7G-02

ラビン暗号に対するCoppersmithのショートパッドアタックの高速化

深瀬道晴†
東北学院大学†

1.はじめに

ラビン暗号はRSA暗号と同様に、冪剰余計算を行う公開鍵暗号システムである。具体的に、

$$C = (M||r)^{\delta} \bmod \mathrm{N}$$

という計算が行われる。ここで、Cは暗号文、Mは平文、$N = pq$は公開鍵の合成数であり、pとqは素数の組である。また、$||$は連結演算子、rはパディングのための乱数である。Nについては、実用上は1,024、または、2,048ビットのサイズが使用される。δについては、ラビン暗号では$\delta = 2$が使用され、RSA暗号では$\gcd(\delta, \varphi(N)) = 1$を満たす任意の$\delta$が使用される。ここで、$\varphi(N) = (p-1)(q-1)$はオイラー関数である。

RSA暗号は、ラビン暗号に対して大きなδが使用されている。初期のRSAでは、計算効率の観点から、できるだけ小さい公開鍵指数として$\delta = 3$の使用が支持されたが、Coppersmithはそのような小さい指数はある条件下で安全でないことを示した。具体的に、ある$\varepsilon \geq 0$に対して$X = N^{\frac{1}{\delta^2}-\varepsilon}$であるとき、乱数パディング$r$が$0 \leq r \leq X$を満たすとき安全でない。Coppersmithは、格子基底簡約アルゴリズムを使用するショートパッドアタックによって、秘密情報のMが2つの暗号文から復元できることを示した。ショートパッドアタックでは、与えられた合成数を法とする単変数、または、二変数多項式の秘密情報に相当する小さな根が求められるが、十分に小さな係数の多項式を求めるためにLLLアルゴリズム[1]が使用される。Coppersmithのショートパッドアタックが示されたことによって、RSAの暗号化のためには$\delta = 65{,}537$が広く使用されるようになった。$\delta = 65{,}537$に対しては、ショートパッドアタックは適用できないことが知られている。

一方、ラビン暗号で短い乱数パディングを使用する場合は、$\delta = 2$、または、$\delta = 4$が使用されるため、ショートパッドアタックを適用することができる。ラビン暗号はRSA暗号や楕円曲線暗号のように広く普及してはいないが、RFIDタグのように計算資源が限定された状況での使用が研究されている。RFIDタグに対しては、楕円曲線暗号の実装も研究されているが[3]、ラビン暗号の方がハードウェアの複雑さと処理速度の両方において効率が高い。RFIDタグなどにラビン暗号を応用する場合は、ショートパッドアタックに対して安全なパディングのサイズを見積もる必要があるが、ラビン暗号における最適なパディングサイズに関する研究は十分に行われていない。ラビン暗号の最適なパディングサイズを見積もる上では、ラビン暗号に対する現状で最も効率的な攻撃法を追求する必要がある。ラビン暗号の最適なパディングサイズを見積もった結果は、[2]において示されている。

本稿では,ショートパッドアタックとは異なる格子に対してLLLアルゴリズムを適用したシミュレーションの結果[2]をデータを示しながら解説する。[2]で示された格子の次元はCoppersmithのショートパッドアタックで使用される格子の次元の半分であり、格子基底簡約の高速化が可能である。

2.オリジナルの格子と低次元格子に対するシミュレーション結果

ここでは、オリジナルのショートパッドアタックに使用される格子と[2]で導入された低次元格子のそれぞれについて、LLLアルゴリズムを適用して計測された実行時間の比較結果を示す。シミュレーションは、それぞれの格子について、$n = 1024,1536,2048$の場合で実行時間を計測した。ここで、nはNのビット長である。nについて、セキュリティパラメータmを$m = 2,4,6,\ldots,30$の範囲で動かしながら、各mについて100の異なる格子を用意して、実行時間の平均を取った。シミュレーションには、2.7 GHz Intel Core i5のプロセッサ、8 GB 1333 MHz DDR3 RAM、macOS High Sierra 10.13.5の計算機を使用した。また、シミュレーションでは、LLLアルゴリズム

The improvement of the Coppersmith's shortpad attack against Rabin cryptosystems
† Masaharu Fukase, Tohoku Gakuin University

の高速実装として広く使用されている fpLLL[4]を使用した。
図1、2、3に、それぞれの場合の実行時間の比較結果を示す。図中の conventional がオリジナルのショートパッドアタックの格子、improved が[2]で導入された低次元格子の結果を示している。

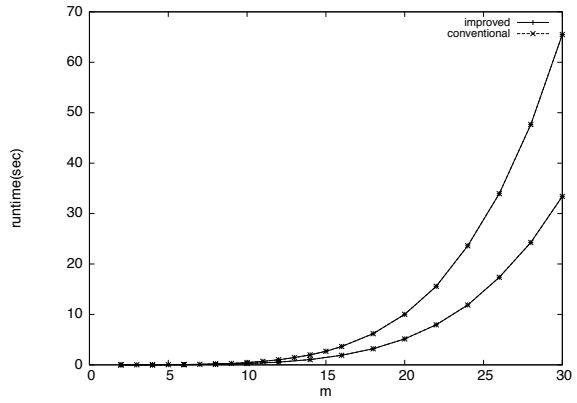

図1 1,024 ビットの場合

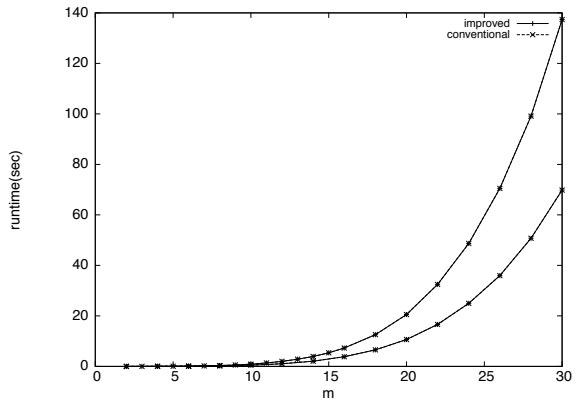

図2 1,536 ビットの場合

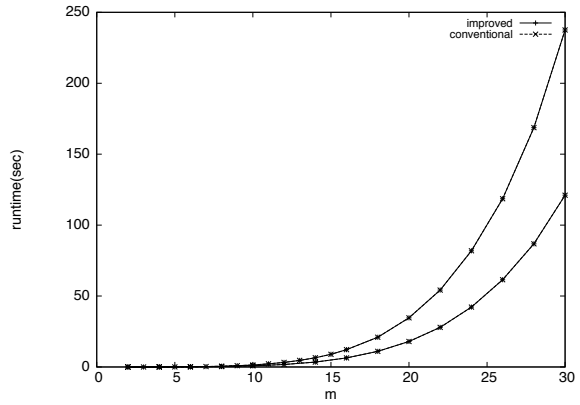

図3 2,048 ビットの場合

図から、$n = 1024,1536,2048$のそれぞれについて、オリジナルのショートパッドアタックの格子よりも、前節で導入した格子を使用した方が 2 倍程度高速に LLL アルゴリズムで短いベクトルが求まることが分かる。後者は前者より次元が半分であるにもかかわらず、高速化が 2 倍程度に留まるのは、格子基底の成分のビット長が、後者の方が大きいためと考えられる。例えば、$n = 1024$、$m = 2$の場合、オリジナルのショートパッドアタックの格子基底の成分のビット長はおよそ 15,000 程度であるが、前節で導入した格子基底の成分のビット長はおよそ 30,000 程度である。

3. まとめ

本稿では Coppersmith のショートパッドアタックとは異なる格子を使用するシミュレーション結果[2]を解説した。[2]で示された格子の次元は Coppersmith のショートパッドアタックで使用される格子の次元の半分であるが、一方で、格子基底の成分のビット長は Coppersmith のショートパッドアタックにおける格子基底のビット長よりも非常に大きい。[2]では、$n = 1024,1536,2048$の場合について、十分な数と思われる格子基底に対して LLL アルゴリズムを適用するシミュレーションが行われている。本稿では、[2]のシミュレーション結果の補足をしながら、高速化の要因を分析した。

参考文献：

[1] A.K. Lenstra, H.W. Lenstra, L. Lovàsz:Factoring Polynomials with Rational Coefficients. Mathematische Ann., vol. 261, pp. 513-534, (1982).
[2] M. Kaminaga, T. Suzuki, M. Fukase, Determining the Optimal Random-padding Size for Rabin Cryptosystems, arXiv:1807.05782 preprint(2018).
[3] P. Pessl and M. Hutter, ``Curved Tags - A Low-Resource ECDSA Implementation tailored for RFID,″ Radio Frequency Identification: Security and Privacy Issues Lecture Notes in Computer Science Vol. 8651, pp. 156-172(2014).
[4]The FPLLL development team, ``fplll, a lattice reduction library,″ 2016, Available at https://github.com/fplll/fplll