高機能暗号の自動適用に向けた暗号プロトコルのステークホルダーと処理フローの整理

金岡 晃[1]

概要：ソフトウェア開発者が高機能暗号を適切に適用可能にすることの難しさが、これまでの研究で明らかになってきている。解決アプローチの１つとして、ソフトウェア開発環境に暗号適用のサポート機能や自動適用機能を付与する方法がある。しかしこれらの研究は現段階では基礎的な暗号だけに対応しており、高機能暗号を含めた多種多様な暗号へ対応するには課題が多く残っている。本論文では、自動適用するための要件を考えるにあたり、既存の暗号技術との違いとして、ステークホルダーと処理フローの差異に着目し、代表的な高機能暗号のステークホルダーと処理フローを整理、分類し、ソフトウェア開発時の暗号技術の自動適用実現を議論する。その結果、十分かつ精査された要件の整理が重要であることが示された。

Organizing the Stakeholders and Process Flow of Cryptographic Protocols for Automatic Application of Advanced Cryptography

Kanaoka Akira[1]

Abstract: Previous research has shown that it is difficult for software developers to properly enable advanced cryptography. One approach is to add support and automatic cryptographic capabilities to the software development environment. However, these studies only deal with basic cryptographic tequniques at the present stage, and many problems remain to cope with various kinds of cryptography including advanced cryptography. In this paper, we focus on the difference between stakeholder and processing flow as the difference between existing cryptography. We also discuss the realization of automatic application of cryptography during software development. As a result, it was shown that it is important to organize the requirements sufficiently and carefully examined.

1. はじめに

暗号技術はさまざまなサービスを安全にするための最も重要な要素の１つであると言える。サービスが高度化するに従い、サービスを開発するソフトウェアやシステム開発者が獲得しなければならない知識は多様化かつ深化している。その時、暗号技術は開発者が獲得すべき知識としては重要であるが必須ではないために、獲得が十分でないままサービスの開発が行われリリースがされることが十分に考えられる。

実際にいくつかの事例において暗号にかかわる実装部分において杜撰な設定や実装をしたサービスやアプリケーションが多く存在していることが確認されており、近年ではそういった開発者が安全なサービスやアプリケーションを開発するための研究をユーザブルセキュリティの視点で行うことが盛んになってきている。

開発者を対象にした暗号ライブラリや API のユーザブルセキュリティ研究では、その対象となっている暗号技術が共通鍵暗号のモード設定や初期値の使いまわしなどの初歩的なものにとどまっており、さらなる公開鍵暗号や TLS に代表される PKI の利用についての研究はまだ黎明期であると言える。さらには、現在広く使われている共通鍵暗号や公開鍵暗号だけではなく、理論的に成熟を迎え社会への展開が期待されるさまざまな高機能な暗号技術についてはほとんど研究されていない。

多くの研究が初歩的な暗号に関する開発者補助に挑んで

[1] 東邦大学
Toho University

いることは、開発者の多くがその段階で多くの問題を引き起こしていることの裏付けでもある。今後の高機能な暗号技術の展開を考えた場合、開発者の多くにとって高機能な暗号技術の技術的内容の理解は困難であり、適切な利用は現状よりもさらに困難になることが容易に考えられる。ID情報を公開鍵として利用可能なIDベース暗号や、利用者の属性情報をもとに暗号化や復号が可能な属性ベース暗号、検索を安全に実施可能な検索可能暗号、データを秘匿したまま様々な演算が可能な秘密計算など、多くの応用が期待される高機能な暗号は、今後の社会においては高いニーズがある技術である一方で、その理論的背景は技術により異なり、理解の困難さを高める要因にもなっている。そのため、開発者が暗号技術を簡便かつ適切に利用するため補助技術や自動化技術の研究開発はより加速する必要がある。しかし現在ではそういった多様かつ高機能な暗号技術適用に関する補助に対しては、要件の議論さえされていない段階である。

そこで本論文では、開発者が高機能な暗号技術を簡便かつ適切に利用するため補助技術や自動化技術の研究開発に向けた要件整理を目的とする。暗号プロトコル間のステークホルダーと処理フローの違いに着目し、代表的な暗号プロトコル技術のステークホルダーと処理フローの整理を行い、ステークホルダーと処理フローを考慮した補助あるいは自動化技術が必要であることを示す。

2. 関連研究

2.1 暗号 API の誤使用

2.1.1 誤使用の調査

2013年、Egeleらにより、暗号APIの適切利用がソフトウェア開発者にとって容易ではないことが議論された[1]。EgeleらはAndroidアプリケーションにおける暗号APIの誤使用があることに着目し、その誤使用の解析を行う軽量手法を提案した。そこで解析対象とされている項目は、共通鍵暗号におけるECB利用の有無やCBCモードにおける非ランダムな初期ベクトル（IV）利用の有無など共通鍵のモードに関する話や、固定の暗号鍵利用、パスワードにおけるSalt値の固定利用、ストレッチング（繰り返し）の回数の少なさ、乱数生成時の関数利用での適切なSeed値の利用であった。

Egeleらの研究は、暗号技術を研究するものにとっては初歩的とも思えるものである。しかし一方でそれらのチェックが有効に効くことが示されており、逆説的に実際のアプリケーション開発の現場における誤使用の発生ポイントと暗号研究者の認識とのギャップを強く示すものになったと言える。なお、これらのチェックリストで最も多く該当したものが、ECBモードの利用であった。これは調査時においてBouncyCastleでdefaultになっていたことが原因である可能性が指摘されている。

その後、2014年になりLazarによる調査や[2]、Liらによる暗号APIの追跡手法の提案があった[3]。Lazarらによる調査は、2011年から2014年にかけて明らかになった暗号関連の脆弱性269件を調査し、その結果83%が暗号の誤使用であることが明らかにされた。Liらは、iOSのアプリケーションをターゲットにし、静的解析と動的解析を組み合わせて暗号APIの利用を追跡するiCryptoTracerを提案した。そして提案システムにより調査対象の98のアプリのうち64のアプリで暗号の誤使用を発見した。それらの誤使用のターゲットは固定の鍵の利用、CBC用に非ランダムなIVが使われているか、そしてStatelessな暗号が利用されていないか、というものであり、Egeleらの研究と同様、暗号技術の種類としては広範に利用されている暗号の一部分がターゲットになっていた。

2015年に発表されたChatzikonstantinouらの論文は、暗号の誤使用の調査ターゲットを大きく広げた[4]。まずターゲットを大きく「Weak Cryptography (C)」「Weak Implementation （I)」「Weak Key (K)」「Weak Cryptographic Parameters (P)」と4つに分類した。そしてそれぞれの分類でさらに項目が細分化されている。分類Cでは6項目、Iで6項目、Kで4項目、Pで8項目となっている。これらの項目はEgeleらが挙げた項目を包含するものとなっている。新たに取り上げられた項目の一例をあげると、分類CではMD5といったアルゴリズムの利用や暗号学的に安全ではない疑似乱数発生器の利用、分類Iでは標準アルゴリズムの再実装やOAEPではないRSAの利用などが入っている。Kでは鍵サイズやハードコードされた鍵についての項目があり、Pでは主に共通鍵のモードやIVに関する項目が並んでいる。

2.2 誤使用の原因把握

なぜ開発者がそういった誤使用をしてしまうかという人間的な視点で調査が複数行われている。

Acarらは、まずポジションペーパーとして断片的に語られてきた開発者が誤使用する原因について、統合的に扱うべく調査が必要だと示した[5]。Acarらはそのポジションペーパーに先立ち、IEEE SPでAndroidアプリケーションを脆弱に作ってしまう原因を探るべく、開発者が開発時の得る情報源に着目し、異なる情報源を与えた場合での脆弱性の発生について調査をしていた[6]。この論文は暗号の誤使用だけに着目したものではなく、Androidアプリケーションでソースコードに起因する脆弱性について調査したものであった。その結果、多くの開発者はオープンな開発者フォーラムであるStackOverflowを利用した場合には動作するアプリケーションが作成可能であるが脆弱なアプリケーションになる傾向があり、公式なドキュメントを参考に開発をした開発者は脆弱なアプリケーションは作らないもののきちんとした動作ができないアプリケーションにな

る傾向があることが示され、公式なドキュメントのユーザビリティ向上が必要であることを主張した。

その他にもいくつか存在するが、ここでは参考文献にとどめるまでとする [7], [8], [9], [10], [11], [12], **?**。

2.3 誤使用への対策

誤使用の原因把握と並行するように、対策手法の研究も進みつつある。

暗号 API の誤使用に対して修正を図るアプローチは、2015 年に Arzt らによって Eclipse のプラグインとして提案がされた [13]。対象言語を Java に限定し、開発中のソースコードの分析を行うアプローチであった。当初は OpenCCE とプロジェクト名を付けていたが、その後 CogniCrypt とプロジェクトが変わり、現在でも開発が行われている [14]。Arzt らの論文では提案だけにとどまっており、実際の評価までは至っていなかった。

2016 年になり、Ma らが CDRep を提案した [15]。Arzt らと同じく、暗号の誤用を自動修復するツールであり、Android や iOS のアプリケーションを対象にしている。修正パターンはヒューリスティックに準備がされており、誤用の分類は 7 種類としていた。この分類は Egele らの研究を踏襲するものであり、共通鍵の利用がが主なターゲットとなっている。ツールの評価では、1262 件のうち 95%の修復に成功したとしている。

暗号 API の誤使用だけではなく、より広く脆弱性全般もターゲットにした動きとして、Nguyen らの研究がある [16]。こちらも統合開発環境にプラグインを導入することで実現するものであり、開発者が陥りがちな 13 の項目を調査対象としている。特徴的なこととして、教育視点としてチュートリアルのようなテストプロジェクトが用意されている。

2.4 高機能暗号の応用

高機能暗号のうち、暗号を応用したシステムを開発し、そのユーザビリティを測る研究も進んできている。

Ruoti らはユーザブルな電子メール暗号化のための Web メールシステムとして Pwm を提案しそのユーザビリティを議論しているが、暗号化に用いられたプロトコルは ID ベース暗号であった [17], [18]。

緑川らは Web メールシステムに対称型検索可能暗号を適用し、そのユーザビリティを議論した [19]。Qin らは秘密計算（マルチパーティ計算）を Web アプリケーションとして実現し、そのユーザビリティや信頼を議論した [20]。

これらのように、高機能暗号は理論の段階から実装を経て、社会への配置を議論する段階に至ってきた。

3. 暗号プロトコルのステークホルダー整理

3.1 整理の必要性

RSA 暗号のプロトコルは、以下の 3 つの作業により構成される。

(1) KeyGen
- 正の整数 e の選択
- 素数 p, q の選択
- $n = pq$ の計算
- $d = e^{-1} \mathrm{mod}(p-1)(q-1)$ の計算

(2) Encryption
- メッセージ m の準備
- $c = m^e \mathrm{mod} n$ の計算

(3) Decryption
- $m = c^d \mathrm{mod} n$ の計算

暗号プロトコルを見ると、KeyGen と Encryption が 1 人のステークホルダー、Decryption がもう 1 人のステークホルダーという 2 人のステークホルダーで成立することがわかる。実運用上では RSA の鍵生成（KeyGen）は CA を含むサーバ側が行うこともあるが、原理としては他のステークホルダーがする必要性はなく、Encryption を行うステークホルダーが実施可能である。そのため、本章におけるステークホルダーの整理では、暗号プロトコルの意味上で最小のステークホルダー数で暗号プロトコルが実行されることを考えて整理を行う。RSA 暗号のケースでは KeyGen と Encryption で 1 人のステークホルダーとする。この時に注意することとして、生成された公開鍵 e, n に対し信頼できる機関である CA（認証局）が電子署名を施すことで公開鍵証明書を発行する PKI の仕組みは暗号プロトコルに含まれていないため、CA は暗号プロトコルのステークホルダーになっていない。

対して、ID ベース暗号の 1 つである Boneh-Franklin 方式は、以下の 4 つに作業により構成される。

(1) Setup
- 位数 q の G_1, G_2, ペアリング関数 $e : G_1 \times G_1 \to G_2$ を準備
- ハッシュ関数 $H_1 : \{0,1\}^* \to {G_1}^*$、$H_2 : G_2 \to \{0,1\}^n$、を準備
- ランダムな生成元 $P \in G_1$ の選択
- ランダムな元 $s \in {Z_q}^*$ を選択し、$P_{pub} = sP$ を計算

(2) Extract
- $ID \in \{0,1\}^*$ から $Q_{ID} = H_1(ID) \in {G_1}^*$ を計算
- プライベート鍵 $d_{ID} = sQ_{ID}$ を計算

(3) Encrypt
- メッセージ M の準備
- $g_{ID} = e(Q_{ID}, P_{pub}) \in {G_2}^*$ を計算
- ランダムな元 $r \in {Z_q}^*$ を選択

- 暗号文 $C =< rP, M \oplus H_2(g_{ID}{}^r) >$ を作成

(4) Decrypt

- 受け取った暗号文 $C =< U, V >$ に対し $V \oplus H_2(e(d_{ID}, U)) = M$ を計算

RSA 暗号と異なり、ID ベース暗号ではそのステークホルダーは 3 者になる。Encrypt を行うステークホルダーと Decrypt を行うステークホルダのほかに、Setup と Extract を受け持つステークホルダーが必要となる。RSA 暗号における KeyGen は ID ベース暗号における Setup と Extract に対応するが、RSA 暗号と同じようにこれらを Encrypt と合わせて 1 人のステークホルダーとすると、暗号プロトコルの意味が崩れる。ID ベース暗号では復号鍵生成をする Extract ではマスターシークレットと呼ばれる s を利用し、それは Encrypt や Decrypt をするユーザーには非開示とするものである。よって、別のステークホルダーとして存在しなければならない。ID ベース暗号ではこのステークホルダーを鍵生成局（Key Generation Center、KGC）と呼んでいる。

現在広く世の中で利用されている公開鍵暗号である RSA 暗号や RSA 署名、DH 鍵交換、ECDH 鍵交換、ECDSA 署名などはすべて 2 者のステークホルダーのモデルである。さらに共通鍵暗号である AES も 2 者のステークホルダーのモデルである。

Ma らによる CDRep[15] や Nguyen らの提案システム [16] では、開発環境での暗号利用サポートとしては共通鍵暗号だけが対応となっている。つまり、前提としている暗号技術は 2 者ステークホルダーのモデルであると言える。一方で、高機能暗号の中には 3 者ステークホルダーのモデルが存在することから、2 者以外のステークホルダーモデルを前提とした支援システムや自動化システムが必要となることがわかる。そしてこういった高機能暗号に対応させるには、支援や自動化対象の暗号プロトコルがどういったステークホルダーを持つかを整理する必要があることがわかる。

3.2 ステークホルダーの整理

※整理の方法

整理の結果を以下に示す。

- 1 者モデル：SSE
- 2 者モデル：RSA 暗号、RSA 署名、ECDSA、PEKS、BLS 署名、Ateniese プロキシ再暗号化、Elgamal 暗号、Elgamal 署名、Fiat-Shamir 署名、DH 鍵交換、ECDH 鍵交換、マルチパーティ計算
- 3 者モデル：BF-IBE、BB1-IBE、Waters-IBE、KP-ABE、CP-ABE、Hess-IBS、Boneh-BE、RSA ブラインド署名、グループ署名、LSAG リング署名、秘密分散、マルチパーティ計算

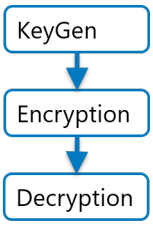

図 1　RSA 暗号の処理フロー

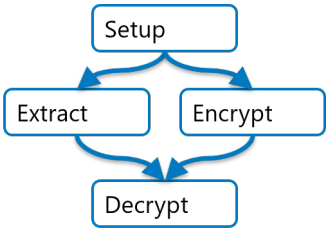

図 2　IBE の処理フロー

4. 暗号プロトコルの処理フロー整理

4.1 整理の必要性

暗号プロトコルを構成する各作業の処理フローを考える。RSA 暗号の場合、Decryption は Encryption が終わった後でしか実行し得ず、Encryption は KeyGen が実行された後でしか実行し得ない。つまり図 1 のような処理フローになる。

RSA 暗号のフローは、AES 等の共通鍵暗号でも同様な形となる。

次に ID ベース暗号の処理フローを考える。ID ベース暗号の場合、Decrypt は Encrypt が終わった後でしか実行しえないが、Encryption は Setup が終わった後であれば Extract が実行される前でも実行が可能である。より正確に言えば、Decrypt は Extract と Encrypt が終わった後でしか実行しえず、Extract と Encrypt は Setup が終わった後でしか実行し得ない、となる。これを図示すると図 2 のようになり、処理のフローとして Extract と Encrypt の実行タイミングは順序がどちらが前後であろうと問題ないものとなっていることがわかる。

このように、暗号プロトコルでフローが大きくことなることがわかる。ステークホルダーと同様に、既存の暗号適用を支援する開発環境での暗号利用サポートとしては共通鍵暗号だけが対応となっているため RSA 暗号の処理フローと同様の直線的なフローだけが対応可能と言っていい。一方で、直線的ではない複雑なフローが存在することから複数の形状を持ったフローが存在することを前提とした支援システムや自動化システムが必要となることがわかる。そしてこういった暗号に対応させるには、支援や自動化対象の暗号プロトコルがどういった処理フローを持つかを整理する必要があることがわかる。

4.2 処理フローの整理

代表的な暗号プロトコルの処理フローを図 3,4,5,6,7 に示す。それぞれ大きく形が異なっており、ステークホルダーのような単純な分類が難しいことがわかる。

これらの結果から、いくつか興味深いフローが見られることがわかる。ECDSA は RSA 等と同じく直線的なフローとなっている。RSA 同様に広く利用されている DH 鍵交換はフローが直線的ではないことがわかる。PEKS の処理フローでは、Test（検索）の実施においては、PEKS（検索対象の登録）は必ずしも必要としていないことがわかる。また Hess-IBS の処理フローでは、BF-IBE と同じく ID ベース暗号系でありながらも既存の電子署名である ECDSA と同様の直線的フローとなっていることがわかる。

さらに Ateniese-PRE は、処理フローを概観するだけでその複雑さがわかる。暗号技術に精通していないソフトウェア開発者がこういった暗号を適切に利用することの難しさはこのフロー図だけ見てもわかると言えよう。

5. 考察と今後の課題

3 章と 4 章で示したように、ステークホルダーと処理フローは、暗号技術に焦点を当てた開発者向けのユーザブルセキュリティ研究で扱っている共通鍵暗号とは大きくことなることが明らかになった。実際にこれらを統合開発環境に適用して補助ないし自動化を行うためには、異なるアプローチをとる必要性も考えられる。

たとえば、プログラム開発において新たなインスタンスを定義した場合にそのインスタンスの用途をヒアリングするなどのインタラクティブ機構を入れることでどの暗号技術を適用すべきかの材料につかうことや、利用 API の特性と合わせて考えることで推奨暗号プロトコルを変えるといった知識ベースの導入などが考えられる。いずれのアプローチも網羅的に行うことの難しさはあるため、当初は特定暗号技術に焦点を当てた補助技術の研究開発を行い、そこで得た知見をフィードバックしつつ網羅性を高める手法が良いと考えられる。

開発者向けのユーザブルセキュリティ研究であるため、ユーザ実験は欠かせないが、適切な実験を行うためには丁寧な設計が必要となる。開発者の知識レベルを問う事前ヒアリングや、開発等を行うタスクの設定、タスク実施時の実験協力者の観察方法、タスク終了後のインタビュー、得られたデータの分析手法など、おそらくこれまでの研究とは異なるアプローチも必要になることが考えられる。その点の検討も深めていかなければならない。

6. まとめ

開発者が高機能な暗号技術を簡便かつ適切に利用するため補助技術や自動化技術の研究開発に向けた要件整理を目的とし、暗号プロトコル間のステークホルダーと処理フローの違いに着目し代表的な暗号プロトコル技術のステークホルダーと処理フローの整理を行った。その結果、ステークホルダーの数は暗号技術により異なり、既存の開発者補助研究は直接適用が難しいことが明らかになった。処理フローについてはステークホルダー以上に複雑な違いがあることが明らかにされ、要件の整理等が重要であることがあらためて明確になった。またそれらを踏まえた将来的な研究アプローチについても考察した。

参考文献

[1] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. 2013. An empirical study of cryptographic misuse in android applications. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13). ACM, New York, NY, USA, 73-84. DOI: http://dx.doi.org/10.1145/2508859.2516693

[2] David Lazar, Haogang Chen, Xi Wang, and Nickolai Zeldovich. 2014. Why does cryptographic software fail?: a case study and open problems. In Proceedings of 5th Asia-Pacific Workshop on Systems (APSys '14). ACM, New York, NY, USA, , Article 7 , 7 pages. DOI=http://dx.doi.org/10.1145/2637166.2637237

[3] Li Y., Zhang Y., Li J., Gu D. (2014) iCryptoTracer: Dynamic Analysis on Misuse of Cryptography Functions in iOS Applications. In: Au M.H., Carminati B., Kuo CC.J. (eds) Network and System Security. NSS 2015. Lecture Notes in Computer Science, vol 8792. Springer, Cham

[4] Alexia Chatzikonstantinou, Christoforos Ntantogian, Georgios Karopoulos, and Christos Xenakis. 2016. Evaluation of Cryptography Usage in Android Applications. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) (BICT'15), Junichi Suzuki, Tadashi Nakano, and Henry Hess (Eds.). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 83-90. DOI: http://dx.doi.org/10.4108/eai.3-12-2015.2262471

[5] Y. Acar, S. Fahl and M. L. Mazurek, "You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users," 2016 IEEE Cybersecurity Development (SecDev), Boston, MA, 2016, pp. 3-8.

[6] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek and C. Stransky, "You Get Where You're Looking for: The Impact of Information Sources on Code Security," 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2016, pp. 289-305.

[7] Soumya Indela, Mukul Kulkarni, Kartik Nayak, and Tudor Dumitraş. 2016. Helping Johnny encrypt: toward semantic interfaces for cryptographic frameworks. In Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2016). ACM, New York, NY, USA, 180-196. DOI: https://doi.org/10.1145/2986012.2986024

[8] S. Indela, M. Kulkarni, K. Nayak and T. Dumitraş, "Toward Semantic Cryptography APIs," 2016 IEEE Cybersecurity Development (SecDev), Boston, MA, 2016, pp. 9-14. doi: 10.1109/SecDev.2016.014

[9] S. Nadi, S. Krüger, M. Mezini and E. Bodden, ""Jump-

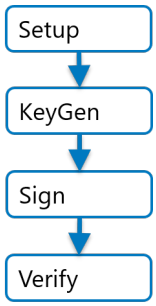

図 3　ECDSA の処理フロー

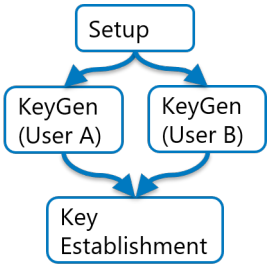

図 4　DH 鍵交換（ECDH）の処理フロー

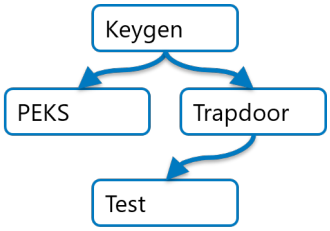

図 5　検索可能暗号（PEKS）の処理フロー

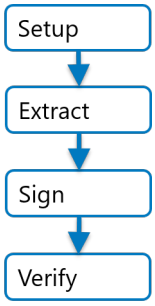

図 6　ID ベース署名（Hess-IBS）の処理フロー

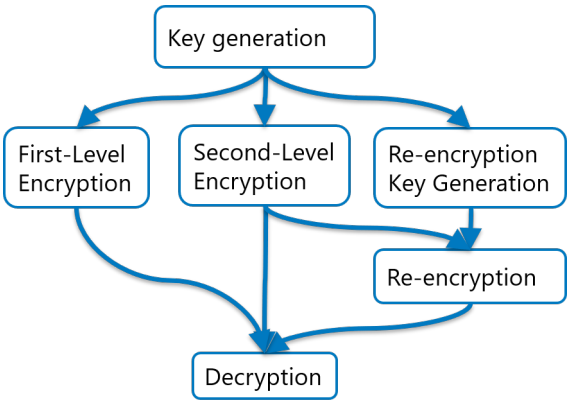

図 7　プロキシ再暗号化（Atenisese-PRE）の処理フロー

ing Through Hoops": Why do Java Developers Struggle with Cryptography APIs?," 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, 2016, pp. 935-946.

[10] Y. Acar et al., "Comparing the Usability of Cryptographic APIs," 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2017, pp. 154-171. doi: 10.1109/SP.2017.52

[11] Ksenia Ermoshina, Harry Halpin, Francesca Musiani: Can Johnny Build a Protocol? Co-ordinating developer and user intentions for privacy-enhanced secure messaging protocols , EuroUSEC, 2017

[12] Ukrop M., Matyas V. (2018) Why Johnny the Developer Can't Work with Public Key Certificates. In: Smart N. (eds) Topics in Cryptology – CT-RSA 2018. CT-RSA 2018. Lecture Notes in Computer Science, vol 10808. Springer, Cham

[13] Steven Arzt, Sarah Nadi, Karim Ali, Eric Bodden, Sebastian Erdweg, and Mira Mezini. 2015. Towards secure integration of cryptographic software. In 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!) (Onward! 2015). ACM, New York, NY, USA, 1-13. DOI: http://dx.doi.org/10.1145/2814228.2814229

[14] Stefan Krüger, Sarah Nadi, Michael Reif, Karim Ali, Mira Mezini, Eric Bodden, Florian Göpfert, Felix Günther, Christian Weinert, Daniel Demmler, and Ram Kamath. 2017. CogniCrypt: supporting developers in using cryptography. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017). IEEE Press, Piscataway, NJ, USA, 931-936.

[15] Siqi Ma, David Lo, Teng Li, and Robert H. Deng. 2016. CDRep: Automatic Repair of Cryptographic Misuses in Android Applications. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16). ACM, New York, NY, USA, 711-722. DOI: https://doi.org/10.1145/2897845.2897896

[16] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. 2017. A Stitch in Time: Supporting Android Developers in WritingSecure Code. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). ACM, New York, NY, USA, 1065-1077. DOI: https://doi.org/10.1145/3133956.3133977

[17] Scott Ruoti, Nathan Kim, Ben Burgon, Timothy van der Horst, Kent Seamons: Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes, Symposium On Usable Privacy and Security(SOUPS)(2013).

[18] Scott Ruoti, Jeff Andersen, Travis Hendershot, Daniel Zappala, and Kent Seamons. 2016. Private Webmail 2.0: Simple and Easy-to-Use Secure Email. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)

[19] T. Midorikawa, A. Tachikawa and A. Kanaoka, "Helping Johnny to Search: Encrypted Search on Webmail System," 2018 13th Asia Joint Conference on Information Security (AsiaJCIS), Guilin, 2018, pp. 47-53.

[20] Lucy Qin and Andrei Lapets and Frederick Jansen and Peter Flockhart and Kinan Dak Albab and Ira Globus-Harris and Shannon Roberts and Mayank Varia, "From Usability to Secure Computing and Back Again", Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019), Santa Clara, 2019