平成 28 年度 卒業論文概要			
所　　属	機械情報工学科	指導教員	光來 健一
学生番号	13237028	学生氏名	兒玉朋大
論文題目	コールドブート攻撃対策のためのキャッシュ暗号化とディスク暗号化の統合		

1　はじめに

スマートフォンやタブレットの普及にともない、端末の盗難による情報漏洩のリスクが高まっている。そこで、Android ではディスク暗号化の機能を提供し、ディスクからの情報漏洩を防いでいる。一方、近年では、端末のメモリから強制的に情報を盗み出すコールドブート攻撃が脅威になっている。Android ではファイルアクセスを高速化するために一度アクセスしたディスク上のデータはメモリ上にキャッシュされている。コールドブート攻撃が行われるとキャッシュ上のデータが漏洩してしまうため、ディスクを暗号化していても情報漏洩を防ぐことができない。これまでに、キャッシュを暗号化することで盗難時の情報漏洩を防ぐ Cache-Crypt[1] が提案されてきた。Cache-Crypt では、Android アプリがファイルにアクセスする時だけキャッシュの必要な領域を復号し、アクセスが終わったら再び暗号化する。しかし、ディスク暗号化と併用すると、Cache-Crypt とディスク暗号化がそれぞれ独立にデータの暗号化・復号化を行うため、オーバヘッドが大きくなるという問題があった。

本研究では、Cache-Crypt によるキャッシュ暗号化とディスク暗号化を統合することで暗号化・復号化のオーバーヘッドを減らす Cache-Crypt+ を提案する。

2　キャッシュ暗号化

Cache-Crypt は端末の盗難時の情報漏洩を防ぐために、ページキャッシュと呼ばれるメモリ上のディスクキャッシュを暗号化する。Cache-Crypt はディスクから読み込んだデータを暗号化してページキャッシュに格納する。アプリがファイルを読み込む時には、ページキャッシュの対応する領域を一時的に復号化してアプリのバッファにコピーし、その後で再びページキャッシュを暗号化する。ファイルにデータを書き込む時には、データを暗号化してページキャッシュの対応する領域に書き込む。そのデータをディスクに書き出す時には、ページキャッシュを一時的に復号してからディスクに書き込み、その後で再びページキャッシュを暗号化する。これにより情報漏洩の対象をコールドブート攻撃時に復号されていたわずかなページキャッシュのみに限定することができる。

しかし、Cache-Crypt を Android のディスク暗号化である dm-crypt と併用すると、暗号化・復号化のオーバヘッドが大きくなる。これは、Cache-Crypt と dm-crypt がそれぞれ独立にデータの暗号化・復号化を行ってしまうためである。ディスクからデータを読む込む時、図 1 のようにまず dm-crypt が暗号化されたデータを復号してページキャッシュに書き込む。そのデータを Cache-Crypt が異なる暗号アルゴリズム・暗号鍵を用いて暗号化する。アプリがそのデータを読み込む時に

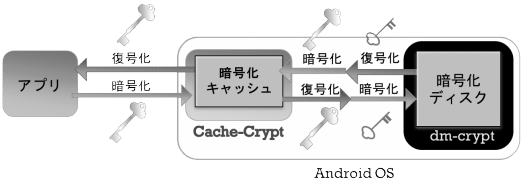

図 1　Cache-Crypt におけるファイルの読み書き

は Cache-Crypt がページキャッシュを復号し、読み込み完了後に再び暗号化する。このように、アプリがディスクのデータを読み込む際には暗号化と復号化が 2 回ずつ必要になる。一方、アプリがデータを書き込む時には、Cache-Crypt がデータを暗号化する。そのデータをディスクに書き出す時には、まず Cache-Crypt がページキャッシュを復号し、dm-crypt が暗号化する。ディスクへの暗号データの書き出しが完了したら、Cache-Crypt が再びページキャッシュを暗号化する。このように、アプリがディスクにデータを書き込む際にも暗号化と復号化が 2 回ずつ必要になる。

Cache-Crypt では暗号化・復号化のオーバヘッドを減らすために暗号化の遅延を行っている。ディスクからデータを読み込む時に、Cache-Crypt は dm-crypt が復号したデータをすぐには暗号化しない。そのため、アプリがすぐにそのデータを読み込めば Cache-Crypt による復号化が不要になる。一定時間たつと Cache-Crypt はそのページキャッシュを暗号化する。このように、ファイルの一般的な読み込みの際には暗号化と復号化が 1 回ずつで済む。一方、アプリがデータを書き込んだ時、Cache-Crypt はすぐには暗号化を行わず、ディスクに書き出す時に dm-crypt が暗号化を行う。一定時間が経過すると Cache-Crypt がページキャッシュを暗号化するため、暗号化と復号化は 1 回ずつで済む。しかし、ページキャッシュ上のデータがしばらく暗号化されないままになるため、セキュリティが低下する。

3　Cache-Crypt+

本研究では Cache-Crypt と dm-crypt を統合することで安全に暗号化・復号化のオーバヘッドを減らす Cache-Crypt+ を提案する。Cache-Crypt+ は、ディスクとページキャッシュの間ではデータの暗号化・復号化を行わず、暗号データを用いて読み書きを行う。そして、アプリとページキャッシュの間でのみ暗号化・復号化を行い、暗号化・復号化の回数を減らす。

Cache-Crypt+ では dm-crypt と同じ暗号アルゴリズムや暗号鍵を用いることで統合を可能にする。その結果、Cache-Crypt+ では従来の dm-crypt の管理ツールをそのまま利用することができる。その代わり、Cache-Crypt では 16 バイトの暗号化ブロック単位できめ細かく暗号化していたが、Cache-

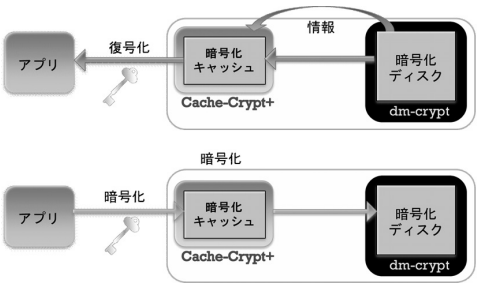

図 2　Cache-Crypt+ におけるファイルの読み書き

Crypt+ では 512 バイトのセクタ単位で暗号化を行う。また、暗号アルゴリズムの初期ベクトルにセクタ番号を用いる。

3.1　ファイルの読み込み

Cache-Crypt+ では、図 2 上のようにディスクからの読み込み時には dm-crypt での復号化を行わず、暗号化されたままの状態でページキャッシュに読み込む。その際に、後で Cache-Crypt+ が復号化を行うために必要となる情報をページキャッシュに登録しておく。この情報には暗号鍵やセクタ番号などが含まれる。アプリがそのデータを読み込む時には、ページキャッシュに登録された情報を用いて復号を行う。復号したデータをアプリのバッファにコピーし、その後で再度ページキャッシュを暗号化する。そのため、アプリがディスクからデータを読み込む際の暗号化と復号化は 1 回ずつとなる。一方、ファイル名などのメタデータの場合には、暗号化されたままページキャッシュ上に置いておくと OS の実行に支障をきたすため、ディスクからの読み込み完了時に復号化を行う。メタデータには通常、機密情報は含まれないため、復号してもセキュリティは低下しない。

3.2　ファイルへの書き込み

アプリがデータをファイルに書き込む際には、Cache-Crypt+ は図 2 下のようにデータを暗号化してからページキャッシュに書き込む。その後、dm-crypt は暗号化されたデータをそのままページキャッシュからディスクに書き出す。そのため、アプリがディスクにデータを書き込む際には暗号化が 1 回だけで済む。メタデータの場合には、Cache-Crypt+ はそれを暗号化せずにページキャッシュに書き込み、dm-crypt がディスクに書き出す際に暗号化する。

Cache-Crypt+ が dm-crypt と同じ暗号処理を行えるようにするために、書き込みに先立って暗号鍵やセクタ番号などの情報をページキャッシュに登録する。これは、書き込み時にページキャッシュを確保し、ディスク上の領域を割り当てる際に行う。この登録は dm-crypt を用いて暗号化されたディスクに書き込む場合にのみ行う。dm-crypt で暗号化されているかどうかを判別するために、まず、書き込み先がデバイスマッパのデバイスかどうかを判定する。次に、デバイスマッパ・テーブルを調べて dm-crypt であるかどうかの判定を行う。

3.3　ページキャッシュの状態管理

Linux はページキャッシュを 4KB 単位で管理しているが、Cache-Crypt+ での暗号化は 512 バイト単位で行われる。そのため、ページキャッシュの状態を 512 バイトのセクタ単位で管理し、それぞれについて暗号化されているかどうかを記録する。この情報に基づいて、ページキャッシュ上で暗号化されていない場合だけ暗号処理を行い、データが暗号化されている場合だけ復号処理を行う。

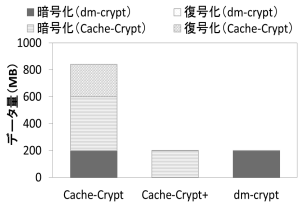

図 3　書き込み時の暗号化・復号化のデータ量

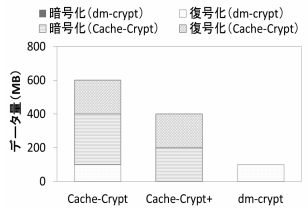

図 4　読み込み時の暗号化・復号化のデータ量

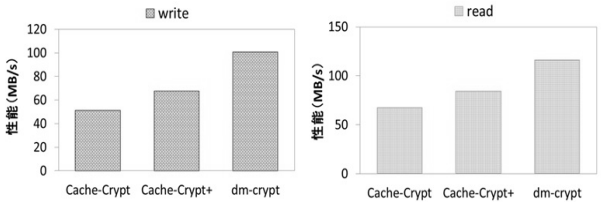

図 5　ファイルの読み書き性能

4　実験

Cache-Crypt+ において暗号化・復号化のオーバヘッドがどの程度減るかを確かめる実験を行った。本実験では、iozone ベンチマークを用いてファイルの逐次読み書きを行った時に、暗号化・復号化が行われたデータ量および読み書き性能を測定した。比較として、Cache-Crypt を dm-crypt と統合せずに用いた場合と dm-crypt のみを用いた場合についても測定を行った。実験には、Intel Xeon W3550 の CPU、6GB のメモリ、457GB の HDD を搭載したマシンを用い、Linux 3.4.112 を動作させた。

書き込み時に暗号化・復号化が行われたデータ量を図 3 に示す。Cache-Crypt と比べて、Cache-Crypt+ では暗号化・復号化されたデータ量は 1/4 になった。これは dm-crypt のみを用いた場合と同程度である。内訳を見ると、dm-crypt での暗号化と Cache-Crypt での復号化がなくなり、Cache-Crypt での暗号化が半分になっている。図 4 に読み込み時のデータ量を示す。Cache-Crypt+ で暗号化・復号化されたデータ量は Cache-Crypt の 2/3 になった。それにともない、図 5 に示すように、Cache-Crypt+ は Cache-Crypt よりも書き込みで 32%、読み込みで 25% 性能が向上した。

5　まとめ

本研究では、Cache-Crypt によるキャッシュ暗号化と dm-crypt によるディスク暗号化を統合することで暗号化・復号化のオーバーヘッドを減らす Cache-Crypt+ を提案した。Cache-Crypt+ ではアプリとページキャッシュの間でのみ暗号化・復号化を行い、ディスクとページキャッシュの間では暗号化データをそのまま扱う。今後の課題は、Cache-Crypt の遅延暗号化を Cache-Crypt+ に組み込むことや、Android 端末を用いて Cache-Crypt+ の性能評価を行うことである。

参考文献

[1] 福田直人, 光来健一, Android 端末のメモリ暗号化によるコールドブート攻撃対策, 第 72 回 CSEC 研究会, 2016 年.