平成23年度

論文題目

ハッシュ関数を用いた安全なｎチャネルメッセージ伝送

神奈川大学 工学研究科 電気電子情報工学専攻

学籍番号 201070083

栗山 知也

指導担当者　木下　宏揚 教授

目 次

図 目 次

表 目 次

第1章

序論

1.1　背景

従来の公開鍵暗号方式では公開鍵の正当性を証明するために認証局のような信頼できる第三者機関が必要であった (図 1.1)。それに対して n チャネルメッセージ伝送方式では事前の鍵が不要なため第三者機関も必要ない。そこで本論文では n チャネルメッセージ伝送方式に着目している。n チャネルメッセージ伝送方式とは n 本の通信路を使用して安全に送信する暗号化通信方式である。もし n 本のうちの何本かに文書を盗聴・改ざんする敵が潜んでいても、残りの通信路の情報を用いて文書を復号することができる (図 1.2)。

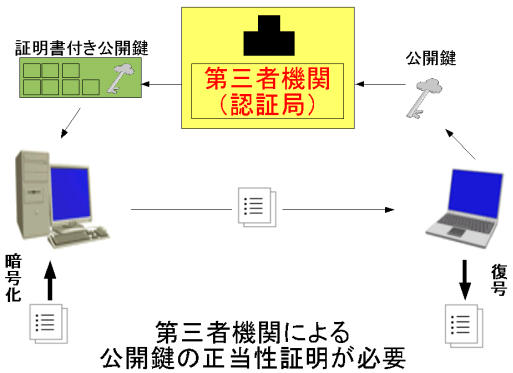

図 1.1　従来の公開鍵暗号方式

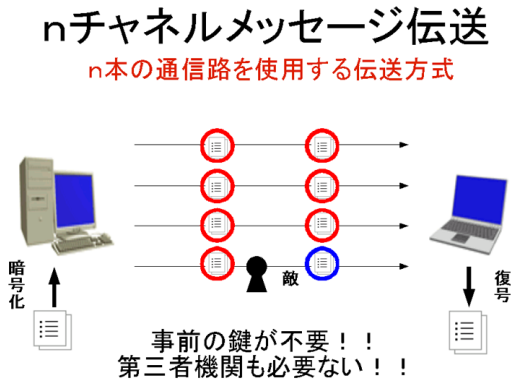

図 1.2　n チャネルメッセージ伝送方式

n チャネルメッセージ伝送方式にはPSMT(Perfectly Secure Message Transmission) という方式がある。PSMT の安全性の定義は以下のとおりである。

1. 敵は送信メッセージに関する情報を何も得られない。(盗聴耐性)
2. 受信者がメッセージを正しく受信できる確率が 100 %である。(改竄耐性)

また、送信者が受信者に 1 回送信するだけで済む方式を 1-round 方式、送信者と受信者が相互に r 回やり取りを行う方式を r-round 方式と呼ぶ。
PSMT は 1993 年に Dolev ら [1] によって提案された。彼らは、敵が n 本の通信路のうち t 本に潜んでいるとしたときに PSMT プロトコルが存在するための必要十分条件は、1-round 方式では $n \geqq 3t+1$、2-round 方式では $n \geqq 2t+1$ であることを証明した。PSMT においては、$n \geqq 3t+1$ でなければ使えない 1-round 方式よりも、$n \geqq 2t+1$ で使える 2-round 方式のほうが優れている。そこで、1-round 方式における必要十分条件を $n \geqq 2t+1$ に改善するために生まれたのが ASMT(Almost Secure Message Transmission) である。
ASMT の安全性の定義は以下のとおりである。

1. 敵は送信メッセージに関する情報を何も得られない。(盗聴耐性)
2. 受信者がメッセージを正しく受信できる確率が $1-\delta$ 以上である。(改竄耐性)
3. 受信者が正しく受信できない確率が δ であり、そのとき受信者は failure を出力できる。(失敗検知能力)
4. 敵が t 本の通信路を遮断しても受信者は残りの通信路で得た情報だけからメッセージを受信できる。(遮断耐性)

ASMT は 2004 年に Srinathan ら [2] によって提案されたが、そのプロトコルには間違いがあった。その後 2007 年に Kurosawa ら [3] によって厳密に定義された。そのなかで $n = 2t+1$ のときの通信効率の限界が示され、限界に近い通信量で通信できるプロトコルが提案された。上記のプロトコルは計算量が指数関数的であるという問題があった。

1.2 本研究の目的

木下研究室では計算量を多項式時間に改善したものを提案した。
しかし厳密な通信効率やCanonicalであることの証明などはなされていなかった。本研究ではそれらの項目に取り組み、以前提案したプロトコルを完成させる。また、実装についての研究も以前から取り組んでおり、実装での問題点について議論し、実装する。

第2章

数学的基礎

2.1　合同式

整数環 Z において、整数 m と n を整数 h で割った余りが等しいとき

$$m \equiv n \mod h$$

と記して、m と n とを同一視することを h を法 (modulo h、略して mod h) として合同という。具体例として $h = 5$ である場合、例えば $12 \equiv 7 \equiv 2 \mod 5$ であるから 12、7、2 は 5 を法として同一視される。すなわち合同式を用いると、無数に存在する整数をある整数で割り、同じ余りを有するものどうしを同様の性質をもつ数として分類し、取り扱うことができるのである。

また合同式では整数 a、b、c、d、自然数 m に対して以下のことが成り立つ。

1. $a \equiv a \mod m$ (反射律)
2. $a \equiv b \Rightarrow b \equiv a \mod m$ (対称律)
3. $a \equiv b$、$b \equiv c \Rightarrow a \equiv c \mod m$ (推移律)
4. $a \equiv b$、$c \equiv d \mod m \Rightarrow a + c \equiv b + d \mod m$

   $a - c \equiv b - d \mod m$

   $ac \equiv bd \mod m$
5. もし $ac \equiv bc \mod m$ かつ $gcd(c, m) = 1$ ならば $a \equiv b \mod m$

合同式については [4]、[5] を参照。

2.2 合同式の除法

合同式は加法・減法・乗法・については良い性質を持ち、扱いも簡単である。しかし除法についてはそうではない。そもそも合同式は整数の範囲での記法である。整数を整数で割った場合、整数にならないこともあるので、合同式で除法を考えるときは注意が必要である。例えば、$ab \equiv ac \mod m$ は両辺 a で割って $b \equiv c \mod m$ とすることができるとは限りません。実際 $3 \times 5 \equiv 15 \equiv 9 \equiv 3 \times 3 \mod 6$ ですが、$5 \equiv 3 \mod 6$ ではない。しかし除法を注意深く、上手に解釈すると除算ができる場合もある。そのために、合同式の一次方程式 (一次合同式) を考えてみる。

2.2.1 一次合同式

a、b を整数とするとき、$ax \equiv b \mod m$ となる x を求めることを一次合同式を解くという。合同式の解は m を法として決まるため x が解で $x \equiv x' \mod m$ ならば、x' も解になる。一次合同式は解を持たないこともある。例えば $3x \equiv 1 \mod 6$ は解を持たない。実際この場合 x に解があったとすると、合同式の定義から、$3x - 1$ が 6 で割り切れることになり矛盾となる。解の存在については、次のことが成立する。

- 一次合同式 $ax \equiv b \mod m$ が解 x を持つための必要十分条件は a と m の最大公約数 $gcd(a, m)$ が b を割り切ることである。

[証明]
(必要性)
x を $ax \equiv b \mod m$ の解とする。このとき、$ax - b$ は m で割り切れるので、その商を k とすると、$ax - b = km$ と表される。b と km を移項すると、$ax - km = b$ となり、左辺は $gcd(a, m)$ で割り切れる。したがって、$gcd(a, m)$ は右辺 b を割り切る。

(十分性)
$gcd(a,m)$ は右辺 b を割り切るとし、その商を s とする。このとき $b = s \times gcd(a,m)$ である。ここで後述する拡張ユークリッドの互除法の結果を使うと、

$$ax' + my' = gcd(a,m)$$

となる x' と y' が存在する。この両辺に s を掛けると

$$asx' + msy' = s \times gcd(a,m) = b$$

となる。ここで $x = s \times x'$ と置くと、$ax - b = -msy'$ であるから、$ax \equiv b \mod m$ となる。すなわち、x が解になる。

この証明から解 x は拡張ユークリッドの互除法から計算できることが分かる。上の性質で $b = 1$ の場合が特に重要である。この場合 $gcd(a,m)$ が 1 を割り切るとは $gcd(a,m) = 1$ に他ならない。よって次のことが成立する。

- 一次合同式 $ax \equiv 1 \mod m$ が解 x を持つための必要十分条件は a と m の最大公約数 $gcd(a,m) = 1$ となることである。

この場合、解 x は m を法にして唯一つしかないことも分かる。すなわち、x と x' が共に $ax \equiv 1 \mod m$ の解だとすると、$x' \equiv x \mod m$ となる。実際 $ax \equiv 1 \mod m$、$ax' \equiv 1 \mod m$ とすると、$axx' \equiv x' \mod m$ となる。したがって、

$$x' \equiv axx' \equiv ax'x \equiv 1 \times x \equiv x \ mod\, m$$

となる。このことから m を法とする逆元を定義することができる。

2.2.2 逆元

$gcd(a,m) = 1$ となるとき、一次合同式 $ax \equiv 1 \ mod\, m$ の解 x が m を法にして唯一つ存在する。その x を m を法とする a の逆元という。m を法とする a の逆

元は $gcd(a, m) = 1$ となる a に対してのみ定義される。また m を法とする a の逆元は整数としては唯一つではないが、m を法にした場合は唯一つである。この m を法とする a の逆元 x は整数であるから、他の数に掛けることができる。ここで $ax \equiv 1 \mod m$ であるから、x を掛けることは m を法として考えている限りは a で割ることを意味する。したがって合同式で除法を行う場合は、次のように逆元を用いて行う。

$$a \div b \equiv a \times (b \text{の逆元}) \mod m$$

合同式の除法については [6] を参照。

2.3 ユークリッドの互除法

ユークリッドの互除法は、2つの自然数の最大公約数を導き出すアルゴリズムである。素因数分解に比べて効率よく計算できる。互除法で2つの自然数 a、$b(a > b)$ の最大公約数を見つけるには、次の手続きを用いる。

1. a を b で割り、余り r とする。
2. $r = 0$ の場合は、最大公約数は b であり、手続きは終わりになる。
3. $r \neq 0$ の場合は、a と b の組を b と r に置き換えて、最初の手続きにもどる。

つまり、この1から3の手続きを繰り返して、余りが0になったときに割った数が、最大公約数となるわけである。言い換えれば、余り0を得たときの直前のステップで得た余りが、最大公約数ということになる。

例として、1365 と 77 の最大公約数をユークリッドの互除法で求めてみる。

$$1365 = 17 \times 77 + 56 \quad (\leftarrow 1365 \div 77 = 17 \text{ 余り } 56 \text{ の計算による})$$
$$77 = 1 \times 56 + 21 \quad (\leftarrow 77 \div 56 = 1 \text{ 余り } 21 \text{ の計算による})$$
$$56 = 2 \times 21 + 14 \quad (\leftarrow 56 \div 21 = 2 \text{ 余り } 14 \text{ の計算による})$$
$$21 = 1 \times 14 + \underline{7} \quad (\leftarrow 21 \div 14 = 1 \text{ 余り } 7 \text{ の計算による})$$
$$14 = 2 \times \underline{7} + 0 \quad (\leftarrow 14 \div 7 = 2 \text{ 余り } 0 \text{ の計算による})$$

となり、最大公約数は 7 となる。
次に互いに素な 20 と 17 で、互除法を用いて最大公約数を求めてみる。

$$20 = 1 \times 17 + 3 \tag{2.1}$$
$$17 = 5 \times 3 + 2 \tag{2.2}$$
$$3 = 1 \times 2 + 1 \tag{2.3}$$
$$2 = 2 \times 1 + 0$$

最大公約数は、当然ながら 1 なので、互除法を使う必要がないように感じられるが、その結果を求める過程の式に大きな利用価値がある。まず式 (2.1)、(2.2)、(2.3) を移項して、次の 3 つの式を得る。

$$20 - 1 \times 17 = 3 \tag{2.4}$$
$$17 - 5 \times 3 = \underline{2} \tag{2.5}$$
$$3 - 1 \times \underline{2} = 1 \tag{2.6}$$

次に式 (2.6) の $\underline{2}$ に式 (2.5) を代入して、3 と 17 に注目してくくる。

$$3 - 1 \times \underline{2} = 3 - 1 \times (17 - 5 \times 3) = 6 \times \underline{3} - 1 \times 17 = 1 \tag{2.7}$$

さらに、式 (2.7) の $\underline{3}$ に式 (2.4) を代入し、20 と 17 に注目してくくる。

$$6 \times 3 - 1 \times 17 = 6 \times (20 - 1 \times 17) - 1 \times 17 = 6 \times 20 - 7 \times 17 = 1$$

この一連の手続きから得た結果を、次のように書き換える。

$$20 \times 6 + 17 \times (-7) = 1$$

上の式は、$ax + by = c$という形になっていて、a、b、c、x、yにあたる数はすべて整数である。このような形の方程式は一次不定方程式といい、整数解のxとyを求めるものである。つまり、ユークリッド互除法の計算過程を利用することで、$a = 20$、$b = 17$のとき、一次不定方程式の整数解$(x, y) = (6, -7)$が得られるということが示されている。この方法は拡張ユークリッドの互除法と呼ばれ、非常に利用価値の高いアルゴリズムである。一般にaとbを0でない整数とし、aとbの最大公約数をcとすると、一次不定方程式

$$ax + by = c$$

は、整数解(x_1, y_1)を持ち、解の1組は、拡張ユークリッドの互除法を用いて求めることができる。ただし、一次不定方程式の解は、1組だけではない。方程式すべての整数解は、任意の整数kを用いて次のように表される。

$$(x, y) = \left(x_1 + k \cdot \frac{b}{c} \, , \; y_1 - k \cdot \frac{a}{c} \right) \tag{2.8}$$

式(2.8)に示す解の公式を用いれば、一次不定方程式$20x + 17y = 1$のすべての整数解は、次のようになる。

$$(6 + 17k, -7 - 20k) \tag{2.9}$$

$k = -1$の場合、解は$(x, y) = (-11, 13)$である。これを一次不定方程式$20x + 17y = 1$に代入する。

$$20 \times (-11) + 17 \times 13 = 1$$

移項して、式を整える。

$$17 \times 13 = 1 + 11 \times 20 \tag{2.10}$$

式 (2.10) をよく見ると、実は、次式と同じ意味であることがわかる。

$$17 \times 13 \equiv 1 \quad mod\ 20 \tag{2.11}$$

2.2.2 で、$ax \equiv 1 \quad mod\ m$ の場合、「x は m を法とする a の逆元である」と説明した。すなわち、式 (2.11) は 20 を法にして 13 が 17 の乗算に対する逆元であることを意味する。つまり、拡張ユークリッドの互除法を使えば、合同式での逆元が効率よく導き出すことができるのである [7]。

2.4　ラグランジェの補間公式

xy 座標上に k 個の点 $(i, f(i))$ が与えられたとき、それら全てを通る $k-1$ 次関数 $f(x)$ は唯一に定まる。この $f(x)$ を求める方法としてラグランジェの補間公式と呼ばれる次の公式がある。

$$f(x) = \lambda_1(x)f(i_1) + \cdots + \lambda_k(x)f(i_k)$$

ただし

$$\lambda_j(x) = \frac{(x-i_1)\cdots(x-i_{j-1})(x-i_{j+1})\cdots(x-i_k)}{(i_j-i_1)\cdots(i_j-i_{j-1})(i_j-i_{j+1})\cdots(i_j-i_k)}$$

例えば、二次関数 $f(x) = ax^2 + bx + c$ 上の点を $(x_1, f(x_1))$、$(x_2, f(x_2))$、$(x_3, f(x_3))$ として、ラグランジェの補間公式を用いると $f(x)$ は以下のように表せる。

$$f(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}f(x_1) + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}f(x_2) + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}f(x_3)$$

ラグランジェの補間公式については [8] を参照。

第3章

暗号理論の基礎

3.1　公開鍵暗号

公開鍵暗号は、その概念が 1976 年にディフィーとヘルマンによって提案された。具体的な公開鍵暗号化方式の最初のものは、1978 年にリベスト (R.L.Rivest)、シャミア (A.Shamir)、エードルマン (L.Adleman) により提案され、RSA 暗号と呼ばれた。その後、ElGamal 暗号、楕円曲線暗号など多くの公開鍵暗号が発明されたが、現在でも RSA 暗号が最も広く用いられている。

公開鍵暗号を用いる場合には、各ユーザは自分の固有の秘密鍵と公開鍵の対を生成する。いま、A が B に暗号化通信をしたい場合には、B はまず、自分の公開鍵を A に伝える。A はこの公開鍵を用いてメッセージを暗号化し、B に送る。B は自分の秘密鍵を用いて、A から送られた暗号文を復号できる。この方式の利点は秘密鍵がユーザごとに一つだけであるから、その管理が非常に楽になることである。しかし、この公開鍵暗号には大きな問題が二つある。

一つは計算量の問題である。暗号化鍵と復号鍵の非対称性を実現するために、公開鍵暗号の暗号化、復号にはかなりの計算量を要することになる。このため、公開鍵暗号は大量の情報の高速な守秘伝送には向いていない。もう一つは公開鍵の管理の問題である。暗号化の場合でもディジタル署名の場合でも、A

やBの公開鍵が間違いなく本人のものであることが保証されないかぎり、安全な通信や認証が行えない。このような保証を与える仕組みの一つが公開鍵認証基盤 (PKI：public key infrastructure) である。これは認証機関 (CA：certification authority) が各ユーザに発行する公開鍵証明書によって、本人の公開鍵であることの認証を行う方式であり、電子政府の基盤となっている方式である [9]。最後に公開鍵暗号系の中でも重要なものをいくつか簡単に紹介する [10]。

RSA

RSA の安全性は大きな整数の因数分解の難しさに基づいている。

Merkle-Hellman ナップサック

これに関係したシステムは部分集合和問題の難しさに基づいている。しかしながら、様々なナップサック系の全てが危険であることが明らかにされている。(Chor-Rivest 暗号系を除く)

McEliece

McEliece 暗号系は、代数的符号理論に基づいており、いまだ安全であると信じられている。この理論は線形符号の復号問題に基づいている。

ElGamal

ElGamal 暗号系は有限体上の離散対数問題の難しさに基づいている。

Chor-Rivest

これも“ ナップサック ”型暗号系の一種であるが、まだ安全だとみなされている。

3.2 秘密分散共有法

秘密鍵を紛失から守るためには、そのコピーを作って複数の場所に保管しておくことが望ましい。しかし、コピーの数を多くすると盗難の危険が増大してしまう。一方、コピーの数を少なくすると、すべてを紛失してしまう危険が増大する。この相矛盾する二つの問題を解決する方法が、秘密分散共有法である。

秘密分散共有法は、秘密 s の保有者 (ディーラ) と、複数の分散管理者の間で行われる。以降、ディーラを D、n 人の分散管理者を $P_1, \cdots, P_n$ で表す。秘密分散共有法は図 3.1 のように分散段階と再構成段階から構成される。

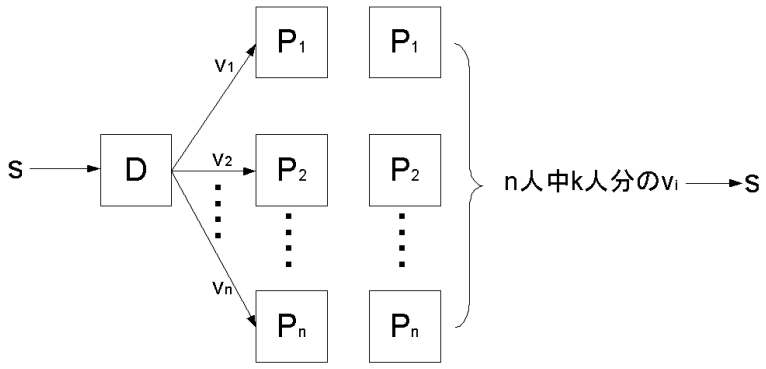

図 3.1 秘密分散共有法

分散段階

図 (a) のように、D は、秘密 s から、n 個のシェアと呼ばれる $v_1, \cdots, v_n$ を計算し、v_i を P_i に与える。

再構成段階

図 (b) のように、n 人の分散管理者のうち何人かが集まり、シェアから s を求める。

(k, n) しきい値秘密分散共有法 (しきい値法) では、$v_1, \cdots, v_n$ は次の条件を満たすように作成されていなければならない。

(条件 1) $v_1, \cdots, v_n$ のうち、任意の k 個から元の秘密 s を復元できる。

(条件 2) どの $k-1$ 個を集めても、s について何も分からない。

秘密分散共有法について詳しくは [8] を参照。

3.3 ハッシュ関数

長いメッセージを短い k ビットに圧縮する関数をハッシュ関数という。ハッシュ関数は、ディジタル署名など、暗号の多くの分野で利用される。

衝突困難なハッシュ関数

ハッシュ関数 H においては、長いメッセージを短くするので、必ず

$$H(x) = H(x')$$

となる衝突ペア (x, x') が存在する。そのような衝突ペア (x, x') を効率よく見つけるのが困難なハッシュ関数を衝突困難なハッシュ関数と呼ぶ。上記のことも含めてハッシュ関数には以下のような安全性が定義されている [10]。

Collision Resistance(衝突困難性)

ハッシュ関数 H は、$x' \neq x$ かつ $H(x') = H(x)$ であるような x と x' を見つけることが計算量的に困難である。

Preimage Resistance(一方向性)

$H(x) = z$ である x を見つけることが計算量的に不可能である。また、このようなハッシュ関数 H を一方向性であるという。

Second Preimage Resistance

x に対して $H(x') = H(x)$ であるような x とは別の x' を見つけることが計算量的に困難である。

固定長圧縮関数

k ビットより長い一定の長さのメッセージを、k ビットに圧縮する関数を固定長圧縮関数と呼ぶ。実際のハッシュ関数の設計においては、長いメッセージに対して固定長圧縮関数を繰り返し適用することにより、可変長の入力に対するハッシュ値を求める、という方針がとられている。

以下に代表的なハッシュ関数について簡単に説明する。

MD4

衝突を有するが、任意の長さの平文を 128 ビットに圧縮する方法で、32 ビットを単位とした演算をする。非常に高速である。

MD5

MD4 をさらに高速にしたものである。

SHA-1

米国政府のシステム調達基準である FIPS 180-1 の中で規定されている代表的なハッシュ関数である。2^{64} ビット未満の任意の長さのメッセージを 160 ビットに圧縮する。

ハッシュ関数については [8]、[10]、[11] を参照。

第4章

nチャネルメッセージ伝送方式

従来の公開鍵暗号方式では公開鍵の正当性を証明するために認証局のような信頼できる第三者機関が必要であった。それに対して n チャネルメッセージ伝送方式では事前の鍵が不要なため第三者機関も必要ない。そこで本論文では n チャネルメッセージ伝送方式に着目している。

n チャネルメッセージ伝送方式は文書を n 本の通信路を使用して安全に送信する暗号化通信方式である。もし n 本のうちの何本かに文書を盗聴・改竄する敵が潜んでいても、残りの通信路の情報を用いて文書を復号することができる (図 4.1)。n チャネルメッセージ伝送方式には PSMT と ASMT という方式がある。

図 4.1　n チャネルメッセージ伝送方式

4.1 PSMT(Perfectly Secure Message Transmission)

4.1.1 PSMTにおける安全性の定義

n チャネルメッセージ伝送方式において次の 2 つの条件を満たしたものを PSMT と呼ぶ。

1. 敵は送信メッセージに関する情報を何も得られない。(盗聴耐性)
2. 受信者がメッセージを正しく受信できる確率が 100 %である。(改竄耐性)

また、送信者が受信者に 1 回送信するだけで済む方式を 1-round 方式、送信者と受信者が相互に r 回やり取りを行う方式を r-round 方式と呼ぶ (図 4.2)。
このとき敵が n 本の通信路のうち t 本に潜んでいるとしたときに PSMT プロトコルが存在するための必要十分条件は、1-round 方式では $n \geqq 3t+1$、2-round 方式では $n \geqq 2t+1$ であることが証明されている。

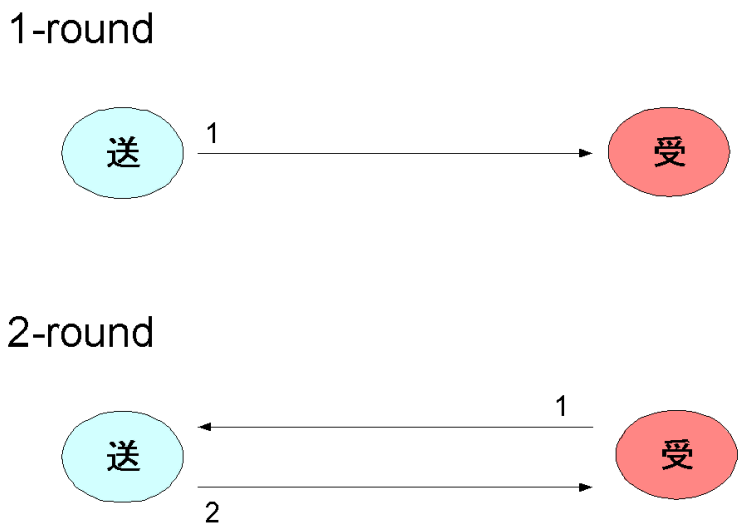

図 4.2 1-round 方式と 2-round 方式

4.1.2 PSMTの歴史

PSMTは1993年にDolevら[1]によって提案された。彼らは、敵がn本の通信路のうちt本に潜んでいるとしたときにPSMTプロトコルが存在するための必要十分条件は、1-round方式では$n \geqq 3t+1$、2-round方式では$n \geqq 2t+1$であることが証明し、また、それぞれの通信量が$O(n)$、$O(2^n)$のプロトコルを提案した。その後2-round方式は1996年にSayeedら[12]が通信量$O(n^3)$のプロトコルを提案し、2006年にはAgarwalら[13]が大量の文書を送ることを前提に通信量$O(n)$で計算量が指数関数的であるプロトコルを提案した。そして2008年にはKurosawaら[14]が通信量$O(n)$、計算量$O(n^3)$となるプロトコルを提案した。このように2-round方式の通信量、計算量は改善されていった。一方、1-round方式は、通信量$O(n)$、計算量が多項式時間となるプロトコルをDolveらが最初に提案しており、既にそれが$n \geqq 3t+1$における最善のプロトコルであった。PSMTにおいては、$n \geqq 3t+1$でなければ使えない1-round方式よりも、$n \geqq 2t+1$で使える2-round方式のほうが優れている。そこで、1-round方式における必要十分条件を$n \geqq 2t+1$に改善するために生まれたのが次に述べるASMTである。

	PSMT	
	2-round	1-round
Dolev 1993	$n \geqq 2t+1$が必要十分条件 通信量：$O(2^n)$	$n \geqq 3t+1$が必要十分条件 通信量：$O(n)$
Sayeed 1996	通信量：$O(n^3)$	
Agarwal 2006	通信量：$O(n)$ ただし大量の文書を送ることが前提 計算量：指数的	
Kurosawa 2008	通信量：$O(n)$ 計算量：$O(n^3)$	

表 4.1 PSMTの歴史

4.2 ASMT(Almost Secure Message Transmission)

4.2.1 ASMT における安全性の定義

ASMT における安全性の定義は以下のとおりである。

1. 敵は送信メッセージに関する情報を何も得られない。(盗聴耐性)
2. 受信者がメッセージを正しく受信できる確率が $1-\delta$ 以上である。(改竄耐性)
3. 受信者が正しく受信できない確率が δ 以下であり、そのとき受信者は failure を出力できる。(失敗検知能力)
4. 敵が t 本の通信路を遮断しても受信者は残りの通信路で得た情報だけからメッセージを受信できる。(遮断耐性)

PSMT と比較して主に異なる点は定義 2 においてメッセージを正しく受信できる確率が $1-\delta$ となっており、失敗した時はそれを検知できるという定義 3 が加わっている点である。

4.2.2 ASMT の歴史

ASMT は 2004 年に Srinathan ら [2] によって提案されたが、そのプロトコルには間違いがあった。その後 2007 年に Kurosawa ら [3] によって厳密に定義された。そのなかで $n = 2t + 1$ での通信効率の限界が以下のように示された。

$$|X_i| \geqq (|S| - 1)/\delta + 1 \tag{4.1}$$

X_i：channel(i) を流れる情報の集合
S：秘密情報の集合
δ：失敗確率

また、Kurosawa らは通信効率の限界に近い通信量で通信できるプロトコルも提案した。そのプロトコルの通信効率は失敗確率を ϵ とすると、

$$|X_i| = \frac{|S|-1}{\delta} + 1 > \frac{|S|-1}{\epsilon} + 1$$

$$\text{ただし } \epsilon = \left\{ \begin{pmatrix} n \\ t+1 \end{pmatrix} - \begin{pmatrix} n-t \\ t+1 \end{pmatrix} \right\} \delta$$

となっている。

4.2.3 Basic プロトコル

Kurosawa らが提案したプロトコルは計算量が指数関数的であるという問題があった。木下研究室がこれを改善して多項式時間となる Basic プロトコルを提案した。

4.2.4 Basic プロトコルの通信量

Basic プロトコルの特徴はハッシュ関数 H を用いる点であった。
敵の Second Preimage Attack が成功したときがこのプロトコルの失敗となる。敵は t 個のハッシュ値に Second Preimage Attack をするので、このプロトコルの失敗確率 ϵ は、

$$\epsilon \leqq [\text{ハッシュ関数 } H \text{ への } \mathit{SecondPreimageAttack} \text{ が成功する確率}] \times t \quad (4.2)$$

ここで H の出力値のビット数を h とすると、出力値は 2^h 通りとなり、H がランダムオラクル (ランダムに値を出力する) と仮定すると H への Second Preimage Attack が成功する確率は $1/2^h$ となる。したがって式 (4.2) は

$$\epsilon \leqq \frac{t}{2^h} \quad (4.3)$$

となる。さらに秘密 s の長さを q ビットとおくと秘密の集合 S との関係は

$$|S| = 2^q \tag{4.4}$$

通信効率の限界式 (式 (4.1)) と式 (4.3)、式 (4.4) より

$$|X_i| \geqq \frac{|S|-1}{\epsilon} + 1 \geqq \frac{2^q - 1}{t/2^h} + 1 = \frac{2^h(2^q-1)}{t} + 1 \tag{4.5}$$

となるので、Basic プロトコルの実際の $|X_i|$ が式 (4.5) の右辺に近ければ通信効率が限界に近いと言える。次にビット数で評価するために両辺の $\log_2$ をとる。

$$\begin{aligned} \log_2 |X_i| &\geqq \log_2 \left\{ \frac{2^h(2^q-1)}{t} + 1 \right\} \approx \log_2 \left\{ \frac{2^h(2^q-1)}{t} \right\} \\ &= \log_2 2^h + \log_2(2^q-1) - \log_2 t \approx h + q - \log_2 t \end{aligned} \tag{4.6}$$

一方、このプロトコルの X_i とはチャンネル ch-i を使って送信者が送る $f(i)$ と $H(f(1))$〜$H(f(n))$ である。秘密 s の長さを q ビットとしているので $f(i)$ も q ビットであり、ハッシュ値は h ビットであるので $|X_i|$ は $q+hn$ ビットとなる。よって式 (5.5) は

$$q + hn = \log_2 |X_i| \geqq h + q - \log_2 t$$

となる。両辺を比較すると左辺の hn と右辺の h との差が大きいことがわかる。

第5章

提案プロトコル

この章ではBasicプロトコルの通信効率を改善したプロトコルを提案する.

5.1　提案プロトコル

以前提案したハッシュ関数を用いたASMTプロトコルは厳密な通信効率の計算がなされておらず,Canonicalであることの証明もなされていなかった. それらを含め完成したASMTプロトコルを提案する.

5.1.1　提案プロトコルの概要

提案プロトコルではBasicプロトコルと違い,1度にm個のs_iを送ることで通信効率を改善している. 提案プロトコルの手順は以下のとおりである.
但し,Hはハッシュ値であり,安全なハッシュ関数を用いるとする.
$n = 2t + 1$(n：通信路の数,t：敵の数), 送信する秘密情報を$s = \{s_1, \cdots, s_m\}$,Pを大きな素数とする.

送信者

1. $f_1(x), \cdots, f_m(x)$をランダムで決める.($f_i(x) = s_i + a_{i_1}x + a_{i_2}x^2 + \cdots + a_{i_t}x^t$)

2. $\boldsymbol{F}_1 = f_1(1)\|f_2(1)\|f_3(1)\|\cdots\|f_m(1)$
   $\boldsymbol{F}_2 = f_1(2)\|f_2(2)\|f_3(2)\|\cdots\|f_m(2)$
   $\vdots$
   $\boldsymbol{F}_n = f_1(n)\|f_2(n)\|f_3(n)\|\cdots\|f_m(n)$ とおく.

3. ハッシュ値 $H(\boldsymbol{F}_1),\cdots,H(\boldsymbol{F}_n)$ を計算する.

4. 各チャンネル ch-i に $\boldsymbol{F}_i$,$H(\boldsymbol{F}_1),\cdots,H(\boldsymbol{F}_n)$ を送る.

敵

- $\boldsymbol{F}_1,\cdots,\boldsymbol{F}_n$ のうち,t 個しか知らない.
  $\rightarrow \boldsymbol{F}_i$ 中の $f_i(x)$ は t 次関数なので t 点からは s について何も分からない.

- ここでハッシュ関数 H は一方向性があると仮定するので $H(m)$ から m を逆算できない.
  $\rightarrow$ ハッシュ値からは s について何も分からない.

受信者

1. $\boldsymbol{F}'_1,\cdots,\boldsymbol{F}'_n$ を得る.

2. n 本の通信路のうち半分以上の $t+1$ 本は正しい情報であることを利用し, 多数決で正しい $H(\boldsymbol{F}_1),\cdots,$
   $H(\boldsymbol{F}_n)$ を得る.

3. $H(\boldsymbol{F}'_1),\cdots,H(\boldsymbol{F}'_n)$ を計算し,$H(\boldsymbol{F}_1),\cdots,H(\boldsymbol{F}_n)$ と等しいか調べる.

4. $H(\boldsymbol{F}_i) = H(\boldsymbol{F}'_i)$ となる $\boldsymbol{F}'_i$ は $t+1$ 個以上ある.
   それら全てを通る t 次関数 $f'_i(x)$ が存在するかどうかはラグランジェの補間公式から容易にわかる. 存在するなら $s' = \{f'_1(0),\cdots,f'_m(0)\}$ を出力し, 存在しなければ failure を出力する.

定理 5.1.1. 我々の提案するプロトコルは *Canonical* である.

証明. $F((\boldsymbol{F}'_{i_1}, H(\boldsymbol{F}_{i_1})), \cdots, (\boldsymbol{F}'_{i_{t+1}}, H(\boldsymbol{F}_{i_{t+1}})))$ を次のように定義する.

1. $H(\boldsymbol{F}'_{i_1}), \cdots, H(\boldsymbol{F}'_{i_{t+1}})$ を計算し,$H(\boldsymbol{F}_{i_1}), \cdots, H(\boldsymbol{F}_{i_{t+1}})$ と等しいか調べる. 等しくないものがあれば $\perp$ を出力する.
2. $t+1$ 点 $(1, \boldsymbol{F}'_{i_1}), \cdots, (t+1, \boldsymbol{F}'_{i_{t+1}})$ を通る t 次関数 g を求めて,$s' = g(0)$ を出力する.

Claim 1 : 受信者が s' を出力するならば F は $s' or \perp$ をのみ出力する.

証明. 受信者が s' を出力するならば, プロトコルの受信者の STEP4 より $H(\boldsymbol{F}_i) = H(\boldsymbol{F}'_i)$ となる全ての $\boldsymbol{F}'_i$ を通る t 次関数 $f'_i(x)$ が存在する. ゆえに任意の $\boldsymbol{F}'_{i_1}, \cdots, \boldsymbol{F}'_{i_{t+1}}$ について, もし $F((\boldsymbol{F}'_{i_1}, H(\boldsymbol{F}_{i_1})), \cdots, (\boldsymbol{F}'_{i_{t+1}}, H(\boldsymbol{F}_{i_{t+1}}))) \neq \perp$ ならば,$\boldsymbol{F}'_{i_1}, \cdots, \boldsymbol{F}'_{i_{t+1}}$ は $f'_i(x)$ 上に存在している. よって,F の STEP2 より,F は s' を出力する. したがって,F は $s' or \perp$ をのみ出力する. □

Claim 2 : 受信者が Failure を出力するならば,F は異なる s'_a と s'_b を出力する.

証明. 受信者が Failure を出力するならば, プロトコルの受信者の STEP4 より $H(\boldsymbol{F}_i) = H(\boldsymbol{F}'_i)$ となる全ての $\boldsymbol{F}'_i$ を通る t 次関数 $f'_i(x)$ が存在しない. ゆえに, ある二組の t+1 点集合 $\{\boldsymbol{F}'_{i_1}, \cdots, \boldsymbol{F}'_{i_{t+1}}\}$,$\{\boldsymbol{F}'_{j_1}, \cdots, \boldsymbol{F}'_{j_{t+1}}\}$ が存在し, それらを通る t 次関数 g_a,g_b は異なる. よって,F の STEP2 より,F は異なる $s'_a = g_a(0)$ と $g'_b = g_b(0)$ を出力する. □

式 (4.1) より, 我々の提案するプロトコルは Canonical である.

□

5.1.2 提案プロトコルの計算量

提案プロトコルでは 1 度に m 個の秘密を送るので,Basic プロトコルの約 m 倍の計算量が必要である. しかし,$m = O(n)$ であれば多項式時間のものを m 倍しても多項式時間であるので, 提案プロトコルの計算量は多項式時間である.

5.1.3 提案プロトコルの通信量

提案プロトコルは1度にm個の秘密を送ることにより,その分のハッシュ値の通信量が削減され通信効率が良くなっている. Basicプロトコルを用いてm個秘密を送る場合と比較し,評価する. まずBasicプロトコルの通信量を計算する. 秘密sの長さをqビット,m個の秘密sを送るとき,送信者が送る可能性のある秘密の集合$\mathcal{S}$とchannel(i)を流れる可能性のある情報の集合は$\mathcal{X}_{1_i}$は

$$|\mathcal{S}| = (2^q)^m, |\mathcal{X}_{1_i}| = (2^{q+hn})^m \tag{5.1}$$

次に提案プロトコルの計算量を計算する. 秘密sの長さをqビット,m個の秘密sを送るとき,送信者が送る可能性のある秘密の集合$\mathcal{S}$とchannel(i)を流れる可能性のある情報の集合は$\mathcal{X}_{2_i}$は

$$|\mathcal{S}| = 2^{qm}, |\mathcal{X}_{2_i}| = 2^{qm+hn} \tag{5.2}$$

送信者が送る可能性のある秘密の集合$\mathcal{S}$をchannel(i)を流れる可能性のある情報の集合$\mathcal{X}_i$で割ったものを通信レートとする.

定義 5.1.1. 通信レート $= \frac{|\mathcal{S}|}{|\mathcal{X}_i|}$

式(5.1),式(5.2),定義5.1.1よりBasicプロトコルと提案プロトコルの通信レートを計算し,比較すると

$$\frac{\frac{|\mathcal{S}|}{|\mathcal{X}_{1_i}|}}{\frac{|\mathcal{S}|}{|\mathcal{X}_{2_i}|}} = \frac{\frac{2^{qm}}{2^{qm+hn}}}{\frac{2^{qm}}{2^{qm+hn}}} = \frac{2^{-hn}}{(2^{-hn})^m} = 2^{hn(m-1)} \tag{5.3}$$

となる.以上より提案プロトコルはBasicプロトコルより通信効率が改善されていることがわかる.

提案プロトコルの失敗確率ϵを求め,Basicプロトコルを用いてm個秘密を送る場合と比較して評価する. ハッシュ関数への衝突攻撃試行回数をkとするとBasicプロトコルでの失敗確率ϵは式(4.3)により

$$\epsilon \leq (\frac{t}{2^h})k$$

m 個秘密を送る場合は敵は mt 個のハッシュ値に Second Preimage Attack をするので

$$\epsilon \leq (\frac{mt}{2^h})k \tag{5.4}$$

となる. 提案プロトコルの失敗確率 ϵ は t 個のハッシュ値に Second Preimage Attack をするので

$$\epsilon \leq (\frac{t}{2^h})k \tag{5.5}$$

$$(\frac{t}{2^h})k \leq \epsilon \leq (\frac{mt}{2^h})k \tag{5.6}$$

となる.以上より提案プロトコルは Basic プロトコルより失敗確率が改善されていることがわかる.

秘密 s の長さを q ビットとおくと,送信者が送る可能性のある秘密の集合 $\mathcal{S}$ と channel(i) を流れる可能性のある情報の集合は $\mathcal{X}_i$ は

$$|\mathcal{S}| = 2^{qm}, |X_i| = 2^{qm+hn} \tag{5.7}$$

となる.

定理 5.1.1 より提案プロトコルは Canonical であるから式 (4.1) より通信効率の限界を計算すると

$$\delta \geq \frac{2^{qm}-1}{2^{qm+hn}-1} \approx \frac{1}{2^{hn}} \tag{5.8}$$

式 (5.5) と式 (5.8) より

$$\frac{1}{2^{hn}} \leq \delta \leq \frac{kt}{2^h} \tag{5.9}$$

ここでは k と t が十分小さいとしても通信効率の限界と比べると $\frac{1}{2^n}$ だけ差があることがわかる. 計算量が多項式時間のままこの差を埋めることが今後の課題である.

第6章

提案プロトコルの実装

この章では提案プロトコルの実装について触れる.

6.1　提案プロトコルの実装

前章ではハッシュ関数を用いた安全な ASMT プロトコルを提案した.
本章では仮想環境で実装した過程と現在のインターネット上での問題について議論する.

6.1.1　n チャネルメッセージ伝送の実装

仮想環境での実装の話に入る前に, n チャネルメッセージ伝送の実装に必要な事柄をあげる. n チャネルメッセージ伝送方式は n 本の通信路を用いて暗号化通信方式である.つまり n 本の通信路が必要である. ここでいう n 本の通信路というのはどういったものなのかを確認していく.

1. 送信者は各チャネルに暗号化された情報を送る.

2. 敵が潜んでいる可能性があり,敵は盗聴や改善を行う.

3. 受信者は各チャネルから受け取った情報を元に復号を試みる.

nチャネルメッセージ伝送がなぜ鍵を用いることなく安全な通信を行えているか.それは複数の通信路を用いているためである.提案プロトコルであれば半分より少ない通信路に敵が潜んでいたとしても$t+1$chの情報がなければ復号することはできない.そのためここでいう通信路とは各チャネルが独立した通信路であると言える.現在のインターネットの規格では最適な経路を通るためそれぞれ異なる通信路を確保することができない.送信者と受信者の間に1つ以上の中継地点を挟むとしたとき,中継しているホストの経路制御表を適当に書き換えればプロトコルは成立する.しかし,それでは各チャネルの中継する全てのホストに対して自分が干渉できる状況でしかnチャネルメッセージ伝送は行えない.nチャネルメッセージ伝送は暗号化通信であることから中継するホストに関係なく通信できることが望ましいと考えられる.

6.1.2 実装案

前節ではnチャネルメッセージ伝送を行うにはそれぞれ異なる通信路が必要なことと、出来る限り中継地点に関係なくプロトコルが成立する必要があることを述べた.本節ではそれらを満たす手法を1つ提案する.それはソースルーティングという経路制御方式を使うことである.ソースルーティングはIPv4では中継地点のIPアドレスを送信者が最大9つまで明示的に経路を指定できる.図6.1にはソースルーティングによるIPヘッダの変化を示した.中継地点のIPアドレスがIPヘッダに記録される.

ソースルーティングを用いて中継地点を重複しないように指定することでそれぞれ異なる通信路を実現する.

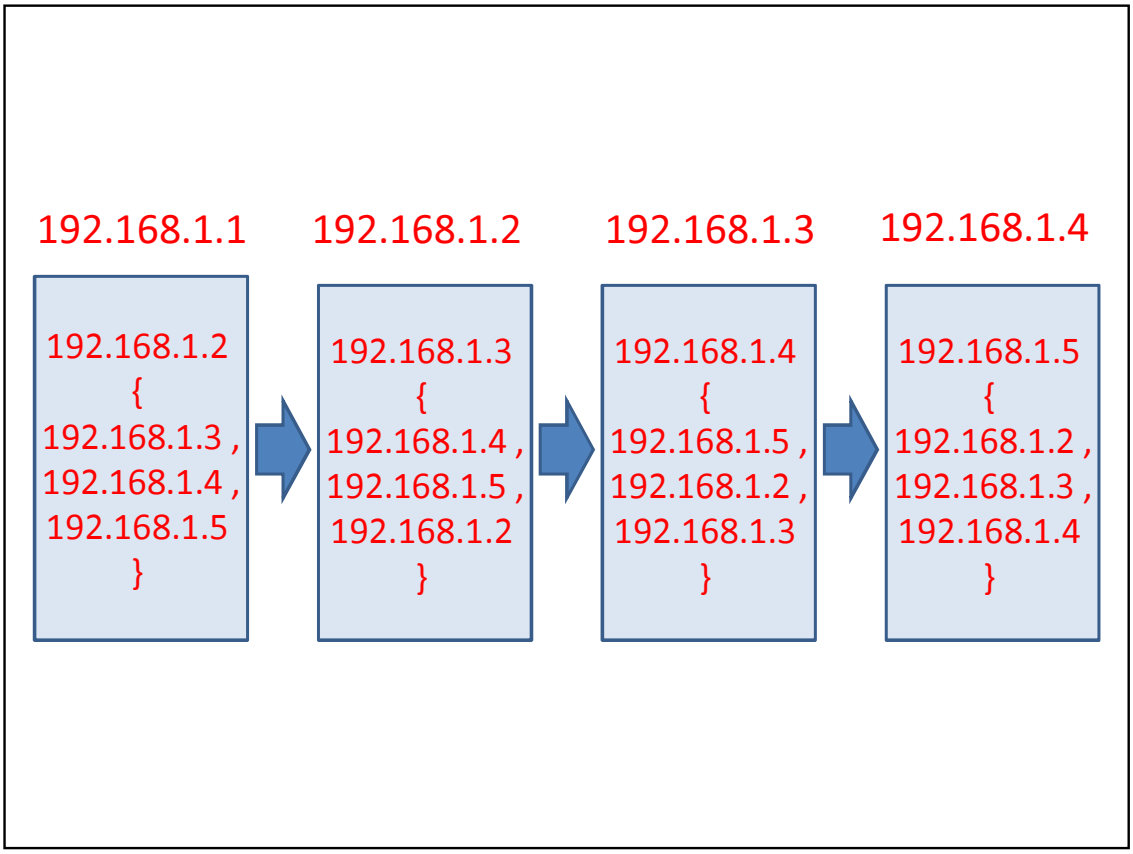

図 6.1　ソースルーティングと IP ヘッダ

6.1.3　実装実験

Oracle 社の VM VirtualBox を使い仮想環境を構築し、実験を行った. オペレーティングシステムには Ubuntu10.10 を使用した. また, ソースルーティングはデフォルトでは禁止にされているので sysctl の設定を $net.ipv4.conf.all.accept\_source\_route = 1$ に変更した. 実験装置を図 6.2 に示す. 送信プログラム概要

1. 送信者が受信者に送信したい秘密を入力する.
2. ランダム係数を決定し,$f(x)$ とハッシュ値を計算する.
3. 適切な長さの偽装データとランダムに選んだ 2 つのチャネルを入れ替え

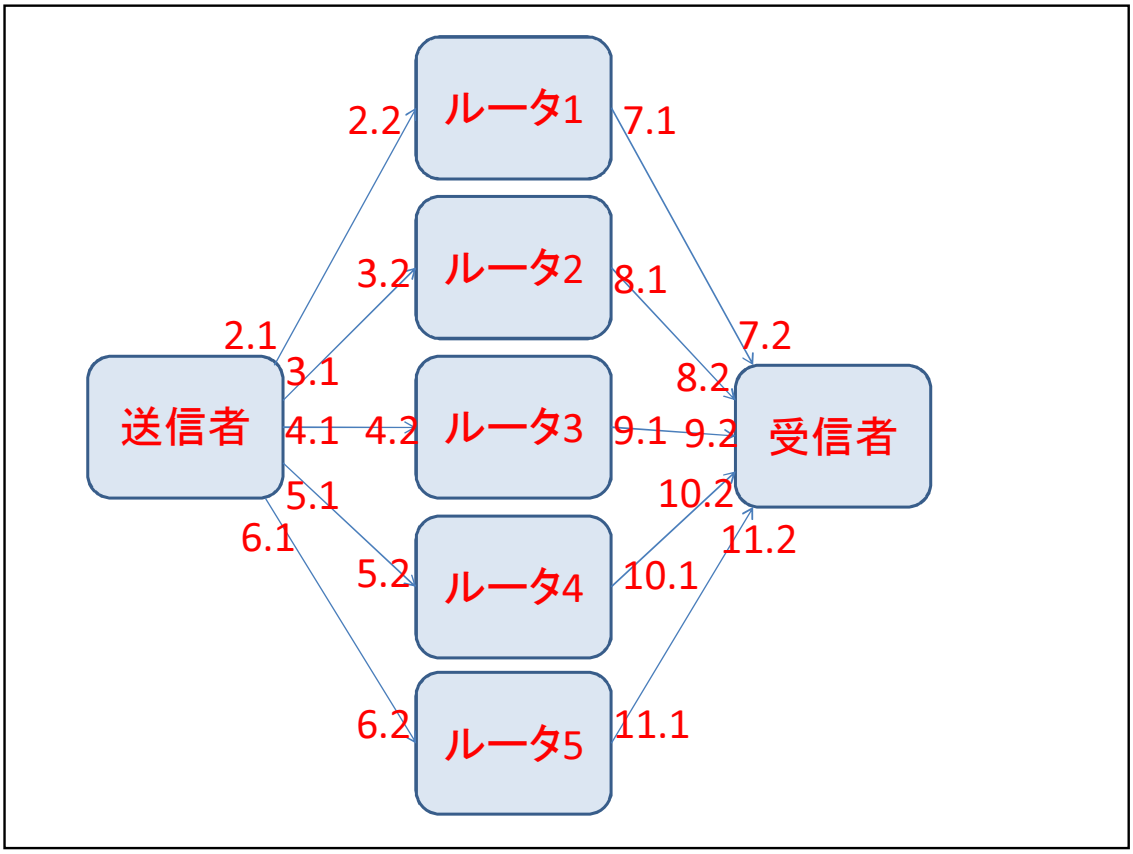

図 6.2　実験装置図

る.（改ざん想定）

4. ソケットをオープンして宛先アドレスとポート番号をセットする.

5. 構造体にソースルーティングに必要な情報をセットする.NOP, オプションコード 0x89（ストリクトソースルーティング）, オフセット 4, 中継アドレス.

6. オープンしてあるソケットのオプションに前の手順で作った構造体を挿入する.

7. それぞれのチャネルに $f(x)$ とハッシュ値を送る.

8. ソケットを閉じる.

受信プログラム概要

1. 指定されたポートから受け取る準備をする.
2. 各チャネルから $f(x)$ とハッシュ値を受け取る.
3. 多数決により正しいハッシュ値を決定する.
4. $f'(x)$ を元に計算したハッシュ値と正しいハッシュ値を比較し、等しい $f'(x)$ を選び出す.
5. 選んだ $f'(x)$ を元に復号を試みる.
6. 復号された秘密を表示する.

以上が送受信に使うプログラムである. このプログラムを用いて以下の実験を行った.
実験手順

1. それぞれのホストの出口と入口で tcpdump を行いパケットを監視する.
2. 受信者が受信プログラムを実行する.
3. 送信者が送信プログラムを実行する.
4. 受信者の受信プログラムが正しく情報を受信し、復号できていることを確認する.
5. tcpdump を行った箇所で正しくソースルーティングが行われたかを確認する.

6.1.4 実験結果

送信プログラム実行時に入力した秘密が正しく復号されたのを確認した. 各ホストで得られたtcpdumpのログの一部分以下に示す.

```
・送信者
nchannel@nchannel-v3:~ route
カーネルIP経路テーブル
受信先サイト    ゲートウェイ    ネットマスク    フラグ Metric Ref 使用数 インタフェース
192.168.6.0     *               255.255.255.0   U     1      0        0 eth7
192.168.5.0     *               255.255.255.0   U     1      0        0 eth6
192.168.4.0     *               255.255.255.0   U     1      0        0 eth5
192.168.3.0     *               255.255.255.0   U     1      0        0 eth4
192.168.2.0     *               255.255.255.0   U     1      0        0 eth3
192.168.1.0     *               255.255.255.0   U     1      0        0 eth2
link-local      *               255.255.0.0     U     1000   0        0 eth2
nchannel@nchannel-v3:~ sudo tcpdump -Xi eth3
[sudo] password for nchannel:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth3, link-type EN10MB (Ethernet), capture size 65535 bytes

01:18:57.218679 IP nchannel-v3.local.50853 > tutetuti-v2.local.12349: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 303d 000b 8e72 3839 00              ..0=...r89.
01:19:39.224121 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 8e72 3136 00              ..0B...r16.
01:19:41.224287 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 8e72 3138 32              ..0B...r182
01:19:43.225126 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 8e72 3338 00              ..0B...r38.
01:19:45.226195 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 8e72 3136 36              ..0B...r166
01:19:47.226382 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 8e72 3137 38              ..0B...r178

・中継地点（送信者側）
nchannel@tutetuti-v2:~ route
```

```
カーネルIP経路テーブル
受信先サイト    ゲートウェイ    ネットマスク    フラグ Metric Ref 使用数 インタフェース
192.168.7.0     *               255.255.255.0   U     1      0        0 eth2
192.168.2.0     *               255.255.255.0   U     1      0        0 eth1
link-local      *               255.255.0.0     U     1000   0        0 eth1
nchannel@tutetuti-v2:~ sudo tcpdump -Xi eth1
[sudo] password for nchannel:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

01:18:53.698573 IP nchannel-v3.local.50853 > tutetuti-v2.local.12349: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 303d 000b 4266 3839 0000 0000       ..0=..Bf89....
01:19:35.685185 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 4964 3136 0000 0000       ..0B..Id16....
01:19:37.684328 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1762 3138 3200 0000       ..0B...b182...
01:19:39.684513 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 4762 3338 0000 0000       ..0B..Gb38....
01:19:41.684404 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1364 3136 3600 0000       ..0B...d166...
01:19:43.683076 IP nchannel-v3.local.50853 > tutetuti-v2.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 4011 128a c0a8 0203  H..+..@.@.......
        0x0010:  c0a8 0202 0189 0b04 c0a8 0702 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1163 3137 3800 0000       ..0B...c178...

・中継地点（受信者側）
nchannel@tutetuti-v2:~ sudo tcpdump -Xi eth2
[sudo] password for nchannel:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 65535 bytes

01:18:53.700824 IP nchannel-v3.local.50853 > nchannel-v4.local.12349: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 303d 000b 4266 3839 00              ..0=..Bf89.

01:19:35.685199 IP nchannel-v3.local.50853 > nchannel-v4.local.12354: UDP, length 3
```

```
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 4964 3136 00              ..0B..Id16.
01:19:37.684339 IP nchannel-v3.local.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1762 3138 32              ..0B...b182
01:19:39.684524 IP nchannel-v3.local.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 4762 3338 00              ..0B..Gb38.
01:19:41.684415 IP nchannel-v3.local.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1364 3136 36              ..0B...d166
01:19:43.683087 IP nchannel-v3.local.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1163 3137 38              ..0B...c178
```

・受信者

```
nchannel@nchannel-v4:~ route
カーネルIP経路テーブル
受信先サイト    ゲートウェイ    ネットマスク    フラグ Metric Ref 使用数 インタフェース
192.168.7.0     *               255.255.255.0   U     1      0        0 eth2
192.168.11.0    *               255.255.255.0   U     1      0        0 eth6
192.168.10.0    *               255.255.255.0   U     1      0        0 eth5
192.168.9.0     *               255.255.255.0   U     1      0        0 eth4
192.168.8.0     *               255.255.255.0   U     1      0        0 eth3
link-local      *               255.255.0.0     U     1000   0        0 eth2
nchannel@nchannel-v4:~ sudo tcpdump -Xi eth2
[sudo] password for nchannel:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 65535 bytes

01:18:57.264338 IP 192.168.2.3.50853 > nchannel-v4.local.12349: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 303d 000b 4266 3839 0000 0000       ..0=..Bf89....

01:19:39.250003 IP 192.168.2.3.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 4964 3136 0000 0000       ..0B..Id16....
01:19:41.248992 IP 192.168.2.3.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
```

```
        0x0020:  c6a5 3042 000b 1762 3138 3200 0000        ..0B...b182...
01:19:43.248498 IP 192.168.2.3.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 4762 3338 0000 0000        ..0B..Gb38....
01:19:45.248661 IP 192.168.2.3.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1364 3136 3600 0000        ..0B...d166...
01:19:47.247981 IP 192.168.2.3.50853 > nchannel-v4.local.12354: UDP, length 3
        0x0000:  4800 002b 0000 4000 3f11 0e87 c0a8 0203  H..+..@.?.......
        0x0010:  c0a8 0702 0189 0b08 c0a8 0701 c0a8 0b02  ................
        0x0020:  c6a5 3042 000b 1163 3137 3800 0000        ..0B...c178...
```

他のチャネルでも上記と同様の結果が得られた.

6.1.5 安全性を考慮した経路

安全性を確保するためにはそれぞれ異なる経路を確保することが求められているが, 現在のインターネットではインターネットに接続するためにはプロバイダを通す必要がある. 送信者から受信者への経路は, 送信者⇒送信者契約プロバイダ⇒受信者契約プロバイダ⇒受信者　となる. プロバイダ内の中継地点は1通りのため異なる経路確保することができない. よって, 経路を考慮すべき箇所はプロバイダ間であると言える. 送信者と受信者のプロバイダの間に別のプロバイダを経由することによってそれぞれ異なる複数の経路を実現することができる. しかし, 契約しているプロバイダが送信者受信者共に同じであった場合や、規模の小さなプロバイダーでプロバイダ間の経路が複数とれない場合がある.

6.1.6 VPNを使った安全な経路の提案

本節では前節の問題を解決する方法を提案する. ここでは仮定として複数のプロバイダ上にVPN(Virtual Private Network)が利用できるホストがあるとする. 送信者は複数のプロバイダ上のホストから必要な数だけホストを選び,

選んだホストにVPNを張る. そしてその各ホストから受信者へとnチャネルメッセージ伝送を行う.そうすることで複数のそれぞれ異なる経路を実現することができる.

第7章

結論

ハッシュ関数を用いたASMTプロトコルを提案した. ASMTは2007年にKurosawaらによって厳密に定義された. そのなかで$n = 2t + 1$のときの通信効率の限界が示され, 限界に近い通信量で通信できるプロトコルが提案された. しかしそこのプロトコルは計算量が指数関数的であるという問題があった. 提案プロトコルはハッシュ関数を用いることでこれを多項式時間まで改善した. 提案プロトコルの通信効率は通信効率の限界と比べると$\frac{1}{2^n}$だけ差があることがわかった. nチャネルメッセージ伝送はまだ実装がなされていない暗号化通信方式である. これをソースルーティングを用いて提案したプロトコルを実装した. 現在のインターネット上での実装案と問題点を取り上げ, その問題の解決案を提案した.

謝辞

本研究を行なうにあたり，終始熱心に御指導していただいた木下宏揚教授、鈴木一弘先生に心から感謝致します。良き研究生活を送らせていただいた木下研究室の方々に深く感謝致します.

参考文献

[1] DANNY DOLEV、CYNTHIA DWORK、ORLI WAARTS、MOTI YUNG ”Perfectly Secure Message Transmission”
Journal of the Association for Computing Machinery、Vol.40,No.1、pp.17-47(1993)

[2] K. Srinathan、Arvind Narayanan、C. Pandu Rangan ”Optimal Perfectly Secure Message Transmission”
CRYPTO 2004、LNCS 3152、pp.545-561(2004)

[3] Kaoru KUROSAWA、Kazuhiro SUZUKI、Members ”Almost Secure (1-Round,n-Channel) Message Transmission Scheme”
IEICE TRANS. FUNDAMENTALS、VOL.E92-A,NO.1(2009)

[4] 笠原正雄、佐竹賢治：”誤り訂正符号と暗号の基礎数理”、コロナ社 (2004)

[5] 澤田秀樹：”暗号理論と代数学”、海文堂 (1997)

[6] ”整数の合同” http://www2.cc.niigata-u.ac.jp/ takeuchi/tbasic/BackGround/-Cong.html

[7] 三谷政昭、佐藤伸一、ひのきいでろう：”マンガでわかる暗号”、オーム社 (2007)

[8] 黒澤馨、尾形わかは：”現代暗号の基礎数理”、コロナ社 (2004)

[9] 今井秀樹：”情報・符号・暗号の理論”、コロナ社 (2004)

[10] Douglas R. Stinson、櫻井幸一：”暗号理論の基礎”、共立出版株式会社 (1996)

[11] DOUGLAS R. STINSON：”CRYPTOGRAPHY THEORY AND PRACTICE THIRD EDITION”、Chapman & Hall/CRC(2006)

[12] HASAN MD. SAYEED、HOSAME ABU-AMARA ”Efficient Perfectly Secure Message Transmission in Synchronous Networks”
INFORMATION AND COMPUTATION 126、pp.53-61(1996)、ARTICLE NO.0033

[13] Saurabh Agarwal、Ronald Cramer、Robbert de Haan ”Asymptotically Optimal Two-Round Perfectly Secure Message Transmission”
CRYPTO 2006、LNCS 4117、pp.394-408(2006)

[14] Kaoru Kurosawa、Kazuhiro Suzuki ”Truly Efficient 2-Round Perfectly Secure Message Transmission Scheme”
Advances in Cryptology、EUROCRYPT 2008 LNCS 4965、pp.324-340(2008)

質疑応答