Vol.2017-DPS-171 No.29
Vol.2017-MBL-83 No.29
Vol.2017-ITS-69 No.29
2017/6/2

偽平文混入によるより安全な暗号通信手法

林 正義[1,a)]　桧垣 博章[1,b)]

概要：盗聴者が取得した暗号文から平文を得る最も基本的な手法は, 可能な復号鍵を総当たりで適用して復号手順を実施するものである. このとき盗聴者が, 適用した可能な復号鍵が正しい復号鍵であり, 得られた復号文が平文であると判断する根拠は, この復号文が文脈的に妥当であることである. そこで, 本論文では, あらかじめ用意した文脈的に尤もらしい偽平文を暗号文に混入し, 可能な復号鍵のひとつである偽復号鍵を適用して復号手順を実施した場合には, 偽平文が得られる暗号通信手法を提案する. これによって, 盗聴者が最初に文脈的に妥当な復号文を得ることで復号手順を終えるのであれば, 偽平文の復号によって復号手順を終え, 平文を取得する機会を逸することとなる. また, 盗聴者がすべての文脈的に妥当な復号文を得るまで復号手順を継続するのであれば, 得られた復号文から平文を選択する手法が存在しなければ, 偽平文を平文であるとして取得し, 平文を平文として取得できないことが考えられる. 本提案手法の簡易な実現方法を示し, 偽平文を混入する提案手法が従来手法と比較して, 復号に要する時間を延長し, 平文を平文として取得する確率を低減することを示す.

キーワード：暗号通信方式, 共通鍵暗号, 偽平文.

Cryptographic Communication with Pseudo Plaintext

Masayoshi Hayashi[1,a)]　Hiroaki Higaki[1,b)]

Abstract: One of the fundamental methods for eavesdroppers to achieve a plaintext from a cryptogram is the round robin attack where available decryption keys are exhaustively applied to the decryption procedure. Here the reason why the eavesdroppers find the correct decryption key and the achieved decrypted text is the plaintext is that the decrypted text is contextually valid. This paper proposes a novel cryptgraphy method where pseudo plaintexts which seem to be contextually valid are mixed into a cryptogram with the plaintext. If a eavesdropper applies a pseudo decryption key to the decryption procedure, the contextually valid pseudo plaintext is achieved that the eavesdropper cannot determine whether it is a plaintext or not. This paper also shows concrete encryption/decryption procedures and an implementation by using Python. In addition, we show that our proposed method with mixture of pseudo plaintexts requires eavesdroppers for longer decryption time and the probability for them to achieve the plaintext extremely decreases.

Keywords: Cryptography, Common Key Cryptosystem, Pseudo Plaintext.

1. はじめに

送信元コンピュータから送信先コンピュータへ配送されるデータを盗聴者に取得されることを防止する安全な通信手法への要求がますます高まっている. ここでは, 送信元コンピュータでもとのデータである平文を暗号文に変換 (暗号化) して送信元コンピュータから送信先コンピュータまで暗号文を配送し, 送信先コンピュータで暗号文を平文へと変換 (復号) する暗号通信手法が用いられる [3]. 暗号通信手法では, 盗聴した暗号文の復号に用いる数値である復号鍵の候補全体を総当たりするのに要する計算が膨大であること, および, 多数の暗号文を盗聴しても統計的に復号鍵を推定することが困難であることを安全性の根拠としている. しかし, コンピュータの性能向上にともない, 総当たり

[1] 東京電機大学大学院ロボット・メカトロニクス学専攻
Department of Robotics and Mechatronics, Tokyo Denki University, Adachi Tokyo 120–8551, Japan
[a)] hayashi@higlab.net
[b)] hig@higlab.net

Vol.2017-DPS-171 No.29
Vol.2017-MBL-83 No.29
Vol.2017-ITS-69 No.29
2017/6/2

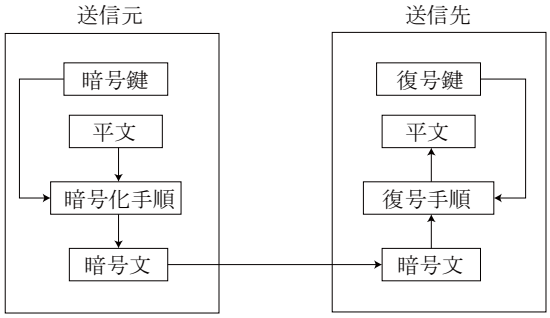

図 1 暗号通信手法

に要する計算時間を削減することが可能となり, 平文を入手される可能性が高まっている. そこで, 本論文では, あらかじめ用意した偽平文を暗号文に混入し, 盗聴者が誤った復号鍵を用いて復号する場合には偽平文が取得されるようにすることで, 平文を取得することを困難にする暗号通信手法を提案する. 本手法を用いると, 盗聴者は平文と偽平文を判別できないことから, 盗聴者に平文を平文として取得される可能性を低減することができる.

2. 関連研究

暗号通信手法では, 送信元コンピュータで平文を暗号文へと暗号化し, 送信元コンピュータから送信先コンピュータまで暗号文を配送し, 送信先コンピュータで暗号文を平文へと復号する. ここでは, 暗号文から平文を得ることが困難となるように暗号化と復号の対である暗号アルゴリズムを構成し, たとえ暗号文を盗聴されても容易には平文を入手することができない, という安全性を提供する. オープンなネットワークであるインターネットに接続する多種多様なコンピュータでは, 暗号アルゴリズムはソフトウェアで実装される. そのため, 平文から暗号文へ変換する暗号化手順と暗号文から平文へ変換する復号手順は盗聴者に対してもプログラムとして事実上公開されている. そこで, 各手順への入力に暗号鍵, 復号鍵という数値を用い, 復号に必要な鍵を盗聴者に対して秘匿する (図 1).

盗聴者は, 盗聴により取得した暗号文に対して, 可能な復号鍵を用いて復号手順を実施して平文の取得を試みる (図 2). このとき, 可能な復号鍵の数が膨大であるために, 総当たりによって平文を取得することが困難となっている. ただし, 多数の暗号文を盗聴し, 統計的な手法を適用することによって可能な復号鍵を十分に削減できる場合には, 高性能コンピュータを安価に利用可能な環境においては, 平文の取得や復号鍵の推定が可能となることが考えられる. そこで, 現在広く用いられている暗号アルゴリズムでは, 暗号文に平文の統計的性質が反映されないようにすることで復号鍵の推定を困難にしている.

ここで, 盗聴者が復号鍵の総当たりによって平文の取得を試みる場合, 復号に用いた復号可能な鍵が正しい復号鍵であり, 復号手順の出力によって得られる復号文が平文で

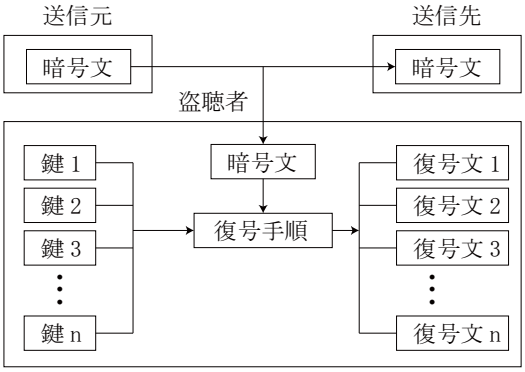

図 2 総当たり攻撃

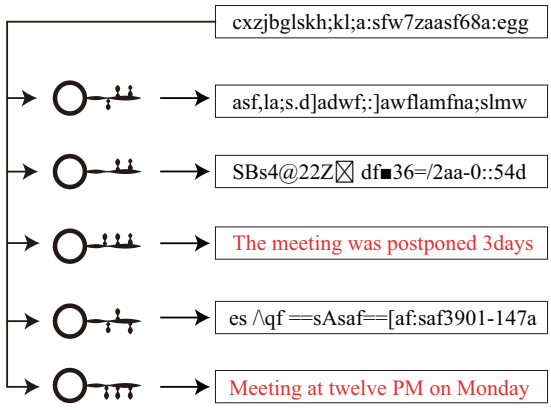

図 3 複数の可読な出力が得られる復号

あることを判断する材料は, その出力が文脈上妥当なものであるか否かである. 本論文では, 文脈上妥当であることを「可読である」ということとする. つまり, ある鍵を用いたときの復号手順の出力が可読でない場合には, その鍵が正しい復号鍵ではないと推定する. 一方, ある鍵を用いたときの復号手順の出力が可読である場合でも, その鍵は正しい復号鍵であるとは限らず, この出力が平文であるかどうかを決定することはできない (図 3).

論文 [2] では, 復号鍵の総当たりによる盗聴手法において複数の可読な平文を取得させる使い捨てパッドが提案されている. 暗号化手順では, 平文 PT と同じ長さの共通鍵 K を用い, PT と K とのビットごとの排他的論理和をとることで暗号文 ET を得る. また, 復号手順では, ET と K とのビットごとの排他的論理和をとることで平文 PT を得る. ここで, 可能な共通鍵の数は $2^{|K|}$ ($|K|$ は K の長さ) であり, このすべての鍵を用いて復号手順を実施すると長さ $|PT|$ のすべての可能な出力が得られ, 多数の可読な出力が得られることが期待できる. したがって, 盗聴者は, 複数の可読な出力から平文を特定することは困難であり, 可能な鍵の一部のみを用いて復号手順の実施を終えた場合には, 平文を出力のひとつとしても得ないことが考えられる. ただし, 本手法では, 平文の統計的な特性を暗号文がそのまま引き継ぐため, 多数の暗号文を盗聴することによって共通鍵を推定される問題がある.

Vol.2017-DPS-171 No.29
Vol.2017-MBL-83 No.29
Vol.2017-ITS-69 No.29
2017/6/2

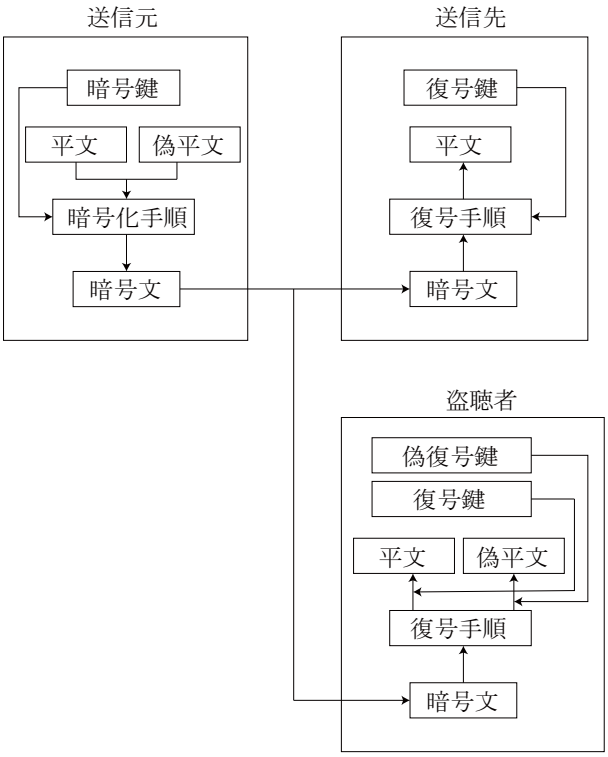

図 4 偽平文を混入する暗号通信手法

なお, 暗号通信手法には, 暗号化と復号に同一の鍵である共通鍵を用いる共通鍵 (秘密鍵) 暗号方式と, 暗号化と復号に異なる鍵を用い, 暗号化に用いる暗号鍵を公開し, 復号に用いる復号鍵のみを秘匿する公開鍵暗号方式とがある. 本論文では, 共通鍵暗号方式を対象とする.

3. 提案手法

3.1 偽攻撃ノード検出通知による攻撃

本論文では, 送信元コンピュータにおいて, 用意した平文と偽平文, および暗号鍵を入力として暗号文を出力する暗号化手順と, 送信先コンピュータにおいて, 暗号文と正しい復号鍵を入力とすると平文を出力し, 盗聴者において暗号文と偽復号鍵を入力とすると偽平文を出力する復号手順とからなる暗号アルゴリズムを提案する (図 4). これによって, 盗聴者が可能な暗号鍵を総当たりで入力として復号手順を実施すると, 出力として得られた可読な復号文が偽平文であることがある. 偽復号鍵を復号手順の入力とすることによって偽平文を復号手順の出力として取得することで, 盗聴者が平文を取得したと誤認し, 正しい復号鍵を用いて平文を取得する機会を逸することや, 正しい復号鍵を復号手順の入力とすることでその出力として取得した平文との間でいずれが平文であるかを決定することができないことから, 平文を平文として取得する可能性を低減することができる (図 5).

[偽平文を混入する暗号アルゴリズム]

平文を PT, 偽平文を PPT, 暗号鍵を K_e, 復号鍵を K_d とするとき, 暗号文を ET とする以下を満足する暗号化手順

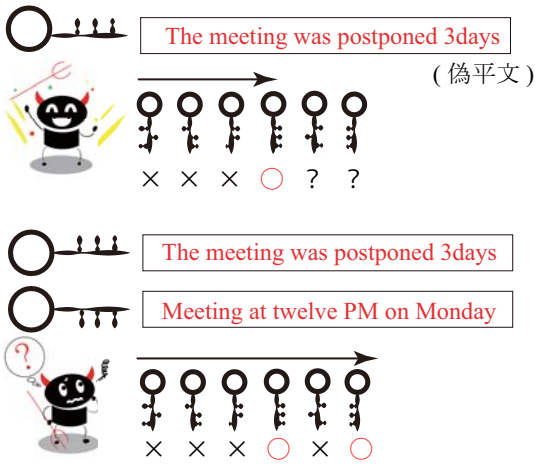

図 5 偽平文混入の効果

$\mathcal{E}$ と復号手順 $\mathcal{D}$ の対を偽平文を混入する暗号アルゴリズムという. ただし, 偽暗号鍵を PK_e, 偽復号鍵を PK_d とする.

$ET = \mathcal{E}(PT, PPT, K_e)$

$PT = \mathcal{D}(ET, K_d)$ where $\exists PK_d PPT = \mathcal{D}(ET, PK_d)$

□

従来の秘密鍵暗号方式と同様に, 送信元コンピュータと送信先コンピュータが暗号鍵 K_e と正しい復号鍵 K_d を保持することが前提となる. 偽暗号鍵 PK_e は暗号化手順 $\mathcal{E}$ で生成されるものの $\mathcal{E}$ の外で用いられることはなく, ネットワークを配送されることもない. 偽復号鍵 PK_d は, 存在することのみが求められるため, PK_e と同様にネットワークを配送されることはない. これを実現する具体的な手法として, 本論文では, 平文を暗号鍵で暗号化したものと偽平文を偽共通鍵で暗号化したものとを連結する手法を提案する. ここでは, 平文を共通鍵で暗号化したものが前後いずれかにあるかを特定できないようにするために, 連結の順序を盗聴者が取得できない共通鍵によって定める. また, 偽平文を暗号化する偽共通鍵は暗号化手順の中で生成するが, 偽共通鍵で偽平文を暗号化した場合にも同一の連結順序となるようにすることで, 盗聴者が総当たりで復号を試みる際に偽共通鍵を用いて復号手順を実施した場合には偽平文が出力されることとしている.

平文を PT, 偽平文を PPT, 共通鍵を K とするとき, 既存の共通鍵暗号アルゴリズムの暗号化関数 $\mathcal{E}'$ と復号関数 $\mathcal{D}'$ をもとにした偽平文を混入する暗号アルゴリズムは以下の手順で与えられる.

[暗号化手順 $\mathcal{E}$]

(1) K に対するパイナリパリティ $parity(K)$ を得る.

(2) PT に対する K を用いた暗号文 $\mathcal{E}'(PT, K)$ を得る.

(3) パイナリパリティの値が $\overline{parity(K)}$ である偽共通鍵 PK, すなわち, $parity(PK) = \overline{parity(K)}$ を満足する PK を生成する.

(4) PPT に対する PK を用いた暗号文 $\mathcal{E}'(PPT, PK)$ を得る.

Vol.2017-DPS-171 No.29
Vol.2017-MBL-83 No.29
Vol.2017-ITS-69 No.29
2017/6/2

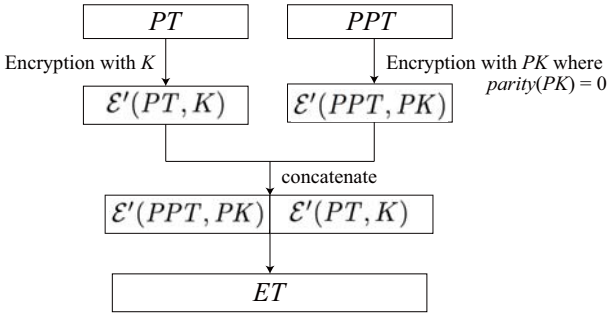

図 6 $parity(K) = 0$ のときの暗号化手順

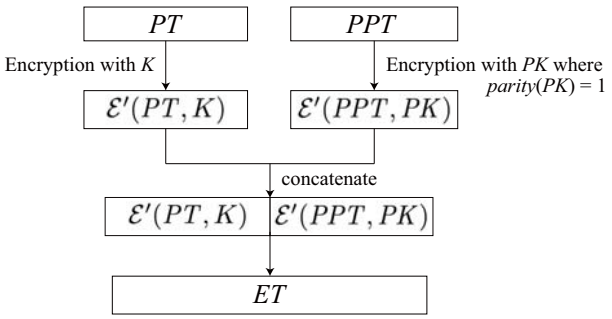

図 7 $parity(K) = 1$ のときの暗号化手順

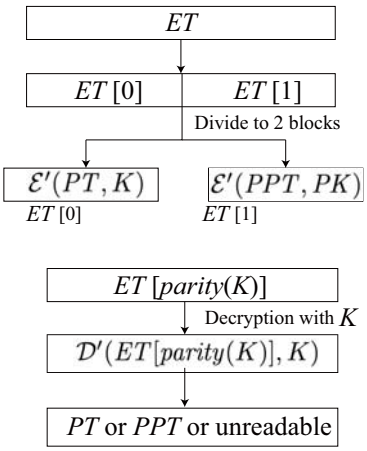

図 8 復号手順

(5) $\mathcal{E}'(PT, K)$ と $\mathcal{E}'(PPT, PK)$ とをそれぞれの共通鍵 K, PK に対するバイナリパリティの値 $parity(K)$, $parity(PK)$ が 0 となるもの, 1 となるものの順に連結して暗号文 ET を得る. すなわち, $+$ を連結とするとき, $parity(K) = 0$ であるならば $ET = \mathcal{E}'(PT, K) + \mathcal{E}'(PPT, PK)$, $parity(K) = 1$ であるならば $ET = \mathcal{E}'(PPT, PK) + \mathcal{E}'(PT, K)$ とする. □

[**復号手順** $\mathcal{D}$]
(1) K に対するバイナリパリティ $parity(K)$ を得る.
(2) ET をサイズが同じ $ET[0]$ と $ET[1]$ に分割する.
(3) $PT = \mathcal{D}'(ET[parity(K)], K)$ により PT を得る.
□

この暗号通信手法は以下の性質を持つ.

[**定理**] 偽共通鍵 PK を用いて復号手順を行なうと偽平文 PPT が得られる. □

すなわち, 復号手順 $\mathcal{D}$ において,
(1) PK に対するバイナリパリティ $parity(PK)$ を得る.
(2) ET をサイズが同じ $ET[0]$ と $ET[1]$ に分割する.
(3) $PPT = \mathcal{D}'(ET[parity(PK)], PK)$ により PPT を得る.

本提案手法のプログラムを Python で実装した. ソースコードを付録のリスト A.1 に示す. 本プログラムでは, 既存の共通鍵暗号アルゴリズムとして AES[1] を選択し, 平文と鍵はそれぞれ 32Byte で実装した.

なお, 混入する偽平文は 1 つには限定されない. 同様の方法によって, 送信元コンピュータでは, 平文および m 個の偽平文を共通鍵および m 個の異なる偽共通鍵によって暗号化したものを連結することで暗号文を生成し, 送信先コンピュータでは, 共通鍵によって平文を暗号化したものを復号することで平文を得る. このとき, 総当たりで可能な共通鍵を用いて復号手順を実施する盗聴者は, 偽共通鍵によって復号を試みる場合には, その偽共通鍵によって偽平文を暗号化したものを復号し, 偽平文を得ることとなる. ここでは, 可能な共通鍵を入力として 0 または 1 を返す $parity()$ に代えて, 0, ..., m のいずれかを返す $location()$ を用いる.

ここでは, 平文を PT, m 個の偽平文を PPT_i, 共通鍵を K とする. このとき, 既存の共通鍵暗号アルゴリズムの暗号化関数 $\mathcal{E}'$ と復号関数 $\mathcal{D}'$ をもとにした偽平文を混入する拡張された暗号アルゴリズムは以下の手順で与えられる.

[**拡張された暗号化手順**]
(1) K に対する連結位置 $location(K)$ を得る.
(2) PT に対する K を用いた暗号文 $\mathcal{E}'(PT, K)$ を得る.
(3) 連結位置の値が 0 から m 個までの整数のうち, $location(K)$ 以外のそれぞれとなる m 個の偽共通鍵 $PK_1, \ldots, PK_m$ を生成する.
(4) PPT_i に対する PK_i を用いた暗号文 $\mathcal{E}'(PPT_i, PK_i)$ を得る.
(5) $\mathcal{E}'(PT, K)$ と $\mathcal{E}'(PPT_i, PK_i)$ とをそれぞれの共通鍵 K, 偽共通鍵 PK_i に対する連結位置の値 $location(K)$, $location(PK_i)$ の順に連結して暗号文 ET を得る. □

[**拡張された復号手順**]
(1) K に対する連結位置 $location(K)$ を得る.
(2) ET をサイズが同じ $m+1$ 個の部分 $ET[0], \ldots, ET[m]$ に分割する.
(3) $PT = \mathcal{D}'(ET[location(K)], K)$ により PT を得る.

4. 評価

本論文で提案した偽平文を混入する手法の有効性を評価する.

まず, 従来手法, すなわち, 偽平文が混入されず, 可能な共通鍵の総当たりによって可読な復号文が得られた場合には, それが平文である場合について評価する. 可能な共通鍵の総数を k とすると, n 番目の可能な共通鍵で平文が得

Vol.2017-DPS-171 No.29
Vol.2017-MBL-83 No.29
Vol.2017-ITS-69 No.29
2017/6/2

られる, すなわち n 番目の可能な共通鍵が正しい共通鍵である確率は $1/k$ である. したがって, n 番目の可能な共通鍵を用いた復号手順の実施によって共通鍵の総当たりを終了する確率は $1/k$ であり, n 番目の可能な共通鍵を用いた復号手順の実施までに平文が得られる確率は n/k である.

一方, 提案手法, すなわち, m 個の偽平文が混入されており, 可能な共通鍵の総当たりによって可読な復号文が得られても, それが平文であるかどうか決定することができない場合について評価する. ここでは, n 番目の可能な共通鍵を用いた復号手順の実施によって, $m+1$ 個の可読な復号文のすべてを取得する確率を求める.

(1) k 個の可能な共通鍵から順に n 個選択する場合の数は ${}_kP_n$ 通りである.

(2) n 番目の可能な共通鍵は, 可読な復号文が得られる共通鍵, すなわち, 正しい共通鍵か偽共通鍵のいずれかである. したがって, 1 番目から $n-1$ 番目の可能な共通鍵のうち m 個が可読な復号文が得られる共通鍵, すなわち, 正しい共通鍵か偽共通鍵のいずれかである. この $n-1$ 個の可読な共通鍵が選択された回数の組合せは ${}_{n-1}C_m$ 通りである.

(3) 可読な復号文が得られる $m+1$ 個の共通鍵の並べ方は $(m+1)!$ 通りである.

(4) 可読でない復号文が得られる $k-(m+1)$ 個の可能な共通鍵から $n-(m+1)$ 個を選んで順に並べる場合の数は ${}_{k-(m+1)}P_{n-(m+1)}$ 通りである.

(1) から (4) により, n 番目の可能な共通鍵を用いた復号手順の実施によって, $m+1$ 個の可読な復号文のすべてを取得する確率は次式で与えられる.

$$\frac{{}_{n-1}C_m \times {}_{k-m-1}P_{n-m-1}}{{}_kP_n} \times (m+1)! \qquad (1)$$

k=1024 のとき, これらをグラフに表したものを図 9 に示す. 提案手法の適用によって, 復号手順の実施によってすべての可読な復号文を得るまでの時間を延長することができていることが分かる.

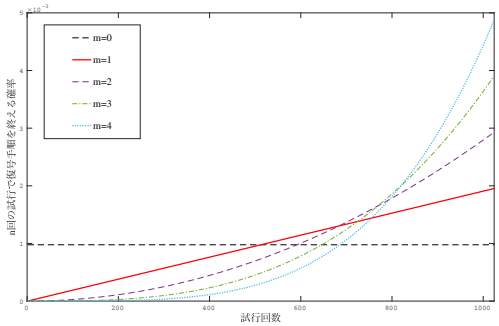

図 9 n 回の試行で復号手順を終える確率

また, 提案手法では, 盗聴者は取得した $m+1$ 個の可読な復号文のひとつを平文であるとして選択しなければならない. このとき, 正しく平文を選択することができる確率は k, n によらず $1/(m+1)$ であるから, n 番目の可能な共通鍵を用いた復号手順の実施までに盗聴者が平文を平文として取得する確率は次式で与えられる.

$$\frac{1}{m+1}\sum_{n=0}^{k}\frac{{}_{n-1}C_m \times {}_{k-m-1}P_{n-m-1}}{{}_kP_n} \times (m+1)! = \sum_{n=0}^{k}\frac{(n-1)!(n-m-1)!}{(n-m-1)!k!} \qquad (2)$$

k=1024 のとき, これらをグラフに表したものを図 10 に示す. 提案手法の適用によって, 復号手順の実施によって平文を平文として盗聴者が取得する確率を従来手法と比較して大幅に低減していることが分かる.

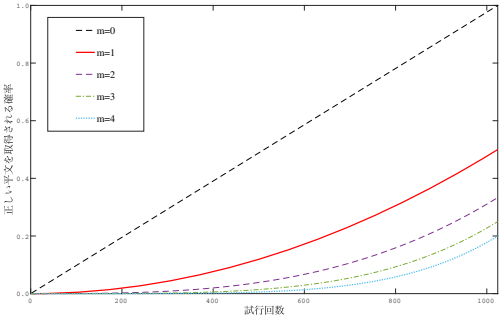

図 10 正しい平文を取得される確率

このように, 偽平文の数 m を盗聴者が知っていることを前提として, 盗聴者がすべての可読な復号文を取得してから平文とする復号文を選択する手法を適用する場合について考えると, 偽平文を混入する提案手法は従来手法と比較して, 復号に要する時間を延長し, 平文を取得する確率を低減していることが分かる. なお, 提案手法では, 既存の共通鍵暗号アルゴリズムの暗号化手順, 復号手順を適用していることから, 盗聴した複数の暗号文から統計的な偏りによって共通鍵を推定することは困難である. ただし, 従来の共通鍵暗号アルゴリズムと比較して, m 個の偽平文を混入する場合, 暗号化に $m+1$ 倍の計算時間を要すること, 暗号文の長さが $m+1$ 倍になるという問題点がある.

5. まとめ

本論文では, 秘密鍵暗号方式を対象として, 暗号文にあらかじめ用意した偽平文を混入し, 正しい復号鍵を用いて復号手順を実施した場合には平文が, 可能な復号鍵のひとつである偽復号鍵を用いて復号手順を実施した場合には偽平文が取得されることで, 盗聴者が復号に要する時間を延長し, 平文を平文として取得する確率を低減する手法を提案した. ここでは, 平文を正しい暗号鍵で暗号化したものと

偽平文を偽暗号鍵で暗号化したものとを連結して暗号文を構成するが, 連結の順序を正しい暗号鍵を用いて定めることから, 盗聴者は平文と偽平文を判別することができない. 本論文では, 盗聴者がすべての平文, 偽平文を取得してから平文として取得する復号文を選択する場合における提案手法の有効性を評価した. 今後は, 盗聴者が最初に取得した文脈的に妥当な復号文を平文として取得する場合における提案手法の有効性を評価する. また, 公開鍵暗号方式における実現方法を検討する.

参考文献

[1] Daemen, J., RijnMen, V., “The Rijndael Block Cipher,” AES proposal, First AES Candidate Conference (AES1), (1998).
[2] Shannon, C.E., “Communication Theory of Secrecy Systems,” *IEEE Press*, pp. 84–143 (1949).
[3] 林, “暗号理論入門,” シュプリンガー・ジャパン (2002).

付　　録

A.1 プログラムのソースコード

```
# −∗− coding: utf−8 −∗−
from Crypto.Cipher import AES
import Crypto.Random.random as rand
import binascii
import sys

#
#平文
message1 = 'This is true message not pseudo.'

#偽平文
message2 = 'Sorry, this is a pseudo message.'


#
#パリティチェック
def parityOf(int_type):
    parity = 0
    while (int_type):
        parity = ~parity
        int_type = int_type & (int_type - 1)
    return(parity)

#
#平文に対応する鍵を生成
#鍵は 256bit
secret_key1 = (rand.getrandbits(256))
if parityOf(secret_key1) == 0:
    parameter_key1 = 0
else:
    parameter_key1 = 1

#偽平文に対応する鍵を生成
#鍵は 256bit
secret_key2 = (rand.getrandbits(256))
if parityOf(secret_key2) == 0:
    parameter_key2 = 0
else:
    parameter_key2 = 1

while parameter_key2 == parameter_key1:
    secret_key2 = (rand.getrandbits(256))
    if parityOf(secret_key2) == 0:
        parameter_key2 = 0
    else:
        parameter_key2 = 1

#鍵の表示
print("True Key is "+str(secret_key1))
print("Pseudo Key is "+str(secret_key2))

#鍵の型変換
secret_key1 = binascii.unhexlify('%x' % int(bin(
    secret_key1), 2))
secret_key2 = binascii.unhexlify('%x' % int(bin(
    secret_key2), 2))

#

#AES 方式で平文を暗号化
crypto1 = AES.new(secret_key1)
cipher_data1 = crypto1.encrypt(message1)


#AES 方式で偽平文を暗号化
crypto2 = AES.new(secret_key2)
cipher_data2 = crypto2.encrypt(message2)


#暗号文の結合
if parameter_key1 == 0:
    last_cipher = cipher_data1+cipher_data2
else:
    last_cipher = cipher_data2+cipher_data1


#最終的な暗号文の表示
print("Cipher Text is "+repr(last_cipher))

#
#暗号文を 2分割
subcipher1 = last_cipher[0:len(last_cipher)/2]
subcipher2 = last_cipher[len(last_cipher)/2:len(
    last_cipher)]

#鍵の入力
print(u'Enter Key')

key = sys.stdin.readline()
key = long(key)

#鍵のパリティをチェック
```

Vol.2017-DPS-171 No.29
Vol.2017-MBL-83 No.29
Vol.2017-ITS-69 No.29
2017/6/2

```
90  if parityOf(key) == 0:
91      parameter_key = 0
92  else:
93      parameter_key = 1
94
95  #鍵の型変換
96  key = binascii.unhexlify('%x' % int(bin(key), 2))
97
98  #鍵のパリティによる分岐復号
99  if parameter_key == 0:
100     DEC = AES.new(key)
101     original_message1 = DEC.decrypt(subcipher1)
102 else:
103     DEC = AES.new(key)
104     original_message1 = DEC.decrypt(subcipher2)
105
106 #復号文の表示
107 print(original_message1)
```