情報処理学会研究報告
IPSJ SIG Technical Report

MaxSATを利用したAES暗号鍵の復元*

廖 暁鵑[1]　越村 三幸[1]

概要：Cold boot attack はサイドチャネル攻撃の一つで，電源断されたメモリからデータを獲得しようとする．電源断の後，メモリのデータは次第に消失していくので，攻撃者が得るデータは壊れている．本稿では，壊れた AES 暗号鍵の復元を論じる．この復元手法として，SAT 問題に帰着させる SAT 符号化が提案されている．これは鍵のビット間の関係に着目した手法で，電源断の後，各ビットは電荷がある状態から電荷のない状態に次第に遷移していき，逆の遷移は起こらない，と仮定している．しかし実際には稀にではあるが，逆の遷移も起こる．本稿では，逆の遷移も起こることを仮定し，復元問題の MaxSAT 符号化を提案する．MaxSAT 符号化では，鍵のビット間の関係はハード制約に，電荷があるビットはソフト制約に符号化される．MaxSAT ソルバーは全てのハード制約を満たし，できるだけ多くのソフト制約を満たすような AES 鍵を見つけようとする．これが復元した鍵に対応する．

キーワード：AES，暗号鍵，cold boot attack，MaxSAT，SAT

Recovering AES Key Schedules with MaxSAT

Liao Xiaojuan[1]　Koshimura Miyuki[1]

Abstract: Cold boot attack is a side channel attack that recovers data from memory, which persists for a short period after power is lost. In the course of this attack, the memory gradually degrades over time and only a corrupted version of the data may be available to the attacker. Recently, great efforts have been devoted to reconstructing the original data from a corrupted version of AES key schedules, based on the assumption that all bits in the charged states tend to decay to the ground states while no bit in the ground state ever inverts. However, in practice, there is a small number of bits flipping in the opposite direction, called *reverse flipping errors*. In this paper, motivated by the latest work that formulates the relations of AES key bits as a Boolean Satisfiability problem, we move one step further by taking the reverse flipping errors into consideration and employing an off-the-shelf MaxSAT solver to accomplish the key recovery of AES-128 key schedules from decayed memory images. Experimental results show that, in the presence of reverse flipping errors, the MaxSAT approach enables reliable recovery of key schedules with significantly less time, compared with the SAT approach that relies on brute force search to find out the target errors.

Keywords: AES, encryption key, cold boot attack, MaxSAT, SAT

1. はじめに

DRAM(Dynamic Random Access Memory) は，キャパシタに電荷を蓄えることにより情報を記憶する．１ビット情報は，キャパシタ一つの電荷の有無により記憶される．DRAM は電源供給がなくなると記憶情報も失われる揮発性メモリであるが，実際には電源を切ってもしばらくは情報が残留することが指摘されている [3]．これによると電源断の後も数秒間，さらに DRAM を冷やしておけば数分間，場合によっては数時間，情報が保持される．この DRAM の残留情報を利用したサイドチャネル攻撃が cold boot attack であり，動作中あるいはシャットダウン後の計算機から情報を収集する．これは，メモリに重要なデータを保持するシステムにとって脅威となる攻撃である．

*本稿は，文献 [10] を日本語で要約したものです．

[1] 九州大学 大学院システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University

このようなシステムの例として，BitLocker，TrueCrypt，FileVault，LoopAES，dm-crypt などのディスク暗号化システムがある [4]．通常，このような自動暗号化システムは，ディスクをマウントしている間，暗号鍵をメモリに保持しており，これが攻撃の標的となる．

電源供給を絶つと DRAM の情報は徐々に消失していくので，最終的に攻撃者の得られるメモリデータは完全ではなく破損している．この破損したメモリデータから暗号鍵を復元には，暗号アルゴリズムに内在する冗長性を利用することによってなされる．実際，多くの暗号プログラムは処理速度を上げるため，前もって暗号鍵から計算したデータをメモリ上に保持している．ブロック暗号では，複数のラウンド鍵（roundkey）からなる鍵スケジュール（key schedule）が，前もってある鍵から計算されているのが普通である．鍵スケジュールは，その作られ方から冗長性があり，その一部が壊れていても元の鍵を復元することができる [4]．本稿では，cold boot attack によって得られる壊れた AES 鍵スケジュールから，元の鍵を復元することに焦点をあてる．

Advanced Encryption Standard (AES) は米国立標準技術研究所（NIST）が 2001 年に採用した共通鍵暗号方式である．ブロック暗号の一種であり，メモリから暗号鍵を読み取る cold boot attack の標的となりうる．AES 暗号鍵は複数のラウンド鍵から構成される鍵スケジュールのことで，ラウンド鍵は，ある初期鍵（initial key）から鍵拡張（key expansion）アルゴリズムによって得られる [1]．AES の初期鍵の長さは，128，192，256 ビットであり，それぞれ AES-128，AES-192，AES-256 で参照される．AES 鍵スケジュールには情報の冗長性があり，これを利用して攻撃者はメモリの内容から，たとえ一部が壊れていても，初期鍵を復元できる．

2. 先行研究

DRAM はキャパシタに蓄えられた電荷によって情報が記憶されるが，電荷は時間とともに失われるために常に電荷を更新し続けなければならない．DRAM の電源が断たれると更新ができなくなり，情報は消失する．Halderman らの実験によると，時間が経つにつれて，殆どのメモリビットは電荷のない状態（ground state）に遷移するが，稀にではあるが，電荷のある状態（charged state）に遷移するビットもある [3]．本稿では，電荷のある状態を 1，電荷のない状態を 0 で表す．このように表すと，ほとんどの状態遷移は 1 から 0 であるが，稀に 0 から 1 に遷移することになる．

1 から 0 に遷移する確率を δ_0，逆に 0 から 1 に遷移する確率を δ_1 と標記することにする．一般的に時間が経つにつれて δ_0 は 1 に近づく．対照的に，δ_1 は比較的一定で，0.05%から 0.1%のごく小さい値をとる [3]．δ_1 が 0 であると仮定しているか，0 でないと仮定しているかに着目して先行研究を考察する．

- 純粋仮定（perfect assumption）：実際の cold boot attack では，δ_1 は非常に小さい．そこで，δ_1 を 0，つまり 0 から 1 への遷移は起こらないと仮定する．この仮定の下では，cold boot attack によって得られる DRAM の各ビットの状態のうち，1 は全て正しいことになる．
- 現実仮定（realistic assumption）：δ_1 が 0.05%から 0.1%と仮定する．この仮定の下では，1 から 0 への遷移も 0 から 1 への遷移も共に起こりうるため，得られる DRAM の各ビットの状態は 0 も 1 もどちらも完全には正しいと言えない．

実際にコンピュータからメモリ情報を読み取り，そこから暗号鍵を見つけ出す手法は，[4] に詳しい．ここでは，見つけ出した AES 鍵スケジュール[*1]を復元する従来研究を概説する．

AES-128 では，128 ビットの初期鍵を元に 10 個のラウンド鍵（それぞれ 128 ビット）が順次作られ，初期鍵とラウンド鍵を合わせた 1408（=128 × 11）ビットの鍵スケジュールが用いられる．Halderman らの復元法では，順次作られるラウンド鍵の線形性に着目している [3]．鍵スケジュール全体を一気に復元する代わりに，直接関係のあるラウンド間の部分関係に着目し，部分的に尤もらしい鍵候補を作り出し，それを元に鍵スケジュール全体を再構築する．Halderman らは，$\delta_0 = 15\%$ と $\delta_1 = 0.1\%$ の下で数秒，$\delta_0 = 30\%$ と $\delta_1 = 0.1\%$ の下で 30 秒以内で AES-128 の鍵を復元した．我々の知る限り，[3] のみが現実仮定で復元を行っている．

Tsow は，AES 鍵スケジュールの構造をより上手く利用し，tree-pruning 制約の下で深さ優先探索を行うアルゴリズムを提案している [14]．Tsow は，$\delta_0 = 70\%$ と $\delta_1 = 0$ の仮定の下，平均 300 秒で AES-128 鍵の復元を行っている．Tsow は，現実仮定でも復元できるといっているが，その方法や性能は示されていない [14]．

Kamal らは，AES 鍵復元問題を SAT 問題に帰着する手法を示した [5]．これは，命題論理の充足可能性（Boolean satisfiability）を判定する SAT ソルバーの高速化に着目した手法である．Kamal らも純粋仮定の下で復元を行っている．この仮定の下では，メモリから読み取られたビットの状態で 1 となっているものは，充足すべき制約となる．また，ラウンド鍵間の制約も充足すべき制約となる．SAT ソルバーはこれら両制約を充足するような解を求める．その SAT 解が復元した鍵に対応する．XOR 演算をサポートする SAT ソルバー CryptoMiniSat を利用して，AES 鍵の復元性能を格段に高め，δ_0 が高い場合でも復元できるように

[*1] これには確率 δ_0 と δ_1 で誤りがある．

なった．Kamal らによると，AES-128 鍵スケジュールが $\delta_0 = 70\%$ と $\delta_1 = 0$ を満たす場合，平均 1.2 秒で復元でき，$\delta_0 = 80\%$ と $\delta_1 = 0$ の場合でも復元可能である．

従来手法と提案手法の比較を**表 1** に示す．

3. SAT と MaxSAT

命題論理の充足可能性判定問題（SAT）は，与えられた命題論理式の充足可能性を判定する問題である．SAT ソルバー [15] の研究が近年大きく発展し，大規模な SAT 問題が解けるようになってきたことを受けて，SAT の枠組みを多方面に拡張する試みも盛んに行われている．MaxSAT もその一つで，通常の SAT で求めるものが制約を満たす「解」であると解釈できるのに対し，MaxSAT の求めるものは「最適解」であると解釈できる．

SAT 問題は，連言標準形（CNF: Conjunctive Normal Form）で表される命題論理式（CNF 式）で与えられるのが一般的である．CNF 式は，節（clause）の連言（論理積）であり，節は，リテラル（literal）の選言（論理和）である．リテラルは，変数（例えば，x）またはその否定（例えば，$\neg x$）である．本論文では，CNF 式をそれを構成する節の集合と同一視する．つまり，連言を省略する．変数への値割当のうち，全ての節を充足するような値割当が SAT 解であるのに対し，充足する節の数が最大となる値割当が MaxSAT 解である [8], [16]．したがって，SAT 解があれば，それは MaxSAT 解でもある．一方，SAT 解がない，つまり充足不能な場合でも，MaxSAT 解はあり，それは，問題がどの程度，充足不能なのかを示している，と考えることができる．

本研究では，MaxSAT の一種である部分 MaxSAT（PMS:partial MaxSAT）を用いて鍵の復元を行う．PMS 問題の各節は，ハード（hard）かソフト（soft）かのいずれかと宣言される．全てのハード節を充足する値割当のうち，充足するソフト節の数が最大となる値割当が PMS 解である．

4. SAT/MaxSAT による AES-128 鍵スケジュールの復元

AES は共通鍵暗号であり，暗号化と復号に同じ暗号鍵を用いる．本稿では AES-128 鍵の復元について論ずるが，AES-192，AES-256 も同様の方法で復元できる．本節では復元法のあらましを述べる．より詳しく知りたい場合は，[10] を参照されたい．

AES-128 の鍵スケジュールは 11 個のラウンド鍵からなる．ラウンド鍵はそれぞれ 128 ビットある．0 番目のラウンド鍵は初期鍵それ自身である．残りの 10 個のラウンド鍵は，初期鍵からラウンド関数と呼ばれる変換を繰り返し適用することによって得られる [12]．ラウンド関数は全単射で，主に次の二つの操作を行う．

- バイト単位でデータを置換する．
- 128 ビットデータ中のビットの順番を入れ替える．

r ラウンド目の第 i 番目ビットを b_i^r と表すことにする．なお，$0 \le r \le 10, 0 \le i \le 127$ である．このとき，ラウンド関数は次のような性質を持つ．

- b_i^r $(0 \le i \le 31, 1 \le r \le 10)$ は，$r-1$ 番目のラウンド鍵の 9 ビットから決定される．
- b_i^r $(32 \le i \le 127, 1 \le r \le 10)$ は，$r-1$ 番目のラウンド鍵の 2 ビットから決定される．

したがって，ビットの誤りは関連する 2 ビットあるいは 9 ビットを元に正せる．また，鍵スケジュールがいくつものビット誤りを含んでいたとしても，十分な数のビット情報があれば，復元できる．

ラウンド関数は容易に CNF 式に変換できる．ただ，ラウンド関数は XOR 演算を含んでいるので，これを通常の変換で CNF 式に変換すると長大な式になる．通常の変換では，長さ n の XOR 式の変換には 2^{n-1} 個の節が必要だからである．本研究で利用した SAT ソルバー CryptoMiniSat [13] は，XOR 式をサポートしているので，XOR 式変換の指数的長大化の問題は生じない．しかし，XOR 式をサポートしている MaxSAT ソルバーは存在しないので，変換に何らかの工夫が必要となる．

本研究では，長大な XOR 式に対しては，補助変数（auxiliary Boolean variable）を導入することにより，指数的長大化を回避する．具体的には，長大な XOR 式を長さ 5 の XOR 式にグループ分けし，それぞれのグループに一つの補助変数を対応づける．こうして 1，長さ 133 の XOR 式の変換は，33 個の補助変数を導入することにより，1036 個の節に変換される．

4.1 純粋仮定での復元

$\delta_1 = 0$，つまりビットの 0 から 1 への遷移は起こらないとする純粋仮定では状態 1 は間違いなく正しい．したがって，状態 1 のビットの情報はそのまま，CNF 式になる．例えば，$b_{22}^9 = 1$ であれば，単位節 b_{22}^9 が得られる．一方，状態 0 は全く信用できないので，これについては何の制約を課さない．

状態 1 のビットから得られる単位節とラウンド関数から得られる CNF 式を合わせて SAT ソルバーに解かせて得られる SAT 解が，復元した鍵スケジュールを表す．

4.2 現実仮定での復元

δ_1 が 0 ではなく 0.05%から 0.1%とする現実仮定では，状態 1 は必ずしも正しい訳ではないが，大体は正しい．1408 ビットからなる鍵スケジュールの状態 1 のビットで正しくないのは実際には 0 個から 2 個である．現実仮定の下で SAT を用いた復元は以下のようになる．

先ず，純粋仮定の下で復元を試みる．SAT 解が求まれば，

表 1 各復元法の比較

	従来手法			提案手法	
	Halderman et al. [3]	Tsow et al [14]	Kamal et al. [5]	SAT approach *	MaxSAT approach
純粋仮定	適用可	適用可	適用可	適用可	適用可
現実仮定	適用可	適用不可	適用不可	適用可	適用可
最大の δ_0 での性能	Recover half of keys with $\delta_0 = 30\%$ and $\delta_1 = 0.1\%$ within 30 seconds	Recover keys with $\delta_0 = 70\%$ and $\delta_1 = 0$ within 300 seconds	Recover keys with $\delta_0 = 78\%$ and $\delta_1 = 0$ within 6960 seconds	Recover keys with $\delta_0 = 76\%$ and $\delta_1 = 0.1\%$ within 1120 seconds	Recover keys with $\delta_0 = 76\%$ and $\delta_1 = 0.1\%$ within 300 seconds

* Kamal et al. [5] の繰り返し適用

復元は成功である．この場合は，鍵スケジュールの状態 1 のビット情報は全て正しかったことになる．状態 1 のビット情報に誤りが一つでもあれば，SAT 解は求まらない．

この場合，状態 1 のビットのうち一つは誤り，つまり状態 0 であったとして復元を試みる．SAT 解が求まれば復元成功である．SAT 解が求まらなければ，状態 1 の別ビットが誤っているとして復元を試みる．したがって，この試みは最悪 n 回行わなければならない．ここで n は，鍵スケジュールの状態 1 のビット数である．

n 回の試みが全て失敗した場合，状態 1 のビット情報には少なくとも二ヶ所に誤りがあることになる．この場合，二つが誤り，つまり二つの状態 1 が状態 0 であったとして復元を試みる．この試みは，最悪 ${}_nC_2$ 回行わなければならない．以上のように，SAT による復元では，繰り返し SAT 問題を解く必要がある．

一方，MaxSAT を用いた復元では次のような一つの MaxSAT 問題を解けばよい．

- ラウンド関数から得られる節をハードと宣言する．
- 状態 1 のビットから得られる単位節をソフトと宣言する．例えば，$b^9_{22} = 1$ であれば，単位節 b^9_{22} をソフトと宣言する．

MaxSAT ソルバーは，出来るだけ多くのソフト節を満足するよう解を探し出すので，得られる MaxSAT 解では，状態 1 のビット情報の誤り数が最小となり，これが復元した鍵スケジュールに対応する．

5. 評価実験

本節では，現実仮定の下での AES-128 鍵スケジュールの復元実験を報告する．

5.1 実験に用いた鍵スケジュール

先ず，128 ビットの初期鍵をランダムに生成し，ラウンド関数を用いて 10 個のラウンド鍵を作り，1408 ビットの鍵スケジュールを作る．こうして得られる鍵スケジュールの 0 のビット数と 1 のビット数は，ほぼ等しくなる．

次に，δ_0 の確率でランダムに 1 を 0 に，δ_1 の確率でランダムに 0 を 1 に書き換える．実験では，δ_1 は 0.1%で固定，δ_0 は 30%から 76%の間で振らした．δ_0 が 70%を超えると復元時間が急に増加するので 30%から 70%までは 10%刻み，70%から 76%の間は 2%刻みで実験を行った．それぞれの δ_0 で 100 個の鍵スケジュールを作った．このようにして作った鍵スケジュールで，0 から 1 にビットの書き換えが起こった数は，0 から 2 であった．

次に

(1) 0 から 1 への書き換えが 1 ヶ所，つまりビット 1 の誤りが 1 ヶ所のみの鍵スケジュール 100 個
(2) 0 から 1 への書き換えが 2 ヶ所，つまりビット 1 の誤りが 2 ヶ所のみの鍵スケジュール 40 個

を用いて実験を行った．十分な時間がなかったので，(2) については 40 個の鍵スケジュールを用いた．また，δ_0 も 74%までしか実験できなかった．

なお，δ_0 が増えるに従い，正しいであろうビット数は減少していくので，復元した鍵スケジュールが間違うこともありうるが，今回の 76%までの復元実験では，そのようなことはなかった．

5.2 実験結果

実験には，2.6GHz quad-core Intel i5-2540 プロセッサ（8GB RAM）を用いた．SAT ソルバーには XOR 式をサポートする CryptoMiniSat [13] を MaxSAT ソルバーには Pwbo2.0 [11] を用いた．MaxSAT ソルバーについては，幾つかのソルバーを用いて予備実験を行い，その中で Pwbo2.0 が最も性能が良かった．予備実験に用いたソルバーは Pwbo2.0 の他に，Akmaxsat [7]，WMaxSatz [9]，QMaxSAT [6]，QMaxSAT-g2 [6]，WPM1 [2]，PM2 [2] である．

SAT ソルバーを用いた暗号鍵の復元では，ソルバーが繰り返し呼ばれるので，性能評価に二種類の時間を用いる．一つは solver time，もう一つは CPU time である．前者は CryptoMiniSat の正味の計算時間，後者は問題を解くのに要した総時間で solver time に入力ファイルの変更時間を加えた時間である．ファイルの変更はソルバーが呼ばれるたびに必要になる．以降，SAT と MaxSAT の比較では solver time を用い，CPU time は参考にとどめる．**表 2** に

表 2 SAT/MaxSAT の平均実行時間（秒）

Decay Factor	CryptoMiniSat		Pwbo2.0
δ_0 (%)	Solver Time	CPU Time	Solver Time
30	45.81	462.24	0.94
40	28.47	228.88	0.96
50	19.67	97.24	1.17
60	26.52	89.19	1.56
70	225.38	255.10	12.53
72	678.45	718.03	26.78
74	1004.16	1035.13	231.61
76	1116.35	1143.29	296.42

δ_0 を 30%から 76%まで増やしていった実験結果を示す．

全般的に δ_0 が変化しても，MaxSAT の方が SAT に比べてより速く鍵の復元に成功している．δ_0 が増えるに従い，MaxSAT の solver time は単調に増えているが，SAT の solver timer は減って増えている．この理由は次のように考えられる．先ず，δ_0 が小さい時，特に 30%の時は，鍵スケジュール中の 1 の数は比較的多い．つまり，1 であるビットの内，どれかが誤っている可能性も高くなる．従って，CryptoMiniSat が呼ばれる回数も多くなる．こうして solver time，CPU time 共に長くなる．次に，δ_0 が大きくなるにつれて 1 であるビット数は減っていき，したがって，どれかが誤っている可能性も低くなる．こうして，CryptoMiniSat が呼ばれる回数が減り，solver time，CPU time 共に短くなる．さらに，δ_0 が大きくなると，今度は問題それ自身が難しくなり，solver time が長くそして殆どを占めるようになる．

次に，(1) の場合，つまりビット 1 の誤りが 1 ヶ所のみの鍵スケジュールの復元と (2) の場合，つまりビット 1 の誤りが 2 ヶ所のみの鍵スケジュールの復元について，SAT と MaxSAT の性能比較を行う．**表 3** が (1) の場合，**表 4** が (2) の場合である．

(1) の場合，MaxSAT の方が SAT に比べ幾分速い．速い．特に，δ_0 が 76%の時，CryptoMinSat は最長 2.9 時間，平均 8 分，中央値 3 分であるのに対し，MaxSAT は最長 1.8 時間，平均 6.4 分，中央値 2.4 分である．

(2) の場合，(1) の場合より MaxSAT の優位性がより顕著である．特に，$\delta_0 = 74\%$ の時，CryptoMiniSat は平均 4 時間，中央値 2.2 時間を要している．さらに，最長でほぼ 2 日を要することもあった．一方，Pwbo2.0 は平均 36 分，中央値 7.9 分で，最長は 5.7 時間であった．

最後に AES-128 のラウンド関数の CNF 式表現のサイズについて評価する．XOR 式をグループ分けし補助変数を導入することにより，CNF 式の指数的長大化を回避したが，それでも 372,240 個の節が必要であった．XOR 式をサポートする CryptoMinSat では，対応する XOR 式と節の数は合わせて 51,440 個であったので，7 倍程多かったことになる．それでもなお，MaxSAT の方が，SAT よりも計算時間の点で優れていることが実験より明らかになった．XOR 式をサポートする MaxSAT ソルバーを開発すれば，MaxSAT の優位性がより顕著になることが期待できる．

6. おわりに

本稿では, cold boot attack で得られる AES 鍵スケジュールからの AES 鍵の復元を論じ，MaxSAT を利用した復元法を提案した．従来の手法の多くが純粋仮定の下での復元であったのに対し，提案手法では MaxSAT のソフト節を利用することにより，現実仮定の下での復元を試みる．SAT を利用した従来手法では，繰り返し SAT ソルバーを呼び出さなければならないが，MaxSAT を利用した提案手法では，MaxSAT ソルバーの呼び出しは 1 回で済む．

AES のラウンド関数は XOR 式を含み，これを通常の変換法で CNF 式にするとその長さは指数的になる．本研究では，XOR 式を長さ 5 の XOR 式にグループ分けし，補助変数を導入することにより指数的長大化を回避した．この長さ 5 は適当に決めたもので，定性的あるいは定量的には何の根拠はない．最適な長さの探求は今後の課題である．

謝辞 本研究は JSPS 科研費 25330085 の助成を受けたものです．

参考文献

[1] Federal Information Processing Standards Publication (FIPS 197). Announcing the advanced encryption standard (aes). 2001.

[2] Carlos Ansótegui, María Bonet, and Jordi Levy. Solving (weighted) partial maxsat through satisfiability testing. In *Proc. of International Symposium on the Theory and Applications of Satisfiability and Testing (SAT'09)*, pages 427–440, 2009.

[3] J. Alex Halderman, Seth. D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold-boot attacks on encryption key. In *Proc. USENIX USENIX Security Symposium (SEC'08)*, pages 45–60, California, USA, Jul 2008.

[4] Nadia Anne Heninger. *Error Correction and the Cryptographic Key.* PhD thesis, Univ. of Princeton, May 2011.

[5] Abdel Alim Kamal and Amr M. Youssef. Applications of SAT solvers to aes key recovery from decayed key schedule images. In *Proc. of International Conference on Emerging Security Information Systems and Technologies (SECURWARE'10)*, pages 216–220, 2010.

[6] Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. Qmaxsat: A partial max-sat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 8(1/2):95–100, 2012.

[7] Adrian Kugel. Improved exact solver for the weighted max-sat problem. In *Proc. of POS*, pages 15–27, 2010.

[8] Chu Min Li and Felip Manyà. *MaxSAT, Hard and Soft Constraints*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 19, pages 613–631. IOS Press, 2009.

表 3 SAT/MaxSAT の実行時間の統計情報（ビット 1 の誤りが 1 ヶ所）

Decay Factor δ_0 (%)			30	40	50	60	70	72	74	76
CryptoMiniSat	Solver Time (秒)	平均	2.05	1.31	1.97	5.03	43.53	47.60	280.83	480.10
		中央	1.76	1.17	1.44	2.29	29.02	43.63	67.48	186.01
		最大	5.16	3.94	8.91	78.32	642.39	233.65	3491.27	10498.97
		最小	0.09	0.07	0.11	0.18	1.05	0.58	0.75	2.02
		標準偏差	1.42	0.97	1.63	8.68	69.66	43.68	646.06	1196.45
	CPU Time (秒)	平均	10.68	6.96	7.33	9.27	47.51	51.45	284.53	483.43
		中央	9.14	6.38	6.02	6.24	31.80	37.30	71.43	188.63
		最大	23.13	18.03	20.88	88.64	646.95	240.95	5691.97	10503.14
		最小	0.33	0.16	0.30	0.22	1.11	1.26	0.79	2.62
		標準偏差	6.65	5.01	5.16	10.58	70.63	45.06	750.88	1165.96
Pwbo2.0	Solver Time (秒)	平均	1.04	1.09	1.35	2.25	14.95	28.35	122.52	384.35
		中央	0.79	0.91	1.15	1.94	8.95	13.31	25.22	142.75
		最大	2.37	2.40	3.24	6.35	262.72	599.01	3122.61	6492.83
		最小	0.71	0.70	0.72	0.77	1.03	1.51	2.28	4.34
		標準偏差	0.45	0.40	0.54	1.31	29.02	61.90	344.66	869.45

表 4 SAT/MaxSAT の実行時間の統計情報（ビット 1 の誤りが 2 ヶ所）

Decay Factor δ_0 (%)			30	40	50	60	70	72	74
CryptoMiniSat	Solver Time (秒)	平均	198.64	162.25	224.69	329.62	3047.82	4909.57	14715.61
		中央	171.62	186.39	127.72	153.84	2077.35	3517.53	7936.68
		最大	387.54	371.53	1358.53	2112.36	9747.16	17027.59	161389.01
		最小	13.74	8.61	7.30	27.48	38.42	10.44	62.77
		標準偏差	87.96	89.68	276.60	493.63	2907.31	5077.26	26695.94
	CPU Time (秒)	平均	1838.27	1438.00	1175.43	908.88	3366.99	5309.83	14967.69
		中央	1547.67	1582.85	902.14	884.41	2208.29	3663.32	8249.49
		最大	3884.49	3182.16	4297.18	3362.83	10226.60	17670.25	161885.50
		最小	64.04	37.68	65.89	68.82	145.92	29.25	90.74
		標準偏差	1052.84	900.46	917.92	770.54	3029.01	5224.46	26753.92
Pwbo2.0	Solver Time (秒)	平均	1.46	1.56	2.18	4.68	47.73	245.18	2160.49
		中央	1.02	1.17	1.88	3.17	37.34	97.37	473.76
		最大	7.12	8.22	5.83	19.19	220.87	1848.42	20687.95
		最小	0.23	0.90	0.91	1.00	5.05	6.71	20.05
		標準偏差	1.11	1.22	1.20	3.94	37.34	358.49	3982.25

[9] Chu Min Li, Felip Manyà, Nouredine Mohamedou, and Jordi Planes. Exploiting cycle structures in max-sat. In *Proc. of International Symposium on the Theory and Applications of Satisfiability and Testing (SAT'09)*, pages 467–480, 2009.

[10] Xiaojuan Liao, Hui Zhang, Miyuki Koshimura, Hiroshi Fujita, and Ryuzo Hasegawa. Using MaxSAT to Correct Errors in AES Key Schedule Images. In *Proc. of ICTAI 2013*, pages 284–291. IEEE, 2013.

[11] Ruben Martins, Vasco Manquinho, and Inês Lynce. Parallel search for maximum satisfiability. *AI Communications*, 25(2):75–95, 2012.

[12] Dr. S. Morioka. 数学いらずの AES 暗号 SubBytes 設計ガイド. *Design Wave Magazine*, pages 152–157, January 2004.

[13] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In *Proc. of International Symposium on the Theory and Applications of Satisfiability and Testing (SAT'09)*, pages 244–257, 2009.

[14] Alex Tsow. An Improved Recovery Algorithm for Decayed AES Key Schedule Images. In *Proc. of Selected Areas in Cryptography (SAC'09)*, pages 215–230, 2009.

[15] 井上 克巳, 田村 直之. SAT ソルバーの基礎. **人工知能学会誌**, 25(1):57–67, 2010.

[16] 平山 勝敏, 横尾 真. *-SAT: SAT の拡張. **人工知能学会誌**, 25(1):105–113, 2010.