機密実行環境を利用した自己破壊・出現データの実装方式の提案

加藤風芽[†]　新城靖[†]

1. はじめに

インターネットでは、日常的に個人情報等のセンシティブなデータが交換されている。**自己破壊データ (self-destructing data)** は、時間経過等の条件が満たされた時、自動的に利用できなくなる仕組みを持ったデータである[3)]。センシティブな情報を自己破壊データとして受け渡すことができれば、情報の漏洩や不適切な利用を防ぐために役立つと期待できる。

本研究では、**機密実行環境**を利用し、セキュアで利便性の高い自己破壊データを実現する。機密実行環境または Trusted Execution Environment (TEE) は、ハードウェアによって実装された、プログラムのコードやデータを秘匿したまま実行できる環境である。本研究では自己破壊とは逆に、ある時刻から利用可能になるデータ（**自己出現データ (self-emerging data)**）も実現する。

2. 関連研究

身近な自己破壊データの応用例に、Instagram 等の SNS (Social Networking Service) における**ストーリー**機能がある。これは決まった時間が経つと閲覧できなくなる投稿である。これは通常のソフトウェアで実装されており、プログラムの改変等に対する保護は提供していない。

Vanish[3)] は、分散ハッシュテーブル (DHT) を利用した自己破壊データの実装である。Vanish では、データの復号鍵を (k, n) しきい値法により n 個のシェアと呼ばれる素片に分け、別々に DHT に保存する。有効期限が過ぎたシェアは各ノードで破棄される。$(n-k+1)$ 台が正しく破棄を行えばよいため、全てのノードを信頼しなくてよい。しかし、データにアクセスする際には DHT に接続する必要がある。本研究では巨大な DHT は不要であり、オフラインでも利用可能なものを実装する。

3. 機密実行環境を利用した自己破壊･出現データの設計

3.1 前提とするハードウェアおよびソフトウェア

本研究では、次のようなハードウェアおよびソフトウェアを利用する。

(1) Intel SGX による機密実行環境

Intel Software Guard Extensions (SGX)[1)] は Intel の第 6 世代 Core CPU (2015 年) 以降に搭載されている拡張命令で、enclave とよばれる機密実行環境を提供する。Enclave 内では、その enclave に固有の鍵を利用してデータを暗号化することができる (data sealing)。Data sealing で得られた暗号文は、enclave の外で保管しておくことができる。

(2) Remote Attestation と鍵共有

コードの真正性を別の計算機から検証する仕組みを Remote Attestation (RA) と言う。SGX は Diffie-Hellman (DH) 鍵共有を応用した RA を提供しており、RA と同時に enclave へのセキュアチャネルが構築できる。

(3) ローカルのファイルシステム

暗号化されたデータの保管場所として、データ利用者のローカルのファイルシステムを利用する。

(4) 信頼された時刻サーバ

これは乱数 (nonce) を受け取り、時刻と共に署名して返す。

(5) Intel Converged Security/Management Engine (CSME)

これは、チップセットの内部で動作する小さな計算機であり、遠隔管理やセキュリティ等の機能を提供する。そのうち、本研究ではモノトニックカウンタ機能および、信頼された時刻を提供する機能を利用する。

3.2 目　標

本研究では、次のような自己破壊・出現データを実装する。

- 有効期間や利用回数等の柔軟な出力条件を指定できる。
- 個人の計算機で完結し、オフラインで利用できる。

本研究では、下記の攻撃に対応する。

- リプレイ攻撃
  ユーザが暗号化されたファイルをバックアップし、これを使って enclave を以前の状態に戻そうとする。また、時刻サーバのデータをキャプチャし、これを使って時刻をごまかそうとする。
- Enclave のコードに対する改ざん

3.3 自己破壊・出現データの転送手順

送信者から受信者へデータ M および出力条件 C を送信する。C はデータの取り出しを許可する条件であり、アクセス回数や有効期間の時刻等を設定できる。

データの転送手順を**図 1** に示す。図中の App という

† 筑波大学

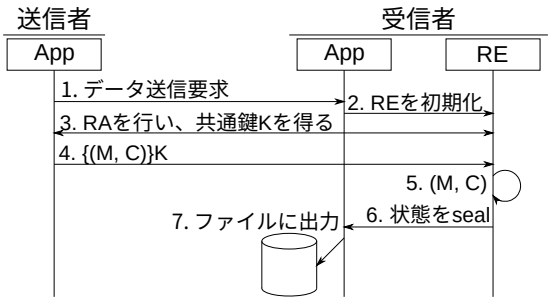

図 1　データ転送手順

表記は通常環境のプログラムを表す。順に説明する。

1. 送信者が受信者にデータ送信の準備を要求する。
2. 受信者の App は機密実行環境 RE (Receiver Enclave) を初期化する。
3. 送信者が RE に対し RA を行う。DH 鍵共有により送信者と RE が共通鍵 K を得る。
4. 送信者は M と C を連結し K で暗号化した暗号文 $\{(M, C)\}_K$ を受信者に送る。
5. M および C は RE 内部で復号される。
6. M, C および内部の状態を data sealing により暗号化する。
7. App は、それをファイルに保存する。

3.4　自己破壊・出現データへのアクセス

受信者は App を通して RE に開封を要求する。RE は C を評価し、それが満たされていれば M を出力する。

C がアクセス回数の場合は、モノトニックカウンタを利用する。時刻の場合、信頼できる時刻サーバが必要になる。サーバは証明書によって信頼され、送信者とは別の主体が用意しても構わない。

時刻を得る手順を以下に示す。

1. RE 内部で乱数 n (nonce) を生成しサーバに送る。
2. サーバは時刻 T と n を連結しデジタル署名を付加したデータを RE に送信する。
3. RE は署名を検証し、n が一致していることを確認する。ここで署名を検証するための公開鍵は、RE に静的なデータとして埋め込まれている。

オフラインでデータを開封するために、M, C に加えてモノトニックカウンタの値も保存しておく。また、時刻の取得には CSME の信頼された時刻を利用する。

3.5　セキュリティ

3.2 節の攻撃を提案手法で防げることを示す。

リプレイ攻撃

ローカルに保存したデータにモノトニックカウンタから取得した値を含めている。それとチップセットから得られた値を比較することで、リプレイ攻撃を検出できる。また時刻サーバのメッセージに関しても、nonce の値からリプレイ攻撃を検出できる。

Enclave のコードに対する改ざん

Enclave のコードが改ざんされた場合、RA が失敗するので、新しくデータを受け取れない。また、enclave に紐付いた鍵の値が変化するため、ローカルファイルシステムに保存されたデータを unseal することもできない。

4. ま と め

本稿では機密実行環境による自己破壊・出現データの実装方式を提案した。自己破壊・出現データは、ある条件が満たされたときに限りアクセスできる仕組みをもったデータである。本方式の特長は、オフラインで利用でき、個人の計算機で完結することである。

現在、提案方式を Rust 言語で実装している。Data sealing を用いて自己破壊データをローカルに保管し、取り出す処理を実装した。今後、時刻の処理と、セキュアチャネルを介したデータの交換を実装する。次に、安全性および開封時間の評価を行う。さらに、CSME を利用したスクリーンショット等からの保護について検討する。

参 考 文 献

1) Intel Software Guard Extensions (Intel SGX), `https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html` (Accessed: 2019-12-02).
2) M. Sabt, M. Achemlal, and A. Bouabdallah., “Trusted Execution Environment: What It Is, and What It Is Not”, IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 57-64, 2015.
3) R. Geambasu, T. Khono, A. Levy, and H. Levy., “Vanish: Increasing Data Privacy with Self-Destructing Data”, in Proc. USENIX Security 09, pp. 299-350, 2009.