリッドの互除法で計算すると，$N$ の素因数分解が簡単にできてしまい，前提としている平方剰余仮定そのものが成り立たなくなる．そのため，$y_0$(や $c$) と $N$ は互いに素であるとしてよい．

5. (シミュレートの完了) $V^*$ にレスポンス $r$ を送る．

シミュレータ S は検証者 $V^*$ のチャレンジ $e$ を $e'$ とランダムに推測し，それにあわせて妥当なレスポンス $y$ とコミットメント $c$ を生成している．($c = y^2/a^{e'} \bmod N$ ならば $y^2 \equiv ca^{e'} \pmod N$ となり検証式が成り立つことに注意．) よって，推測 $e'$ が正しいときには，S 内部の $V^*$ が得るコミットメント $c$ とレスポンス $r$ は，本当の証明者 $P$ が知識 $w$ を用いて生成するものと全く同じ (分布) となる．(生成される順序が異なるだけ．) 推測 $e'$ が正しくないときには，もう一度最初からやり直せば，$e$ の値は 0,1 の 2 通りしかないので，平均 2 回の繰り返しでチャレンジ $e$ の正しい推測が得られる．(コミットメント $c$ は乱数 $y$ や推測値 $e'$ に依存するので，最初からやり直すと，$V^*$ が受け取るコミットメント $c$ が変わり，したがって $V^*$ が出力するチャレンジ $e$ も変わってしまうことに注意．) このように，シミュレータ S は，知識 $w$ を用いずに，任意の検証者 $V^*$ が真の証明者とやり取りして入手する情報を効率的に再現できた．つまり，任意の検証者 $V^*$ が真の証明者とのやり取りで入手する情報 $c$ と $y$ は，(驚くことに) 知識 $w$ を用いなくても生成できるものである．よって，それには，知識 $w$ に関する有益な情報はまったく何も含まれていない．だれもが，整数ペア $(a, N)$ を入力としてシミュレータ S を起動すれば得られる，自明な情報でしかない．

## 3.2 離散対数の知識のゼロ知識証明，シュノアプロトコル

平方剰余の知識以外にも様々な知識についてそのゼロ知識証明が知られている．それらのうち，ここでは，離散対数の知識のゼロ知識証明であるシュノアプロトコルをみる．

### 3.2.1 離散対数問題

素数 $p$ と 2 つの整数 $g, h$ について，方程式 $g^x \equiv h \pmod p$ の解 $x$ を求める問題，すなわち，(素数である) 法 $p$ のもとでの整数 $h$ の整数 $g$ を底とする対数 (離散対数) を求める問題を離散対数問題という．

例として，素数 $p = 43$ について，方程式

$$4^x \equiv 35 \pmod{43}$$

の解 $x$ を求めよう．$x = 0$ からはじめて $4^x \bmod 43$ を順番に計算していくと，

$$4^0 \bmod 43 = 1,\ 4^1 \bmod 43 = 4,\ 4^2 \bmod 43 = 16,$$
$$4^3 \bmod 43 = 21,\ 4^4 \bmod 43 = 41,\ 4^5 \bmod 43 = 35,\ \cdots$$

よって，$x = 5$ が得られた．ただし，このように計算すると解 $x$ に辿りつくまでに最悪で $p$ 回程度のモジュロ演算が必要となってしまう．素数 $p$ が大きくなるととても計算しきれない．現在知られている最速のアルゴリズムを用いれば $2^{\mathrm{len}(p)^{1/3}}$ 回程度のモジュロ演算で離散対数を計算できるが（$\mathrm{len}(p)$ は $p$ のビット長），それでも素数 $p$ を十分大きくとるととても計算しきれない．

**離散対数仮定**

$p$ が大きな素数のとき，離散対数問題の解 $x$ を効率的に求めるアルゴリズムは存在しない．

以下，本解説では，離散対数仮定は正しいとする．(離散対数仮定が成り立つような大きな素数 $p$ をとる．)

### 3.2.2 シュノアプロトコル

離散対数問題 $g^x \equiv h \pmod{p}$ の解である，離散対数の知識 $x = w$ は，図 2 のシュノアプロトコル [6] によってゼロ知識で証明することができる．

証明者 $P$ は入力として離散対数問題 $g^x \equiv h \pmod{p}$ を表す整数組 $(p, g, h, q)$ とその解 $x = w$ を得る．(ただし，ここで $q$ は $g^q \equiv 1 \pmod{p}$ となる最小の正整数 $q$ とし，そのような整数 $q$ が素数となるような $g$ が選ばれているとする．このような $g$ を実際にどのように選ぶかはここでは気にしない．) 検証者 $V$ は入力として整数組 $(p, g, h, q)$ のみを得る．

1. (証明者の第 1 メッセージ，“**コミットメント**”) 証明者 $P$ は 0 以上 $(q-1)$ 以下の範囲の乱数 $r$ を選び，$g$ の法 $p$ での $r$ 乗であるコミットメント $c = g^r \bmod p$ を計算する．証明者 $P$ は $c$ を検証者 $V$ に送る．

2. (検証者のメッセージ，“**チャレンジ**”) 検証者 $V$ はチャレンジ $e$ として 0 以上 $(q-1)$ 以下の範囲の乱数を選択し，証明者 $P$ に送る．

3. (証明者の第 2 メッセージ，“**レスポンス**”) 証明者 $P$ は，受け取った $e$ と先に選んだ乱数 $r$ そして知識 $w$ を用いて，レスポンス $y = (r + ew) \mod q$ を計算する．証明者 $P$ は $y$ を検証者 $V$ に送る．

4. (検証者の判断) 検証者 $V$ は証明者のコミットメント $c$，自身で選択したチャレンジ $e$，証明者のレスポンス $y$ の間で以下の等式が成り立つかどうか調べる．

$$g^y \equiv ch^e \pmod{p}.$$

成り立つならば承認し，成り立たないなら拒否する．

図 2: 離散対数の知識のゼロ知識証明，シュノアプロトコル

図 2 のシュノアプロトコルはフィアット・シャミアプロトコル (図 1) と似た構造をしているがそれとは重要な違いがある．チャレンジ $e$ のビット長をみると，フィアット・シャミアプロトコルでは 1 ビットだったが，シュノアプロトコルでは素数 $q$ と同じサイズでずいぶん長い．そのおかげで，フィアット・シャミアプロトコルではコミット・チャレンジ・レスポンスからなる基本プロトコルを何度も繰り返す必要があったが，シュノアプロトコルではその必要はなく，1 回実行するだけとなっている．ただし，いいことばかりではな

く，後に見るようにゼロ知識性に制限がついてしまう．以下，シュノアプロトコルの完全性・健全性・ゼロ知識性を順に見ていく．

シュノアプロトコルにおいて，証明者 $P$ が離散対数の知識 $w$ を用いて正しく $y$ を計算したならば，

$$g^y = g^{r+ew} = g^r(g^w)^e \equiv ch^e \pmod{p}$$

が成り立ち，検証者 $V$ は必ず承認する (完全性)．

シュノアプロトコルの健全性も，フィアット・シャミアプロトコルの場合と同様にして確かめることができる．ある証明者 $P^*$ が検証者 $V$ に承認されるとすると，ある同じコミットメント $c$ に対して，(少なくとも)2 つの異なるチャレンジ $e_0, e_1$ について共に検証式をみたすようなレスポンス $y_0, y_1$ を返せなければならない．このとき，

$$g^{y_0} \equiv ch^{e_0},\ g^{y_1} \equiv ch^{e_1} \qquad \pmod{p}$$

が成り立つので，辺々わって，

$$g^{y_0-y_1} \equiv h^{e_0-e_1} \pmod{p}.$$

よって，$P^*$ は離散対数の知識 $w = (y_0 - y_1)/(e_0 - e_1)$ をもつ正しい証明者である．

シュノアプロトコルのゼロ知識性には制限がつく．悪意ある検証者 $V^*$ がまったく任意に振る舞うときのゼロ知識性は示されていない．シュノアプロトコルのゼロ知識性は，チャレンジ $e$ を正しくランダムに生成する (が内部で証明者 $P$ の知識 $w$ を不正に導き出そうとする) ような検証者 $V^*$ についてのみ以下のように示されている．

上の意味で制限された，悪意ある検証者 $V^*$ が証明者 $P$ とのやりとりで得る情報は，

$$\{(c, e, y) \ : \ r \stackrel{\$}{\leftarrow} \{0, \ldots, q-1\}, e \stackrel{\$}{\leftarrow} \{0, \ldots, q-1\}, c = g^r \bmod p, y = (r + ew) \bmod q\}$$

という分布をもつ．これは，知識 $w$ を用いなくても，

$$\{(c, e, y) \ : \ e \stackrel{\$}{\leftarrow} \{0, \ldots, q-1\}, y \stackrel{\$}{\leftarrow} \{0, \ldots, q-1\}, c = g^y h^{-e} \bmod p\}$$

としても生成できる．よって，$V^*$ が $P$ から入手しうる情報は $w$ がなくても生成できる自明な情報のみである．

## 3.3 プロトコル実行の知識とそのゼロ知識証明

一般に，各パーティによるプロトコルの実行は，次にような数学的なモデルで表すことができる．

$F$ をプロトコルによって定まる関数とする．$F$ は効率的に計算することができる．各パーティは内部に状態 $s$ と乱数 $r$ をもち，次のループを繰り返す．

- LOOP:

  1. メッセージ $m$ を受け取る．

2. 状態 $s$ と乱数 $r$ を用いて，次にとるべき状態 $s'$ と送信すべきメッセージ $m'$ を関数 $F$ を用いて計算する:

$$(s', m') \leftarrow F(s, r, m).$$

3. メッセージ $m'$ を送信する．
4. 状態 $s$ を状態 $s'$ に更新し，LOOP の先頭に戻る．

このようなモデルのもとで，つぎの問題を考える．

**プロトコルに従っていることを表す問題**

与えられた $(m, m')$ について，$(s', m') = F(s, r, m)$ となる $s, s', r$ を求めよ．

プロトコルを $F$ に従って正しく実行しているパーティならば，上の問題に正しく答えられる．所定の手続き $F$ を用いていない，悪意あるパーティが正しく答えることは難しい．答えが正しいか否かを検証することは，$F$ は効率的に計算できるので，容易である．よって，この「プロトコルに従っていることを表す問題」の答えは「知識」であり，知識のゼロ知識証明をもつ．この知識のゼロ知識証明を用いれば，プロトコルを実行する各パーティのプライバシーを守りつつ (ゼロ知識性)，プロトコルに従うことを強制することができる (健全性)．

ただし，これはあくまで理論上の話であって，実際の $F$ は非常に複雑な関数となるため，それに対応する知識のゼロ知識証明も，フィアット・シャミアプロトコルやシュノアプロトコルのように効率的なものは，早々期待できない．

# 4 知識のゼロ知識証明の公開鍵暗号への応用

## 4.1 公開鍵暗号の概念

公開鍵暗号とは暗号化の鍵と復号用の鍵を別々にした暗号方式である．暗号化の鍵は公開鍵として公開し，復号用の鍵は秘密鍵として秘密に保管する．

公開鍵暗号を用いて，アリスがボブにメッセージ $M$ を暗号化して送るには次にようにする．

1. アリスはまず公開されているボブの公開鍵 $pk_{Bob}$ を入手する．
2. アリスはボブの公開鍵 $pk_{Bob}$ とメッセージ $M$ を暗号化アルゴリズムに入力し，暗号文 $C$ を作成する．
3. アリスは暗号文 $C$ を (他の誰かがのぞき見する恐れのある手段を用いて) ボブに送信する．
4. ボブは自分だけが知っている自分の秘密鍵 $sk_{Bob}$ とアリスから受け取った暗号文 $C$ を復号アルゴリズムに入力し，メッセージ $M$ を得る．

ボブの公開鍵 $pk_{Bob}$ で暗号化された暗号文 $C$ を復号できるのは，対応する秘密鍵 $sk_{Bob}$ を持っているボブだけなので，暗号文 $C$ を他の誰かがのぞき見する恐れのある手段で送っても，ボブ以外の誰かに暗号化されたメッセージ $M$ を盗み見られることはない．というのが公開鍵暗号の概念である．

公開鍵 $pk_{Bob}$ と対応する秘密鍵 $sk_{Bob}$ は無関連ではなく，数学的対応をもつ．もしも攻撃者が無限の計算能力をもっていたら，ボブの公開鍵 $pk_{Bob}$ からそれに対応する秘密鍵 $sk_{Bob}$ を直接計算してしまう．そうなれば，その攻撃者はボブへの暗号文 $C$ を好きに解読できることとなり，公開鍵暗号の概念は意味を失う．しかし，現実の攻撃者は無限の計算能力をもつわけではなく，それも一つの効率的なアルゴリズムにすぎない．攻撃者の計算能力を効率的なアルゴリズムの範囲に限定すれば，それが公開鍵 $pk_{Bob}$ を知っても秘密鍵 $sk_{Bob}$ を計算することはできない，ということがあり得ると考えるのは合理的である．

### 4.1.1 公開鍵暗号の定義

公開鍵暗号 (の実体) は，3 つの確率的で効率的なアルゴリズムの組 $\mathsf{ES} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ である：

- 鍵生成アルゴリズム $\mathsf{Gen}(k)$：
  指定されたセキュリティパラメータ $k$ に対応した公開鍵 $pk$ と秘密鍵 $sk$ のペアを生成する．(確率的アルゴリズムなので同じ $k$ について実行のたびに異なる鍵ペアを生成する．)

- 暗号化アルゴリズム $\mathsf{Enc}(pk, m)$：
  メッセージ $m$ を公開鍵 $pk$ のもとで暗号化し，暗号文 $c$ を生成する．

- 復号アルゴリズム $\mathsf{Dec}(sk, c)$：
  秘密鍵 $sk$ を用いて暗号文 $c$ を復号し，メッセージ $m$ を得る．

ただし，これらのアルゴリズムは

- （完全性)
  正しく計算された公開鍵で正しく暗号化された暗号文は対応する秘密鍵で必ず正しく復号される，すなわち $(pk, sk) \leftarrow \mathsf{Gen}(k),\ c \leftarrow \mathsf{Enc}(pk, m)$ ならば必ず，$\mathsf{Dec}(sk, c) = m$.

- （安全性)
  秘密鍵 $sk$ をもたないどのような効率的な攻撃者も，公開鍵 $pk$ と暗号文 $c$ から，もとのメッセージ $m$ に関するどのような部分情報も求められないこと．

を満たさなければならない．

## 4.2 エルガマル暗号

公開鍵暗号の例としてエルガマル暗号 [2] を示す．エルガマル暗号は離散対数問題の困難さを利用する公開鍵暗号で，その鍵生成・暗号化・復号の各アルゴリズムは以下の通り．

- 鍵生成アルゴリズム $\mathsf{Gen}(k)$：
  2 つの素数 $p, q$ と (1 より大きい) 整数 $g$ を $g^q \equiv 1 \pmod{p}$ となるように選ぶ．ただし，$q$ は $k$ ビット以上の長さになるようにとる．(このような $p, q, g$ を具体的にどのような方法で選び出すかはここでは気にしない．) 0 以上 $(q-1)$ 以下の範囲の乱数 $x$ を選び，$g$ の法 $p$ での $x$ 乗である $y = g^x \bmod p$ を計算する．組 $(p, q, g, x)$ を秘密鍵 $sk$ とし，組 $(p, q, g, y)$ を公開鍵 $pk$ とする．

- 暗号化アルゴリズム $\mathsf{Enc}(pk, m)$：
  公開鍵 $pk$ から $p, q, g, y$ を取り出す．0 以上 $(q-1)$ 以下の範囲の乱数 $r$ を選び，$g$ の法 $p$ での $r$ 乗である $c_1 = g^r \bmod p$ を計算する．さらに，公開鍵 (の主要素)$y$ をやはり法 $p$ で $r$ 乗し，それに暗号化したいメッセージ $m$ をかけて，積 $c_2 = my^r \bmod p$ を得る．$c_1$ と $c_2$ の組 $c = (c_1, c_2)$ を暗号文 $c$ として出力する．

- 復号アルゴリズム $\mathsf{Dec}(sk, c)$：
  秘密鍵 $sk$ から $p, q, g, x$ を取り出す．暗号文 $c$ から $c_1, c_2$ を取り出す．$c_1$ を秘密鍵 $x$ 乗したもので $c_2$ をわって，メッセージ $m = c_2/c_1^x \bmod p$ を得る．

ただしここで，暗号化の対象となるメッセージ $m$ は，鍵生成アルゴリズムが出力する $p, q, g$ について，0 以上 $(p-1)$ 以下の整数で表されているとする．(厳密には，メッセージ $m$ は単に 0 以上 $(p-1)$ 以下の整数であるだけでなく，$g$ のべき乗の形 (ある $\alpha$ について $g^\alpha \bmod p$) で表わされる数でないと安全性に問題が生じる．エルガマル暗号の悩ましいところである．)

エルガマル暗号の実行例を示す．見易さのため，セキュリティパラメータは小さくとり，$k = 3$ とする．(実際のセキュリティパラメータ $k$ は 160 以上にとる．)

- 鍵生成：
  2 つの素数として $p = 43,\ q = 7$ を選ぶと $g = 4$ について $4^7 \equiv 1 \pmod{43}$ となる．$q$ は $k = 3$ ビット．0 以上 $q - 1 = 6$ 以下の範囲の乱数として $x = 3$ を選んだとすると，$y = g^x \bmod p = 4^3 \bmod 43 = 21$ となる．組 $(p = 43, q = 7, g = 4, x = 3)$ を秘密鍵 $sk$ とし，組 $(p = 43, q = 7, g = 4, y = 21)$ を公開鍵 $pk$ とする．

- 暗号化：
  メッセージ $m = 11$ を暗号化したいとする．公開鍵 $pk$ から $p = 43, q = 7, g = 4, y = 21$ を取り出す．0 以上 $q - 1 = 6$ 以下の範囲の乱数として $r = 2$ を選んだとすると，$c_1 = g^r \bmod p = 4^2 \bmod 43 = 16$．さらに，積 $c_2 = my^r \bmod p = 11 \cdot 21^2 \bmod 43 = 35$ を得る．$m = 11$ の暗号文は $c = (16, 35)$ となる．

- 復号アルゴリズム $\mathsf{Dec}(sk, c)$：
  秘密鍵 $sk$ から $p = 43, q = 7, g = 4, x = 3$ を取り出す．暗号文 $c$ から $c_1 = 16, c_2 = 35$

を取り出す．復号結果は $c_2/c_1^x \bmod p = 35/16^3 \bmod 43 = 11$ となり，正しく復号された．

エルガマル暗号の完全性を確認する．正しく作られた公開鍵 $pk = (p, q, g, y)$ と秘密鍵 $sk = (p, q, g, x)$ の組は，$y \equiv g^x \pmod{p}$ を満たす．公開鍵 $pk = (p, q, g, y)$ を用いて，メッセージ $m$ を正しく暗号化した暗号文 $c = (c_1, c_2)$ は，ある $r$ について $c_1 = g^r, c_2 = my^r$ となっている．このとき，

$$\begin{aligned} c_2/c_1^x &\equiv (my^r)/(g^r)^x \pmod{p} \\ &\equiv (m(g^x)^r)/(g^r)^x \pmod{p} \\ &\equiv m \pmod{p}. \end{aligned}$$

よって，暗号文 $c$ は必ずもとのメッセージ $m$ に復号される．

エルガマル暗号の安全性について考察する．(秘密鍵 $sk$ をもたない) 効率的な攻撃者 $A$ が公開鍵 $pk = (p, q, g, y)$ と暗号文 $c = (c_1, c_2)$ を入手したとする．$c_1, c_2$ はある $r$ について，$c_1 = g^r, c_2 = my^r$ となっている．$c_2$ の形をみると，攻撃者 $A$ が $m$ を入手することは，それが $y^r$ を入手することと同じである．よって，エルガマル暗号が安全であるためには，攻撃者 $A$ にとって，$g, y, c_1 = g^r$ を知っていても $y^r$ を計算することが難しければよい．離散対数仮定を思い出すと，攻撃者 $A$ が $c_1 = g^r$ から $r$ を取り出してのち，それで $y$ をべき乗して $y^r$ を計算することは難しい．以上からエルガマル暗号は安全そうに見えるが，この議論はまだ次の 2 点において不十分である．

- $g, y, c_1 = g^r$ から $y^r$ を計算するのに，上のように途中で $r$ を導くことが必要とは限らない．他の計算方法があるかも知れない．離散対数仮定だけを安全性の根拠にするのは無理そうである．
- 安全性の定義では「もとのメッセージ $m$ に関するどのような部分情報も求められないこと.」としたが，上の議論では $m$ 全体を求めることを問題としており，$m$ の部分情報に関する考察がない．そもそも部分情報を求めるとはどういうことか？

以上の点を明確にできるように，まず安全性の定義を改める．

## 4.3 公開鍵暗号の安全性定義

### 4.3.1 公開鍵暗号の CPA 安全性

4.1.1 節の「公開鍵 $pk$ と暗号文 $c$ から，もとのメッセージ $m$ に関するどのような部分情報も求められない」という安全性定義は，以下のように “ゲーム” を用いると，より明確に記述することができる．公開鍵暗号 $\mathsf{ES} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ とそれに対する効率的な攻撃者 $\mathcal{A}$ について以下の CPA ゲームを考える．

**CPA ゲーム**: セキュリティパラメータ $k$ について，

1. (初期化.) $k$ を入力として Gen を実行し，公開鍵と秘密鍵のペア $(pk, sk)$ を生成する．

2. ($\mathcal{A}$ の呼び出し.) 公開鍵 $pk$ を与えて攻撃者 $\mathcal{A}$(のアルゴリズム) を呼び出す．$\mathcal{A}$ から，同じ長さの 2 つのメッセージからなる “チャレンジクエリ”$(m_0, m_1)$ を受け取ったら，$b$ としてランダムに 0 または 1 を選択する．そして，メッセージ $m_b$ を公開鍵 $pk$ のもとで Enc で暗号化して暗号文 (“チャレンジ暗号文”)$c'$ を生成し，$\mathcal{A}$ に渡す.

3. (判定.) $\mathcal{A}$ が出力 $\hat{b}$ で終了したら，$\hat{b} = b$ ならば攻撃者 $\mathcal{A}$ の勝ち，そうでないならば攻撃者 $\mathcal{A}$ の負けと判定する.

攻撃者 $\mathcal{A}$ は公開鍵 $pk$ をもとに，暗号文が解読しやすそうな，そしてその暗号文が区別しやすそうな，2 つの同じ長さのメッセージ $m_0, m_1$ を見つけ出し，チャレンジクエリ $(m_0, m_1)$ として提出する．そして，どちらかランダムに選ばれたメッセージ $m_b$ の暗号文をチャレンジ暗号文 $c'$ として受け取る．$\mathcal{A}$ はチャレンジ暗号文 $c'$ の解読を試み,（完全解読は無理としても）それが $m_0$ を暗号化したのかそれとも $m_1$ を暗号化したのか当てようとして，その推測結果 $\hat{b}$ を出力する．推測が当たっていたら（$\hat{b} = b$）攻撃者 $\mathcal{A}$ の勝ち，外れていたら攻撃者 $\mathcal{A}$ の負けとなる.

二者択一のゲームなので，攻撃者 $\mathcal{A}$ は何も考えなくても 1/2 の確率で CPA ゲームに勝つことができる．暗号文 $c'$ からもとのメッセージ $m_b$ の情報が少しでも漏れていたら，それを利用する攻撃者が，1/2 をその分超える確率で $b$ を言い当て，CPA ゲームに勝つはずである.

公開鍵暗号 ES が **CPA 安全**とは，どんな効率的な攻撃者 A も，上の CPA ゲームに高々 1/2（を本質的に超えない) の確率でしか勝てないことをいう.

### 4.3.2 エルガマル暗号の CPA 安全性

エルガマル暗号は，判定 DH 仮定と呼ばれる仮定のもとで，CPA 安全である．まず，判定 DH 仮定について説明する.（例によって）素数 $p, q$ および整数 $g$ を $g^q \equiv 1 \pmod{p}$ となるようにとる．$g^a \bmod p$ と $g^b \bmod p$ を与えられて $g^{ab} \bmod p$ を求める問題を計算 DH 問題という．計算 DH 問題は，離散対数問題が解けると解けるので離散対数問題よりも易しいが，難問と信じられている.

**計算 DH 仮定**

$p$ が大きな素数のとき，0 以上 $q-1$ 以下の範囲からランダムに選ばれた $a, b$ について，計算 DH 問題を効率的に解くアルゴリズムは存在しない.

計算 DH 問題をさらに易しくしたものが判定 DH 問題である：$g^a \bmod p$ と $g^b \bmod p$ と $g^c \bmod p$ を与えられて，$c \equiv ab \pmod{q}$ となっているか否か判定せよ．このようなことも分からないとは寧ろ意外なほどだが，判定 DH 問題も一般には困難な問題と信じられている.（つまり，計算 DH 問題は答えを教えてもらってもそれが正しいかどうか分からない．よって，計算 DH 問題の答えは 2.1 節の意味での知識にはならない，ということになる.）

**判定 DH 仮定**

$p$ が大きな素数のとき，0 以上 $q-1$ 以下の範囲からランダムに選ばれた $a, b, c$ について，$g^a \bmod p$，$g^b \bmod p$ そして $z = g^{ab} \bmod p$ または $z = g^c \bmod p$ を与えられ

て，$z$ がそのどちらなのか判定できる効率的なアルゴリズムは存在しない.（より正確には，$z$ は，それぞれ $1/2$ の確率で，$z = g^{ab} \bmod p$ または $z = g^c \bmod p$ であり，与えられた $z$ がこれらのどちらなのか $1/2$ を有意に超える確率で正しく判定することができる，効率的なアルゴリズムは存在しない，ということ.）

以上の準備のもと，エルガマル暗号の安全性はつぎのようであることが証明できる.

**定理 1** エルガマル暗号は判定 DH 仮定のもとで CPA 安全である.

なぜならば，エルガマル暗号の CPA 安全性を破る効率的な攻撃者 $\mathcal{A}$ が存在したとすると，それを用いて判定 DH 問題を効率的に解く効率的なアルゴリズム $\mathcal{D}$ が構成でき，存在することになるから．これは判定 DH 仮定に反する．というのが証明の流れである.

**証明** 対偶を示す．エルガマル暗号は判定 DH 仮定のもとで CPA 安全ではないと仮定する．CPA 安全性の定義より，ある効率的な攻撃者 $\mathcal{A}$ が存在し，それは CPA ゲームでエルガマル暗号に $1/2$ を有意に超える確率で勝つ，ことになる．そのような攻撃者 $\mathcal{A}$（のアルゴリズム）を用いて，判定 DH 問題を解くある効率的なアルゴリズム $\mathcal{D}$ を構成する．$\mathcal{D}$ には，0 以上 $q-1$ 以下の範囲からランダムに選ばれた $a, r, c$ について，組 $(g^a \bmod p, g^r \bmod p, z)$ が与えられる．ここで $z$ は，それぞれ $1/2$ の確率で，$z = g^{ar} \bmod p$ または $z = g^c \bmod p$ である．$\mathcal{D}$ の目的は，与えられた $z$ がこれらのどちらなのか $1/2$ を有意に超える確率で正しく判定することである.（以下，式の見易さのため $\bmod p$ は省略する.）

**アルゴリズム** $\mathcal{D}$: 組 $(g^a, g^r, z)$ を入力として，

1. (初期化.) 攻撃者 $\mathcal{A}$ を呼び出し，$pk = (p, q, g, g^a)$ をエルガマル暗号の公開鍵として与える（$g^a$ は $\mathcal{D}$ の入力の第 1 要素）.
2. (チャレンジクエリに対する応答.) $\mathcal{A}$ がチャレンジクエリ $(m_0, m_1)$ を提出したら，$b$ として 0 または 1 をランダムに選択し，$\pi = (g^r, zm_b)$ をチャレンジ暗号文として $\mathcal{A}$ に与える（$g^r$ は $\mathcal{D}$ の入力の第 2 要素，$z$ は第 3 要素）.
3. (出力.) $\mathcal{A}$ が出力 $\hat{b}$ で終了したら，$\hat{b} = b$ ならば $z = g^{ar}$ と判定し，そうでないなら $z = g^c$ と判定する.

アルゴリズム $\mathcal{D}$ へ入力された $z$ が $z = g^{ar}$ のとき（これは確率 $1/2$ で起きる），攻撃者 $\mathcal{A}$ が受け取っているチャレンジ暗号文 $\pi$ は

$$\pi = (g^r, zm_b) = (g^r, g^{ar} m_b) = (g^r, (g^a)^r m_b)$$

となる．これは，$pk = (p, q, g, g^a)$ を公開鍵とする，$m_b$ の正しいエルガマル暗号文である．よって，$z = g^{ar}$ のとき，アルゴリズム $\mathcal{D}$ は攻撃者 $\mathcal{A}$ と CPA ゲームを行っている．$\mathcal{A}$ は CPA ゲームに $1/2$ を有意に超える確率で勝つという仮定より，$\mathcal{A}$ が $b$ を言い当てる $(\hat{b} = b)$ 確率，すなわち，$\mathcal{D}$ が $z = g^{ar}$ と判定する確率は $1/2$ より有意に大きい．以上より，$z$ が $z = g^{ar}$ のとき，$\mathcal{D}$ が $z$ をそれと正しく判定する確率 $p_0$ は，$1/2$ より有意に大きい.

一方，アルゴリズム $\mathcal{D}$ へ入力された $z$ が $z = g^c$ のとき（これも確率 $1/2$ で起きる），チャレンジ暗号文 $\pi = (g^r, zm_b) = (g^r, g^c m_b)$ からは $b$ に関する情報が全く消えている．$c$

がランダムに選ばれているため，$b=0$ であろうと $b=1$ であろうと，$\pi$ の第 2 成分 $g^c m_b$ は同じランダムな分布となるからである．(厳密にはここで，以前に述べた，メッセージ $m_b$ が $g$ のべき乗の形 (ある $\alpha$ について $g^\alpha$) であるという条件が必要となる．$m_b = g^\alpha$ で $c$ がランダムならば，$\alpha$ がどうであろうと，$g^c m_b = g^{c+\alpha}$ は $\{g^0, g^1, \ldots, g^{q-1}\}$ 上ランダムに分布する．つまり，$b$ のランダムネスは $c$ のランダムネスで完全に上書きされる．$m_b$ が $g$ のべき乗でないときは，$c$ のランダムネスと $b$ のランダムネスの範囲がずれ，$\pi$ に $b$ の情報が残ってしまう.）よって，$\mathcal{A}$ が $b$ を言い当てる ($\hat{b}=b$) 確率は 1/2 である．($b$ のランダムネスは $c$ のランダムネスで完全に上書きされているので，$\mathcal{A}$ が $\hat{b}$ を出力してのち，コイン $b$ が振られたと思ってよい.）このとき，$\mathcal{D}$ が $z=g^c$ と正しく判定する確率 $p_1$ は 1/2 である．

以上より，$\mathcal{D}$ が入力を正しく判定する確率は $1/2p_0 + 1/2p_1 = 1/2p_0 + 1/4$ となるが，$p_0$ は 1/2 より有意に大きいので，これも 1/2 より有意に大きい．このようなアルゴリズム $\mathcal{D}$ の存在は，判定 DH 仮定に反する □

### 4.3.3 エルガマル暗号に対する能動的な攻撃

エルガマル暗号は CPA 安全ではあるものの，つぎのようなシナリオが示すように，より能動的な攻撃者に対しては脆弱である．

1. アリスは，オンラインショップ「ボブ」で商品を 10 個購入するために，注文個数 $m=10$ をエルガマル暗号で暗号化し，暗号文 $c=(g^r \bmod p, 10\cdot y^r \bmod p)$ をショップ「ボブ」に送信した．ボブの公開鍵を $(p,q,g,y(=g^x))$ としている．(簡単のためショップ「ボブ」では商品は 1 種類しか扱っていないとする.）

2. 攻撃者 $\mathcal{A}$ は送信途中のアリスの暗号文 $c=(c_1,c_2)$ を盗聴し，$c_2$ に 100 を掛け $c_2' = 100\cdot c_2 \bmod p$ とし，$c'=(c_1,c_2')$ を $\mathcal{A}$ 自身の注文の暗号文としてボブに送る．($c_1$ は変えない．)

3. 攻撃者 $\mathcal{A}$ から暗号文 $c'=(c_1,c_2')$ を受け取ったボブは，それを復号する：

$$c_2'/c_1^x = (100\cdot 10\cdot y^r)/(g^r)^x \bmod p = 1000.$$

   復号したボブは、注文者である攻撃者 $\mathcal{A}$ に「1000 個の注文を確かに受け取りました.」との確認メッセージを送る．ここで，攻撃者 $\mathcal{A}$ はアリスになりすまそうとしていたわけでもなく，ボブにとってまったく正当な注文者であることに注意する．(正当に見えるのではなく，事実正当な注文者である.）

4. 確認メッセージを受け取った攻撃者 $\mathcal{A}$ は 1000 を 100 でわって，アリスが商品を 10 個注文していたことを知る．つまり，横取りしたアリスの暗号文 $c=(c_1,c_2)$ の解読に成功した．その後、$\mathcal{A}$ は注文をキャンセルする．

### 4.3.4 公開鍵暗号の CCA 安全性

4.3.3 節のシナリオにあるような能動的な攻撃者をも考慮した安全性は CCA 安全性と呼

ばれ，以下の CCA ゲームを用いて定義される．公開鍵暗号 $\mathsf{ES} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ とそれに対する効率的な攻撃者 $\mathcal{A}$ について：

**CCA ゲーム**: $k$ をセキュリティパラメータとする．

1. (初期化.) $k$ を入力として Gen を実行し，公開鍵と秘密鍵のペア $(pk, sk)$ を生成する．
2. ($\mathcal{A}$ の呼び出し.) 公開鍵 $pk$ を与えて攻撃者 $\mathcal{A}$(のアルゴリズム) を呼び出す．$\mathcal{A}$ が満足するまで，$\mathcal{A}$ からのクエリに以下のように返答する．
   - (復号クエリ.) $\mathcal{A}$ から復号クエリ $c$ を受け取ったら，それがチャレンジ暗号文 $c'$ と同一でない限り，それ $c$ を秘密鍵 $sk$ のもとで Dec で復号し，結果 $m$ を $\mathcal{A}$ に返す．復号クエリは $\mathcal{A}$ が望むだけ，チャレンジクエリの前でも後でも，何度でも許される．
   - (チャレンジクエリ.) $\mathcal{A}$ から，同じ長さの 2 つのメッセージからなるチャレンジクエリ $(m_0, m_1)$ を受け取ったら，$b$ としてランダムに 0 または 1 を選択する．そして，メッセージ $m_b$ を公開鍵 $pk$ のもとで Enc で暗号化してチャレンジ暗号文 $c'$ を生成し，$\mathcal{A}$ に渡す．チャレンジクエリは一度のみ許される．
3. (判定.) $\mathcal{A}$ が出力 $\hat{b}$ で終了したら，$\hat{b} = b$ ならば攻撃者 $\mathcal{A}$ の勝ち，そうでないならば攻撃者 $\mathcal{A}$ の負けと判定する．

攻撃者 $\mathcal{A}$ がチャレンジクエリ $(m_0, m_1)$ を提出し，どちらかランダムに選ばれた $m_b$ の暗号文をチャレンジ暗号文 $c'$ として受け取り，それが $m_0, m_1$ のどちらを暗号化したのか当てようとして推測結果 $\hat{b}$ を出力する，のは CPA ゲームと同じである．

ただし，CCA ゲームでは，攻撃者 $\mathcal{A}$ は任意のタイミングで復号クエリを発することができる．$\mathcal{A}$ は復号クエリとそれへの回答を通じ，(自分で作り出した) 任意の暗号文の復号結果を知ることができる．CCA ゲームにおける攻撃者 $\mathcal{A}$ は，そのような復号クエリを活用しながら，チャレンジクエリに用いるメッセージ $m_0, m_1$ を選んだり，チャレンジ暗号文 $c'$ の解読を試みることができる．(ただし，チャレンジ暗号文 $c'$ そのものを復号クエリとすることは許されない．それを許すと攻撃者 $\mathcal{A}$ が回答として $m_b$ を直接に知ってしまい，ゲームにならないので.)

公開鍵暗号 ES が **CCA 安全**とは，どんな効率的な攻撃者 A も，CCA ゲームに高々$1/2$ (を本質的に超えない) の確率でしか勝てないことをいう．

エルガマル暗号は CCA 安全ではない．実際，CCA ゲームにおいて，以下の効率的な攻撃者 $\mathcal{A}$ が常にエルガマル暗号に勝つ．

**攻撃者** $\mathcal{A}$: 公開鍵 $pk = (p, q, g, y)$ を入力として，

1. (チャレンジクエリ.) 任意の異なる (同じ長さの) メッセージ $m_0, m_1$ を選び，チャレンジクエリ $(m_0, m_1)$ を提出する．返答として，チャレンジ暗号文 $c = (c_1, c_2)$ を受け取る．

2. (復号クエリ.) $c_2$ に $g$ をかけ $c_2' = c_2 g \bmod p$ とし，復号クエリとして $c' = (c_1, c_2')$ を提出する．返答として，メッセージ $m'$ を受け取る．

3. (推測.) $m'$ が $m_0 g \bmod p$ に等しいなら 0 を出力し，そうでないなら 1 を出力する．

この $\mathcal{A}$ は 4.3.3 節のシナリオの攻撃者 $\mathcal{A}$ と本質的に同じである．

## 4.4 署名付きエルガマル暗号

署名付きエルガマル暗号 [5, 8] は，エルガマル暗号に離散対数（$c_1 = g^r$ の $r$）の知識の証明を組み込んだ公開鍵暗号である．エルガマル暗号と同様に $r$ をランダムに選択し，さらに $r$ を知っていることの知識の証明 $\sigma$ をシュノアプロトコルによって計算し，暗号文 $c = (g^r, my^r, \sigma)$ に組み込む．ただし，シュノアプロトコルは対話的なプロトコルで，その実行には検証者からのチャレンジ $e$ が必要だった．署名付きエルガマル暗号では，ハッシュ関数[2] $H$ にその代役をつとめさせる．ハッシュ関数 $H$ はその値がランダムに見え，かつその値をコントロールできないので，ランダムなチャレンジを生成する（だけの）検証者の代わりとなるだろうという発想である．

署名付きエルガマル暗号の鍵生成・暗号化・復号の各アルゴリズムは以下の通り.（鍵生成アルゴリズムは，新たにハッシュ関数 $H$ を選ぶところ以外は，エルガマル暗号の場合とまったく同じである.）

- 鍵生成アルゴリズム $\mathsf{Gen}(k)$：
  2つの素数 $p, q$ と (1 より大きい) 整数 $g$ を $g^q \equiv 1 \pmod{p}$ となるように選ぶ．ただし，$q$ は $k$ ビット以上の長さになるようにとる．(このような $p, q, g$ を具体的にどのような方法で選び出すかはここでは気にしない．) ハッシュ関数 $H : \{0,1\}^* \to \{0, 1, \ldots, q-1\}$ を選択する．0 以上 $(q-1)$ 以下の範囲の乱数 $x$ を選び，$g$ の法 $p$ での $x$ 乗である $y = g^x \bmod p$ を計算する．組 $(p, q, g, x)$ を秘密鍵 $sk$ とし，組 $(p, q, g, y)$ を公開鍵 $pk$ とする．

- 暗号化アルゴリズム $\mathsf{Enc}(pk, m)$：
  公開鍵 $pk$ から $p, q, g, y$ を取り出す．0 以上 $(q-1)$ 以下の範囲の乱数 $r$ を選び，公開鍵 $pk$ の下での，$m$ のエルガマル暗号文 $(c_1 = g^r, c_2 = my^r)$ を計算する．さらに，$r$ が $c_1$ の離散対数であることの知識の証明 $\sigma$ をシュノアプロトコルを用いて計算する．すなわち，0 以上 $(q-1)$ 以下の範囲の乱数 $s$ を新たに選びコミットメント $u = g^s \bmod p$ を計算し，チャレンジ $e$ をハッシュ関数 $H$ の組 $(c_1, c_2, u)$ における値 $e = H(c_1, c_2, u)$ とし，それに対するレスポンス $z$ を知識 $r$ を用いて $z = s + er$ と計算する．$\sigma = (u, z)$ とする．先のエルガマル暗号文 $(c_1, c_2)$ に $\sigma$ を付加して，$c = (c_1, c_2, \sigma)$ とし暗号文として出力する．

- 復号アルゴリズム $\mathsf{Dec}(sk, c)$：
  秘密鍵 $sk$ から $p, q, g, x$ を取り出す．暗号文 $c$ から $c_1, c_2, \sigma = (u, z)$ を取り出す．ま

[2] 一般に，効率的で衝突困難な（圧縮）関数 $H : \{0,1\}^* \to \{0,1\}^h$ をハッシュ関数と呼ぶ．関数 $H$ が衝突困難とは，どのような効率的なアルゴリズムも $H(x) = H(y)$ となる $x, y$ $(x \neq y)$ を見つけることはできないことをいう．

ず，知識の証明 $\sigma$ を検証する．すなわち，$e$ をハッシュ関数 $H$ の組 $(c_1, c_2, u)$ における値 $e = H(c_1, c_2, u)$ としたとき，$(u, e, z)$ がシュノアプロトコルとして承認できるコミット・チャレンジ・レスポンスであるかを，検証式 $u = g^z c_1^{-e} \bmod p$ でチェックする．検証式が成り立てば，エルガマル暗号文 $c = (c_1, c_2)$ を秘密鍵 $sk$ を用いて復号し，メッセージ $m = c_2/c_1^x \bmod p$ を得る．成り立たないなら，復号を放棄する．

ただしエルガマル暗号のときと同様に，暗号化の対象となるメッセージ $m$ は，鍵生成アルゴリズムが出力する $p, q, g$ について，0 以上 $(p-1)$ 以下の整数で表されているとする．(厳密には，メッセージ $m$ は単に 0 以上 $(p-1)$ 以下の整数であるだけでなく，$g$ のべき乗の形 (ある $\alpha$ について $g^\alpha \bmod p$) で表わされる数であるとする．)

署名付きエルガマル暗号の完全性は，エルガマル暗号の完全性とシュノアプロトコルの安全性より，直ちに従う．

### 4.4.1 署名付きエルガマル暗号の安全性

署名付きエルガマル暗号の安全性をここで厳密に扱うことはできないが，エルガマル暗号を破った，先の能動的な攻撃者 $\mathcal{A}$ が署名付きエルガマル暗号に対しては通用しないことは確認しておきたい．

署名付きエルガマル暗号について，能動的な攻撃者 $\mathcal{A}$ がチャレンジ暗号文 $c = (g^r, c_2, \sigma)$ を取得したとする．$\mathcal{A}$ が暗号文 $c = (g^r, c_2, \sigma)$ から，その第 1 成分 $g^r$ を共有する形で別の暗号文 $c' = (g^r, c_2', \sigma')$ を作れるかどうか，検証したい．(これができるとエルガマル暗号に対してと同様に $\mathcal{A}$ は署名付きエルガマル暗号を破ることができる．)

暗号文 $c$ 中の $\sigma$ は，暗号文の作成者が暗号文の第 1 成分 $g^r$ について $r$ を知っていることの，シュノアプロトコルによる知識の証明だった．

まず，シュノアプロトコルのゼロ知識性によって，証明 $\sigma$ をもらっても $\mathcal{A}$ には，暗号文の第 1 成分 $g^r$ で用いられた「知識」$r$ は全くわからない．ゼロ知識性とは知識に関する情報をまったく与えないことだった．証明 $\sigma$ をつけることで，新たなぜい弱性を招いていることはなさそうである．

さらに，シュノアプロトコルの健全性によって，$\mathcal{A}$ は，与えられた暗号文 $c = (g^r, c_2, \sigma)$ から，$g^r$ を共有する別の暗号文 $c = (g^r, c_2', \sigma')$ を作ろうにも，$r$ を知らない以上，妥当な証明 $\sigma'$ をつくれない．$r$ が同じなのだから $\sigma'$ も同じ $\sigma$ でよさそうなものだが，$c_2$ が $c_2'$ に変わっているので（$c_2'$ まで $c_2$ と同じなら $\mathcal{A}$ に利得はない），$(g^r, c_2')$ についてハッシュ関数 $H$ が生成するチャレンジ $e' = H(g^r, c_2', u)$ も元の $e = H(g^r, c_2, u)$ とは異なるものになり，その結果正しいレスポンス $z'$ も元の $z$ とは異なるので，同じ $\sigma = (u, z)$ では駄目である．(逆に見れば，署名付きエルガマル暗号の構成で，もしも $e = H(c_1, c_2, u)$ とするところを，うっかり $c_2$ を省いて $e = H(c_1, u)$ としてしまっていたら，攻撃者 $\mathcal{A}$ に破られていたということである．このあたり，暗号技術の subtle（コワくてオモシロイ）なところといえる．)

以上より，攻撃者 $\mathcal{A}$ がチャレンジ暗号文 $c = (g^r, c_2, \sigma)$ から，$g^r$ を共有する形で別の暗号文 $c' = (g^r, c_2', \sigma')$ を作ることは不可能である．よって，エルガマル暗号を破った，先の能動的な攻撃者 $\mathcal{A}$ が署名付きエルガマル暗号に対しては通用しないことが直感的には了

解できた．

署名付きエルガマル暗号の安全性について，以下の定理が知られている．

**定理 2 (Schnorr, Jakobsson '00[7])** ランダムオラクルモデルとジェネリック群モデルにおいて，署名付きエルガマル暗号は CCA 安全である．

ランダムオラクルモデルとは，ハッシュ関数 $v \leftarrow H(u)$ の実行をランダムオラクルへの問い合わせに置き換える，アルゴリズムの実行モデルである．ランダムオラクルモデルにおいては，各アルゴリズムはハッシュ関数 $v \leftarrow H(u)$ の実行ステップに出くわすと，自身でそれを計算することはせず，その代わりにランダムオラクルに $u$ を問い合わせ，返答 $v$ をもらってその後の処理を続ける．ランダムオラクルは，あらかじめ（ある指定された出力長 $h$ をもつ）すべての関数の集合 $\{H : \{0,1\}^* \to \{0,1\}^h\}$ の中からランダムに関数 $H$ を一つ選択しておき，何か問い合わせ $u$ があったらその関数 $H$ の $u$ における値 $H(u)$ を返す存在である．現実にこのようなオラクルが存在していると思っているわけではなく，モデルとしての設定である．

また，ジェネリック群モデルとは，群演算の実行を群演算オラクルへの問い合わせに置き換える，アルゴリズムの実行モデルである．ジェネリック群モデルにおいては，各アルゴリズムは群演算 $h \leftarrow g_1^{a_1} \cdots g_i^{a_i}$ の実行ステップに出くわすと，自身でそれを計算することはせず，その代わりに群演算オラクルに $(g_1, \ldots, g_i), (a_1, \ldots, a_i)$ を問い合わせ，返答 $h$ をもらってその後の処理を続ける．各アルゴリズムが自身では決して群演算できないように，ジェネリック群モデルでは群の各要素はランダムな文字列で表現される．群演算オラクルは，問い合わせ $(g_1, \ldots, g_i), (a_1, \ldots, a_i)$ を受けると，群演算の結果 $g_1^{a_1} \ldots g_i^{a_i}$ （を表すランダムな文字列）を返す．やはり，現実にこのようなオラクルが存在していると思っているわけではない．

これらランダムオラクルモデルとジェネリック群モデルは，それぞれ，ハッシュ関数と群演算器に対し，ある種の理想化を施したものと考えることができる．ただし，その理想化の結果得られる，これらのモデルのもとでの安全性証明が，現実に対しどのような意味をもつと考えるべきか，よくわかっていないところがあり，意見の分かれるところである．

## 4.5 改良署名付きエルガマル暗号

署名付きエルガマル暗号では，暗号文 $c = (g^r, my^r, \sigma)$ に，シュノアプロトコルを用いた，$r$ の知識の証明 $\sigma$ を組み込んでいた．

改良署名付きエルガマル暗号は，シュノアプロトコルの代わりに，DH 指数 $(c_1 = g^x, z = h^x$ の $x)$ のゼロ知識証明プロトコル [1] を用いる．こうすることで，暗号化や復号にかかる計算コストは増えてしまうが，安全性証明においてジェネリック群モデルには頼らずにすむという利点がある．また，計算コストについても，暗号化において必要なべき乗演算はすべて暗号化対象のメッセージには依存しないため，それらを事前計算し結果を保存しておけるような環境では，暗号化は署名付きエルガマル暗号と同程度の速度で実行できる．

以下に，改良署名付きエルガマル暗号の鍵生成・暗号化・復号の各アルゴリズムを示す．

- 鍵生成アルゴリズム $\mathsf{Gen}(k)$：
  2 つの素数 $p, q$ と (1 より大きい) 整数 $g$ を $g^q \equiv 1 \pmod{p}$ となるように選ぶ．ただし，$q$ は $k$ ビット以上の長さになるようにとる．(このような $p, q, g$ を具体的にどのような方法で選び出すかはここでは気にしない．) 2 つのハッシュ関数 $H : \{0,1\}^* \to \{g^i \bmod p \mid i = 0, 1, \ldots, q-1\}$ および $G : \{0,1\}^* \to \{0, 1, \ldots, q-1\}$ を選択する．0 以上 $(q-1)$ 以下の範囲の乱数 $a$ を選び，$g$ の法 $p$ での $a$ 乗である $b = g^a \bmod p$ を計算する．組 $(p, q, g, a)$ を秘密鍵 $sk$ とし，組 $(p, q, g, b)$ を公開鍵 $pk$ とする．

- 暗号化アルゴリズム $\mathsf{Enc}(pk, m)$：
  公開鍵 $pk$ から $p, q, g, b$ を取り出す．0 以上 $(q-1)$ 以下の範囲の乱数 $x$ を選び，公開鍵 $pk$ の下での，$m$ のエルガマル暗号文 $(c_1 = g^x, c_2 = mb^x)$ を計算する．さらに，DH 指数プロトコルを用いて $c_1$ について $x$ の知識の証明 $\sigma$ を計算する．すなわち，0 以上 $(q-1)$ 以下の範囲の乱数 $k$ を新たに選び，$u = g^k \bmod p$, $h = H(u, c_1)$ を計算し，さらに $z = h^x$, $v = h^k$ とする．チャレンジ $c$ をハッシュ関数 $G$ の組 $(c_1, c_2, h, z, u, v)$ における値 $c = G(c_1, c_2, h, z, u, v)$ とし，それに対するレスポンス $s$ を知識 $x$ を用いて $s = k + cx$ と計算する．$\sigma = (z, s, u, v)$ とする．先のエルガマル暗号文 $(c_1, c_2)$ に $\sigma$ を付加して，$c = (c_1, c_2, \sigma)$ とし暗号文として出力する．

- 復号アルゴリズム $\mathsf{Dec}(sk, c)$：
  秘密鍵 $sk$ から $p, q, g, a$ を取り出す．暗号文 $c$ から $c_1, c_2, \sigma = (z, s, u, z)$ を取り出す．まず，知識の証明 $\sigma$ を検証する．すなわち，$c$ をハッシュ関数 $G$ の組 $(c_1, c_2, h, z, u, v)$ における値 $c = G(c_1, c_2, h, z, u, v)$ としたとき，$(u, v, c, s)$ が DH 指数プロトコルとして承認できるコミット･チャレンジ･レスポンスであるかを，検証式 $u \equiv g^s c_1^{-c}$, $v \equiv h^s z^{-c} \pmod{p}$ でチェックする．検証式が成り立てば，エルガマル暗号文 $c = (c_1, c_2)$ を秘密鍵 $sk$ を用いて復号し，メッセージ $m = c_2/c_1^a \bmod p$ を得る．成り立たないなら，復号を放棄する．

改良署名付きエルガマル暗号の安全性について，以下の定理が証明されている．

**定理 3 (石井，鶴留，有田 '09 [9])** ランダムオラクルモデルにおいて， 決定的 DH 仮定のもとで改良署名付きエルガマル暗号は CCA 安全である．

### 参考文献


[1] B. Chevallier-Mames, An Efficient CDH-based Signature Scheme With a Tight Security Reduction, CRYPT 2005, pp. 1-27, 2005.

[2] T. ElGamal, A Public-key Cryptosystem and a Signature Scheme Basedon Discrete Logarithms, CRYPTO 84, LNCS 196, pp.10-18, Springer-Verlag, 1985

[3] Fiat and A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, Proc. Of Crypto '86, LNCS 263.

[4] O. Goldreich, Foundations of Cryptography: Volume 1 - Basic Tools, Cambridge University Press, 2001.

[5] M. Jakobsson, A Practical Mix, Eurocrypt '98, LNCS 1403, pp. 448-461, 1998.

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

[7] C. P. Schnorr and M. Jakobosson, Security of Signed ElGamal Encryption, ASIACRYPT 2000, LNCS 1976, pp. 73-89, 2000.

[8] Y. Tsiounis and M. Yung, On the Security of ElGamal Based Encryption, PKS' 98, LNCS 1431, pp. 117-134, 1998.
[9] 石井 貴之, 鶴留 浩児, 有田 正剛, Signed エルガマル暗号の一改良について, 2009 年暗号と情報セキュリティシンポジウム, SCIS 2009, 2B2-1, 2009/1.