形式的安全性検証ツールを用いた暗号教育の実践とそのe-Learning教材化の課題について

岡崎 裕之　紫村 彰吾　宮本 樹　渡邊 樹　布田 裕一　村上 恭通

暗号技術は情報セキュリティを実現するための基盤要素である．暗号を専門とする研究者や技術者のみならずネットワークエンジニアや運用者等にとっても暗号技術に関する知識の習得は必須である．実際のシステムは，公開鍵暗号や電子署名等の複数の要素技術を組み合わせて構築する必要があるため，様々な攻撃について必要な対策を施しつつ安全性要件を満たすように実現しなければならない．しかしながら，システム構築を座学だけでは初学者に学ばせることは困難である．この課題を克服するために，報告者等は計算機援用による形式的暗号プロトコル安全性検証ツールを利用した暗号技術の基礎知識と利用方法を学習する教材を作成し，演習形式による授業を実践した．形式的暗号プロトコル安全性検証ツールの利用により，既存の教材では学習困難な実際の暗号技術の動作や攻撃を，学習者が設定した暗号システム上でシミュレーションしてインタラクティブに学ぶことが可能となった．本論文では，まず，実践した教育内容の概要とその成果を報告し，更に，現在進めている本教材を元とする CAI 教材開発方針の紹介と，その課題について報告する．

Cryptographic technology is a fundamental element used to realize information security. Learning about cryptography and training in cryptographic techniques are processes that are indispensable to not only researchers and engineers specialized in cryptography but also to information and communications technology (ICT) engineers, such as network engineers and operators. However, the educational environment and teaching materials on the topic of cryptographic technology for ICT engineers are not sufficient to provide practical knowledge. There are also not enough teachers who can teach the theoretical foundations as well as the technological applications of cryptography. In this report, we present the trial performed to teach the basics of cryptographic technology using ProVerif, one of the most successful automatic cryptographic protocol verifiers. Moreover, we propose a plan to develop an e-learning system for topics in cryptographic technology using ProVerif.

1　はじめに

我が国の情報セキュリティ向上は急務であるが，それを担う情報セキュリティ人材の不足が叫ばれている．情報セキュリティはウィルスやネットワーク不正侵入，情報漏洩の防御等の様々な分野での対策が必要であるが，暗号技術はそれらの対策を実現する基盤技術に位置づけられる．さらに暗号技術は，フィンテックや IoT 等の次代の産業を興すための要素技術としても期待されている．しかし，暗号の専門家以外のネットワーク技術者等の暗号技術を利用する立場の ICT 技術者には十分に暗号技術に関する知識が共有されているとはいい難い．例えば，インターネットの標準暗号プロトコルである SSL/TLS の実装において，POODLE，Heartbleed 等のセキュリティホールが相次いで発見されたが，これらのほとんどは実装した者が暗号技術を正しく理解していなかったことに起因する．暗号学の成り立ちが計算機科学，数学，通信工学等の様々な研究分野の境界領域上にあるた

Educating Cryptography using Formal Security Verification tool for Cryptographic Protocols.

Hiroyuki Okazaki, Shogo Shimura, 信州大学, Shinshu University.

Tatsuki Miyamoto, Tatsuki Watanabe, Yasuyuki Murakami, 大阪電気通信大学, Osaka Electro-Communication University.

Yuichi Futa, 東京工科大学, Tokyo University of Technology.

コンピュータソフトウェア, Vol.37, No.1 (2020), pp.99–113.

[研究論文] 2019 年 4 月 19 日受付.

本論文は 第 5 回 実践的 IT 教育シンポジウム rePiT2019 in 愛媛 2019 の発表論文をもとに発展させたものである．

めに，暗号技術の技術文書が非常に難解なテクニカルタームを利用したり，学術的，あるいは技術的に厳格な書式を取ったりして難解に記載されていることが一因である．さらにセキュリティ技術の特性上，攻撃者は取り得る限りの攻撃手法を様々な組み合わせで試みるため，たとえ学習者にわかりやすい暗号技術の入門書が執筆されたとしても座学での暗号技術の教育を行うことは暗号の専門家以外にとっては困難であろう．この問題に対し，一般の ICT 技術者への説明責任を果たしていないということが，暗号研究者の間でしばしば議論されるようになっている．

現状のような難解な表現を用いずに暗号研究者以外にも理解しやすく，かつ厳密性を失わずに正しく説明を行うためには何らかの革新的な表現方法を用いる必要があり，形式検証技術はその有力な表現方法の一つとして期待されている．暗号学において，形式検証技術は主に計算機援用による安全性検証に利用されている．形式的安全性検証では暗号に関する概念や定理の証明のライブラリ化，あるいは要素技術や安全性要件のモデル化を形式記述言語を用いて研究者が行い，計算機を用いて安全性を検証する．形式化されたモデルは形式化記述言語の文法さえ習得すれば理解しやすく，暗号技術の概念を説明することも比較的容易であり，一度ライブラリ化，モデル化された技術要素は誰もが再利用可能である．そのため，形式化されたモデルや技術要素は，安全性検証のみならず，暗号に関する専門知識を有しない ICT 技術者や情報系の学生を対象とした，暗号技術のドキュメント化や CAI 教材としても利用することが可能となる．特にモデル探索による形式的暗号プロトコル安全性検証ツールでは，与えられた形式モデル上で，攻撃者が行える処理を組み合わせることで，攻撃が成功する手順を網羅的に探索し，もし攻撃が発見されれば具体的にその攻撃方法を表示する．

そこで報告者等は，計算機援用による形式的暗号プロトコル安全性検証ツール ProVerif[3] を利用した暗号技術の基礎知識，利用方法を学習する演習形式での授業を行った．形式的暗号プロトコル安全性検証ツールの利用により，既存の教材では実現しえない実際の暗号技術の動作や攻撃を，学習者が設定した暗号システム上でシミュレーションしてインタラクティブに学べる，より学習効果の高い演習を行えた．本報告では報告者等が信州大学工学部電子情報システム工学科 3 年生を対象に行った形式的暗号プロトコル安全性検証ツールの利用による暗号技術に関する演習授業の紹介とその評価を行い，さらに現在進めている本演習の Moodle 上での CAI 教材化方法の概要説明と，その課題について報告する．

2 背景

本研究では，計算機援用による形式的暗号プロトコル安全性検証ツールを利用した暗号技術の基礎知識，利用方法を学習する演習授業の実践と，その成果を基にした CAI 教材の開発を行っている．既存の教材や解説書のほとんどは，暗号機能やその理論を対象としており，暗号研究者や暗号技術者向けに書かれている．そのため，暗号を利用する ICT 技術者の大半を占めるネットワーク技術者や，暗号技術を利用したシステムやアプリケーション開発者等の暗号利用者，あるいはそれらの技術者を志す学生を対象としたものは少ない．ICT 技術者にとっては個々の暗号技術 (ハッシュ関数，公開鍵暗号，電子署名等) の基礎理論は必要なく，それらの適切な使い方を習得すれば十分である．学習者がこのような暗号の利用方法を学ぶ際には，目的 (データの秘匿，相手認証，改竄検知等やその組み合わせ) を実現するために，暗号技術を用いたシステムやアプリケーションを実現するための暗号プロトコルを実際に設計し，攻撃をシミュレーションすることが効果的であろう．しかしながら，学習者が設計した暗号プロトコルの安全性検証，即ち暗号プロトコルに対して攻撃が可能であるか否かを判断し，もし攻撃が可能であればその解決方法を指摘するために理論的な検証が必要であり，この検証は専門家以外には困難であろう．さらに，たとえ指導者が暗号の専門家であったとしても，多人数を対象とした演習授業において全ての学習者の設計した暗号プロトコルを評価し，指導を行うことは困難である．そこで，形式的暗号プロトコル安全性検証ツールを適切に利用すれば，どのようなプロトコルを設計しても，もし攻撃が成功するのであれば具体的な攻撃手順を導出す

るため直観的に理解しやすく，さらに学習者自身で暗号プロトコルの安全性検証を行えるので，指導者は形式的暗号プロトコル安全性検証ツールが導出した攻撃に対する解決策について学習者に解説すればよく，多人数対象の演習を行うことも可能となる．

3 ProVerif を用いた暗号教育のねらい

本章では形式的暗号プロトコル安全性検証ツール ProVerif での安全性検証の特徴や暗号初学者が学習時に陥り易い問題点を踏まえたうえで，本演習の狙いと期待される学習時のメリットについて述べる．実際の暗号プロトコルは目的とする安全性要件を満足するように，秘密鍵暗号，暗号学的ハッシュ関数，電子署名等の要素技術を組み合わせて設計する．設計した暗号プロトコルを実装する際には用途に応じて必要となる安全性強度を満たすようにそれぞれの要素技術を選択する．この際，各要素技術は CRYPTOREC 暗号リスト [6](電子政府推奨暗号リスト) 等の安全性と性能の評価が行われている規格に準じた暗号スィーツを用いることが望ましい．例えば，CRYPTOREC 暗号リストでは秘密鍵暗号として AES，Camellia，暗号学的ハッシュ関数として SHA–256 以上，電子署名 DSA，ECDSA 等がそれぞれ推奨されている．ProVerif での安全性検証は前述のような具体的な暗号スィーツを想定した安全性の定量的な評価ではなく，それぞれの要素技術は理想的な暗号技術としてモデル化して形式的に検証を行う．個々の要素技術が理想的，即ち安全であったならば，それらを用いて構成される暗号プロトコルもまた安全であるように思いがちであるが，実際にはたとえ十分な安全性強度を満たした暗号スィーツを利用していたとしても，利用方法や組み合わせ方が不適切であれば暗号プロトコルは攻撃を許してしまう．

そこで，本演習ではまず ProVerif による暗号要素技術の形式モデルを用いて，秘密鍵暗号，暗号学的ハッシュ関数，電子署名等の各要素技術の概念，性質や達成すべき安全性要件を学習する．さらに各要素技術の形式モデルを組み合わせることにより暗号プロトコルの形式記述を行い，暗号技術の適切な組み合わせ方や利用方法，さらに実用的な暗号プロトコルについて学習する．本章では公開鍵暗号の形式モデル化，公開鍵暗号を利用した単純な暗号通信プロトコルを例として ProVerif による形式化および，ProVerif を用いた演習のメリットとそのねらいを紹介する．さらに 4 章で実際に行った演習の詳細な報告を行う．なお，ProVerif については付録 A 章を参照されたい．

3.1 先行研究との比較

形式検証技術を利用した暗号を含む情報セキュリティ分野の教育実践や教材開発は盛んに研究されている [16]．本研究と同じ形式的暗号プロトコル安全性検証ツール ProVerif を利用した教材に限っても既にいくつかの教育実践がなされている [2] [5] [11]．しかしながら，これらは，情報セキュリティや暗号技術の専門家を対象とした教育コースとして開発されているため，形式的手法を利用した安全性検証技術の学習を主な目的としている．一方，本研究では，暗号技術の専門家ではなく他の分野を専門とする ICT 技術者を対象とした暗号技術に関する基礎知識の教育を目的としていることが大きな特徴である．

今後ますます需要が増えるネットワーク技術者や IoT 技術者には，暗号技術に関する高度な知識ではなく，暗号技術の専門家やセキュリティベンダから提供される暗号モジュール等を正しく利用するための知識や脆弱性情報の報告を理解するに足る知識が求められる．本研究ではそのような暗号技術を理想的なモデル，あるいはブラックボックスと見なして，これらの暗号技術の利用方法を受講者が正しく対話的に学ぶための手段として，形式検証技術を利用した．

3.2 各要素技術の学習

ProVerif では安全性検証対象の暗号プロトコルを宣言部，および実行部に分けて形式化する．秘密鍵暗号，暗号学的ハッシュ関数，電子署名等の要素技術の形式モデルは宣言部において形式定義し，実行部ではそれらの形式モデルを利用して暗号プロトコルをモデル化する．宣言部での要素技術の形式モデルは関数の宣言とその理想的な入出力関係を与える等価式として形式的に定義される．例えば公開鍵暗号の場合，暗号化，復号の具体的な計算方法ではなく，ある

公開鍵を用いて暗号化した暗号文は，その公開鍵と対となる秘密鍵を用いることによってのみ復号されるといった公開鍵暗号方式の概念を項の変換ルールとして形式モデル化するのである．なお，具体的な暗号方式や，暗号理論に必要な数学等については本演習終了後に，別の科目で講義を行っている．以下に宣言部での公開鍵暗号の形式定義例を明記する †1．

```
1. (*begin 公開鍵暗号形式モデル*)
2. fun pk(skey):pkey.
3. fun Penc(bitstring, pkey):bitstring.
4. fun Pdec(bitstring, skey):bitstring.
5. equation forall x:bitstring, y:skey;
6.     Pdec(Penc(x, pk(y)), y) = x.
7. (*end 公開鍵暗号形式モデル*)
```

ProVerif は検証ツールであるため，実行形式を記述するものではないが，上記のようなコードで論理的なルールを定義するために何らかのプログラミング言語の既習者にとっては形式モデル，さらにはその元となる要素技術の概念を自然言語による解説よりも誤解の余地なく理解しやすいといえるであろう．

本演習では宣言部での形式モデルは教員があらかじめ定義を行い，公開鍵暗号の他にも暗号学的ハッシュ関数等の基本となる暗号技術について講義形式での解説を行った．理論的な解説を行うとともに，履修者共々 ProVerif での検証を実行することで暗号技術の利用方法や達成できる安全性要件，あるいは単体ではできないこと，即ち攻撃を許してしまうこと等を確かめながら学習を進めることが可能となる．座学や書籍のみでの学習で陥ってしまいやすい勘違いや理解不足があった場合には，ProVerif での安全性検証を実行することで攻撃の具体的な手順を導出されるため，履修者自身が個々に理解度を確認することが本演習のメリットの一つである．

3.3 暗号プロトコル設計の学習

3.2 節では暗号技術の形式モデル化についての概説を行った．暗号プロトコルの形式化は形式モデル化された暗号技術を利用して，通信路を介したプロセス間の通信手順として形式化する．以下は 3.2 節で定義した公開鍵暗号モデルを利用した単純な暗号通信プロトコルの形式化例である．送信者プロセス Sender と受信者プロセス Receiver の形式定義を行い，17 行目で Sender と Receiver を並列実行させるというものである．

```
1. (*begin 公開鍵暗号プロトコル*)
2. (*イベント宣言*)
3. event REC.
4. (*送信者プロセスの定義*)
5. let Sender =
6.   in (c, pkx:pkey);
7.    let ctxt = Penc(m, pkx) in
8.      out(c, ctxt).
9. (*受信者プロセスの定義*)
10. let Receiver =
11. new wsk:skey;
12.   let pkk = pk(wsk) in out(c, pkk);
13.   in (c, rc:bitstring);
14.   let decm =  Pdec(rc, wsk) in
15.   if (decm=m) then event REC.
16. (*プロセス実行*)
17. process  Receiver | Sender
18. (*end 公開鍵暗号プロトコル*)
```

送信者プロセス Sender は通信路 c から受け取った公開鍵 pkx を利用してメッセージ m を暗号化して通信路 c に暗号文を送信する．一方，受信者プロセス Receiver はまず秘密鍵を選び，対応する公開鍵を計算して通信路 c に送信する．さらに通信路より暗号文 rc を受け取って自身の秘密鍵を利用して復号する．復号結果が m と一致した場合にはイベント REC に到達する．ProVerif でのイベントはプログラミング言語における一種のラベルであり，ProVerif の検証クエリ query event(REC) によって到達可能性の検証を行える．上記の例のように，適切な暗号技術の形式モデルを教員が準備しておけば，プログラミングにある程度慣れた履修者であればプロトコルを直観的に記述できる．上述のイベントはプロセス内の任意の場所に設定可能であるので，イベント到達可能性を検証することで履修者の意図通りの形式化が行われているかを履修者自身でデバッグすることが可能である．

3.4 攻撃手順とその対策の学習

3.3 節で例示した公開鍵暗号の形式化モデルは座学の講義や教科書等で学ぶ公開鍵暗号方式の概念と直観

†1 注：実際のコードには行番号は記述しない．(* *) はコメントアウトを示している．

的に一致しているため理解しやすい．しかし，このモデルでは受信者が暗号文を成功裏に復号できることが検証できるが，メッセージの秘匿性を検証するクエリ `query attacker(m)` を実行すると以下のように具体的な攻撃手順が ProVerif によって導出される．まず ProVerif は検証コードより実行されるプロトコル `Receiver | Sender` を以下のように展開し，ステップ番号を付す．

```
Process:
(
    {1}new wsk: skey;
    {2}let pkk: pkey = pk(wsk) in
    {3}out(c, pkk);
    {4}in(c, rc: bitstring);
    {5}let decm: bitstring = Pdec(rc, wsk) in
    {6}if (decm = m) then
    {7}event REC
) | (
    {8}in(c, pkx: pkey);
    {9}let ctxt: bitstring = Penc(m, pkx) in
    {10}out(c, ctxt)
)
```

次にメッセージ `m` の秘匿性の検証を行う．今回は `m` の秘匿性をやぶる攻撃，即ち攻撃者が `m` を入手する攻撃が発見されるのでその手順が提示される．

```
-- Query not attacker(m[])
Completing...
Starting query not attacker(m[])
goal reachable: attacker(m[])

1. The attacker has some term y_120.
    attacker(y_120).
2. By 1, the attacker may know y_120. Using the
   function pk the attacker may obtain pk(y_120).
   attacker(pk(y_120)).
3. The message pk(y_120) that the attacker may
    have by 2 may be received at input {8}. So
    the message Penc(m[], pk(y_120)) may be sent
    to the attacker at output {10}.
    attacker(Penc(m[], pk(y_120))).
4. By 3, the attacker may know
   Penc(m[], pk(y_120)). By 1, the attacker may
   know y_120. Using the function Pdec the
   attacker may obtain m[].
    attacker(m[]).
```

この攻撃は 1，2 で攻撃者が秘密鍵，公開鍵ペアを生成し，3 でプロトコルの第 {8} ステップで通信路より攻撃者の公開鍵を送信者プロセスに送信している．通信路より受け取った公開鍵の確認を本プロセスでは何ら行っていないので攻撃者が作成した偽の公開鍵を用いて `m` を暗号化した暗号文が通信路へ送信される．そこで 4 で攻撃者が受信した暗号文を攻撃者自身の秘密鍵を用いて復号することにより攻撃者は `m` を入手する．これは公開鍵の配送時に十分な確認が行われないことに起因する典型的な中間者攻撃 †2 である．このように検証結果は比較的容易な英語で攻撃手順が提示されるのでプロトコルにどのような不備があるのかを履修者は理解しやすく，攻撃に対する対策を形式化モデルに加えて試行錯誤することによって暗号プロトコルの学習を対話的に行うことができる．

以上のように，本演習は履修者自身が形式記述した暗号プロトコルを ProVerif で安全性検証することで，ProVerif を教師として対話的に理解度を確認することの実現を目指している．著者のような暗号の専門家がマンツーマンで指導を行えばより効果的な教育が可能であるが，多人数を対象にしたインタラクティブな暗号教育を実現することは有意義であると考える．

4 実践した演習概要

著者の岡崎は暗号理論の専門家であり，自身の研究に形式的暗号プロトコル安全性検証ツール ProVerif を利用している．そこで岡崎は ProVerif を利用した暗号技術に関する教材を作成し，信州大学工学部電子情報システム工学科 3 年生を対象とした演習形式による授業を実践した．本演習は必修の実験・演習科目の 1 テーマとして，3 コマ (90 分×3) 連続の演習を 2 週行い，1 週目に ProVerif の利用方法と，ProVerif を利用した暗号技術とその形式化モデルについての解説，および比較的簡単なプロトコルの形式化演習問題を行う．2 週目には前週に学習した暗号技術の形式化モデルを利用して応用的な暗号プロトコルの設計演習を行った．1 クラスにつきそれぞれ 24，25 名の演習を 5 クラス，合計 124 人に行った．2 週目の暗号プロトコル設計では，学生を 4，5 名ごとのグループに分けてグループごとに協力して 1 つの暗号プロトコルの設計や安全性検証を行い，そのプロトコルの解

†2 中間者攻撃を防ぐための PKI 等について本演習でも解説を行っている．

説と安全性検証結果についてレポート提出を課し評価を行った．本演習では岡崎のほかに，著者の紫村をはじめとする 3 名の大学院生が TA として指導の補助を行った．

本学科では学生に個人用ノート PC を持参させているので，本科目においても各自で ProVerif をダウンロードさせ，著者等が準備したサンプルコードを含む教材を用いて演習を行った．

4.1 1 週目：ProVerif の利用方法と暗号技術形式化モデル

演習 1 週目には ProVerif を利用して暗号技術についての学習を行う．ProVerif 本体に付属するサンプルコードや [15] で著者等が開発した形式モデルを利用して以下の暗号技術についての理解と，それらを利用して暗号プロトコルの形式記述を行う方法を習得する．1 週目に学習する暗号技術は以下のとおりである．

- 秘密鍵暗号・公開鍵暗号 (秘匿性)
- 暗号学的ハッシュ関数・MAC・電子署名 (認証)

それぞれの要素技術には利用目的とそれを実現するための安全性要件があり，ProVerif の形式化モデルを利用して実際に検証を行いながら学習することで理解を促す．その際，著者の岡崎が理論的解説と ProVerif での形式記述方法を解説するとともに ProVerif の利用方法について TA が補助的指導を行った．特に初学者が陥り易い誤った暗号技術の利用方法に基づく暗号プロトコルの記述を行い，ProVerif の検証器に具体的な攻撃手順を導出させることで，要素技術の安全性要件や技術的限界の理解を促した．例えばパスワード認証に対する再送攻撃や公開鍵暗号に対する中間者攻撃の手順を示すことで，それぞれの対策となるチャレンジアンドレスポンス認証や PKI についての解説等を行った．

4.1.1 例 1：秘密鍵暗号

秘密鍵暗号は比較的方式が単純であるため初学者にも理解しやすい．そこでまず秘密鍵暗号の ProVerif 検証コードを利用して秘密鍵暗号の形式モデル化方法と，安全性要件 (秘匿性) の検証方法および，プロセスの記述方法について解説した．形式モデルは著者等によるサンプルコード内で既に記述されており，学習者は主に形式モデルの読み方と，実行プロトコルの記述方法，即ちプロセス定義とその実行方法についての理解を目的とする．以下に秘密鍵暗号のサンプルコードを明記する．

```
1. (*begin 秘密鍵暗号形式モデル*)
2. (*宣言部*)
3. (*項の宣言*)
4. free c:channel. (*通信路 c*)
5. free m:bitstring [private]. (*m を秘密とする*)
6. free sk:bitstring [private]. (*秘密鍵*)
7. (*秘密鍵暗号形式モデル*)
8. fun enc(bitstring, bitstring): bitstring.
9. fun dec(bitstring, bitstring): bitstring.
10. equation forall x: bitstring, s: bitstring;
11.              dec(enc(x, s), s) = x.
12. (*受信プロセスの定義*)
13. let RESP =
14.   in (c, chip:bitstring);
15.   let p = dec(chip, sk).
16. (*クエリ*)
17. query attacker(m).
18. (*実行部*)
19. process
20.    out (c, enc(m, sk)) | RESP
21. (*end 秘密鍵暗号形式モデル*)
```

宣言部

宣言部では 8–9 行目でそれぞれ暗号化，復号関数，`enc`，`dec` を宣言している．このままでは両関数とも単なる 1 対 1 写像の関数となるため 10–11 行目で項の書き換えルール `forall x: bitstring, s: bitstring; dec(enc(x, s), s) = x` を導入し秘密鍵暗号 `(enc, dec)` の満たすべき性質，即ち秘密鍵 `s` によって暗号化された暗号文は同じ秘密鍵 `s` を用いることによってのみ復号されることを形式化している．この形式化モデルは直観的で学習者にとって理解しやすいが，秘密鍵暗号の満たすべき安全性要件「秘匿性」を満たすためには `enc`，`dec` の両関数を適切に利用する必要がある．

一方 13–15 行目では受信プロセス `RESP` の定義を行っている．14 行目で通信路 `c` より受信した値で `bitstring` 型の項 `chip` を初期化し，15 行目で事前に共有済の秘密鍵 `sk` を用いて復号を行っている．ここではあくまでプロセスの定義を行っているのみであり，プロトコル内での呼び出しは実行部にて記述す

る必要がある．

クエリ

17 行目では検証器に検証させるクエリを記述している．ここでは private 指定された項 m の秘匿性，すなわち攻撃者が項 m を知り得る方法があるか否かの検証を行わせる．

実行部

実行部では検証対象なるプロトコルの記述を行う．19–20 行目ではメインプロセスにおいて秘密の項 m を事前に共有済の秘密鍵 sk を用いて暗号化を行い，暗号文 enc(m, sk) を通信路 c に送信する．さらにメインプロセスと並列して受信プロセス RESP を実行している．

検証演習

本演習では秘密鍵暗号等の暗号技術の形式モデル化は著者等が行い，学習者は主にプロセスの定義と実行部を編集することにより学習を進めている．本コードはプロセスの記述方法と暗号学的な秘密鍵暗号の性質を学習させるために作成した．本コードをこのまま ProVerif で検証した結果は true，即ち項 m はプロトコル終了まで秘密裏に保持されると判定され，秘密鍵暗号の形式化モデル (enc, dec) は安全性要件 「秘匿性」を有すると確認できた．

ここで，学習者に実行部の記述を変更させて検証結果が false，即ち攻撃者が項 m を知り得る，安全性要件 「秘匿性」を有さないプロトコルを作成させる．例えばメインプロセスを out (c, (sk, enc(m, sk))) のように変更すると通信路 c に暗号文とともに秘密鍵を送信することになり，攻撃者は傍受した秘密鍵を用いて暗号文を復号関数 dec を用いて項 m を計算する．以上より学習者は ProVerif を用いて秘密鍵暗号は秘密鍵を秘密裏に保持することにより安全性要件「秘匿性」を満たすことができると理解を深めることができる．秘密鍵暗号の場合はこのように安全性要件を満たすための条件は自明のように思えるが，複数の暗号技術を組み合わせて応用的な暗号プロトコルの設計を行う 2 週目の演習では慎重に設計しなければ思わぬ攻撃手順を ProVerif の検証器に指摘されることになり，より深く暗号プロトコルについて学習することが可能となる．

4.1.2 例 2：パスワード認証

認証は秘匿と並ぶ暗号技術の主要な用途の一つである．実用的な暗号プロトコルのほとんどは秘匿機能と認証機能を適切に組み合わせることによって実現されている．認証にはユーザ認証，メッセージ認証等のアプリケーションがあり，その実現方法も様々であるが，本演習ではまず最も典型的なユーザ認証方式であるパスワード認証の ProVerif での形式モデルを用いて認証機能の実現方法について学習した．パスワード認証にはパスワードを平文で送信するベーシック認証や，パスワードを秘匿するためにパスワードのハッシュ値を送信するダイジェスト認証等，様々な認証方式がある．以下にまずダイジェスト認証のサンプルコードを明記する．ただし，パスワード認証では通常 ID とパスワードを送信するが，簡単のために ID 無しでパスワードのみの認証として形式化した．

```
1. (*begin ダイジェスト認証モデル*)
2. (*宣言部*)
3. (*項の宣言*)
4. free c:channel. (*通信路 c*)
5. free s:bitstring [private]. (*パスワード*)
6. fun hash(bitstring): bitstring.(*ハッシュ関数*)
7. (*イベントの宣言*)
8. event PROB.(*パスワードのダイジェストを送信*)
9. event ACC.(*ユーザ認証受理*)
10. (*認証プロセスの定義*)
11. let AuthS (PPW:bitstring) =
12.   in (c, (pw:bitstring));
13.   if( hash(PPW) = pw)then event ACC.
14. (*クエリ*)
15. query event(ACC) ==> event(PROB).
16. query attacker(s).(*パスワードの秘匿も検証*)
17. (*実行部*)
18. process
19.    (event PROB;out(c, hash(s))) | !AuthS(s)
20. (*end ダイジェスト認証モデル*)
```

宣言部

ProVerif では認証の安全性検証をプロセス内に設定したイベントの到達可能性を判定することによってチェックする．宣言部 11–13 行目で認証プロセス AuthS を形式化した．11 行目のようにプロセスに実行時に引数を指定して定義することができ，引数としてプロセスに与えるユーザのパスワード，項 PPW は private に保たれる．13 行目では通信路より受信したダイジェスト pw と事前に共有したユーザパス

ワードのハッシュ値 hash(PPW) が一致した場合に限りユーザ認証イベント ACC が実行 †3 される．

クエリ

15 行目は認証イベントの到達可能性を検証させるクエリである．event(ACC) ==> event(PROB) の検証結果が true であるならば，ユーザ認証イベント ACC が実行されるには必ず事前にユーザがパスワードのダイジェストを送信するイベント PROB が実行される．つまりユーザがパスワードのダイジェストを送信しない限り認証プロセスがユーザを受け入れることが無いことを意味する．また，16 行目のように複数のクエリを同時に指定することもできる．

検証演習

ProVerif の検証器は 15 行目，16 行目の 2 つのクエリともに true と判定する．これはパスワードによるユーザの認証，パスワードの秘匿性ともに実現していることを意味する．しかし実際のサービスでは同じユーザが何度も繰り返しパスワード認証を行うため，一度ユーザがダイジェスト認証に成功すれば，能動的な攻撃者はその際に送信されたパスワードのダイジェストを利用する再送攻撃を行うことが可能である．ProVerif ではイベントの対応付けが一対一であることも判定可能であり，15 行目のクエリを inj-event(ACC) ==> inj-event(PROB) に変更することにより再送攻撃を検出する．以下は再送攻撃を指摘した ProVerif の出力の一部抜粋である．

```
1. The event PROB (with environment @occ1 =
   @occ_cst) may be executed at {1}. So the
   message hash(s[]) may be sent to the attacker
   at output {2}.
   attacker(hash(s[])).
2. The message hash(s[]) that the attacker may
   have by 1 may be received at input {4}. So
   event ACC may be executed at {6} in session
   endsid_112.
   end(endsid_112, ACC).
```

認証プロセスが毎回ランダムなチャレンジコードを送信し，ユーザはチャレンジコードとパスワードを用いてダイジェストを計算することによって再送攻撃を防ぐチャレンジアンドレスポンス認証が知られている．以下のようにダイジェスト認証サンプルコードの一部を変更することによってチャレンジアンドレスポンス認証の学習を行った．

```
1. (*begin チャレンジアンドレスポンス抜粋*)
2. (*認証プロセスの定義*)
3. let AuthS (PPW:bitstring) =
4.   new challenge:bitstring;
5.    in (c, (pw:bitstring));
6.    if( h(PPW, challenge) = pw)then event ACC.
7.  (*実行部*)
8.  process
9.   in (c, rchallenge:bitstring);
10.  (event PROB;out(c,h(s, rchallenge))) | !AuthS(s)
11. (*end チャレンジアンドレスポンス抜粋*)
```

同様にして 3.3 節で示した公開鍵暗号方式に対する中間者攻撃を ProVerif の検証器に導出させ，PKI に関する解説を行った．ただし PKI については 2 週目の演習例題の 1 つとしたため具体的な形式化サンプルコードを提示することはしなかった．

4.2 2 週目：ProVerif を用いた暗号プロトコル設計演習

2 週目の演習では 1 週目に学習した，秘密鍵暗号・公開鍵暗号・暗号学的ハッシュ関数・MAC・電子署名の 5 つの形式化モデルを利用して応用的な暗号プロトコルの設計演習を行う．グループごとに 1 つの暗号プロトコルを設計，ProVerif を用いて安全性検証を行いその成果をレポートにて報告させた．設計する暗号プロトコルは自由に提案してよいとするが，実際には初学者にとって実用的なプロトコルを提案することは困難であろう．そこで本演習では以下のようないくつかの例題を与えた．

- PKI
- ハイブリッド暗号
- ネット決済
- S/KEY

この演習において選択された暗号プロトコルは図 1 のようであった．特に DH 鍵交換プロトコルにおいては書き換えルール equation を自ら作成し，DH 鍵交換プロトコルの形式化ができているグループが一組あり，プロトコル設計の理解が深まっていると考える．

†3 イベント ACC はラベルであり，ここでの実行とはイベントに実行パスが到達することを意味する．

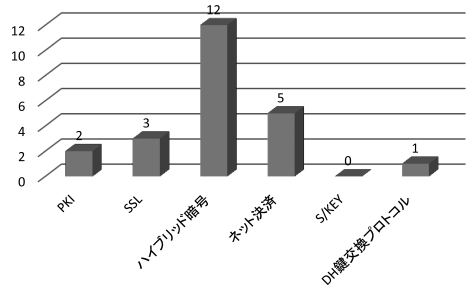


図 1　選択された暗号プロトコル

著者らはこれらのプロトコルについて詳細の解説は行わず，利用目的と満たすべき安全性要件のみを提示することにより，グループ内で相談して暗号プロトコルの仕様を設計させた．3 コマ連続の演習中に暗号プロトコルの設計と ProVerif による安全性検証を行い，全てのグループが安全なプロトコルを完成させることができるまで教員及び TA が補助的な指導を行った．ハイブリッド暗号は秘密鍵暗号と公開鍵暗号，PKI は公開鍵暗号と電子署名の組み合わせによって実現できるため，1 週目で学習した形式化モデルのみを利用して設計可能である．

本演習の学習効果は事前に著者らが予想していたよりも高く，学習者等は各グループで熱心に討論し暗号プロトコルの設計と ProVerif による安全性検証を行っていた．中にはよりインターネットの標準暗号である SSL/TLS と本質的に等価な暗号プロトコルの設計を行ったグループもあった．これは上述の PKI とハイブリッド暗号を組み合わせることにより実現したものであり，非常に完成度の高い優秀なレポートが提出されたと評価した．

5　演習の評価と議論

5.1　学習者による演習のアンケート結果

授業の終了後に下記のアンケートを取り，ProVerif を用いた暗号技術教育の効果について確認した．

アンケート 1：従来型の講義形式だけの授業に比べて，暗号プロトコル自動検証ツール ProVerif を用いた授業は個々の暗号技術 (共通鍵暗号，公開鍵暗号，ディジタル署名等) の理解が高まったと思う．

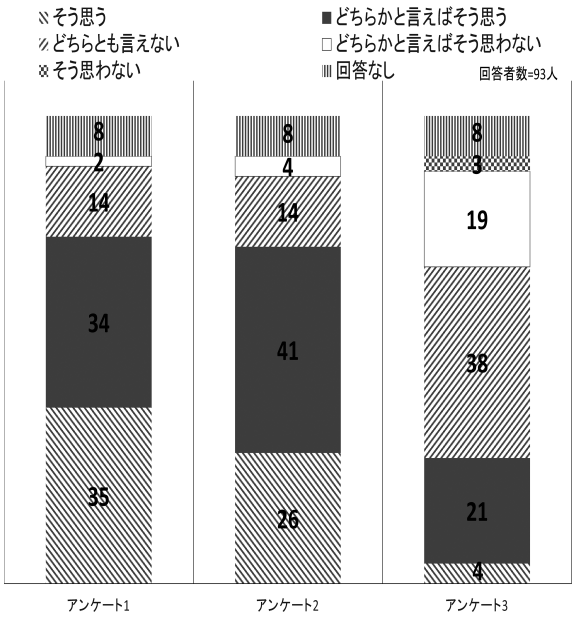


図 2　アンケートを集計したグラフ

アンケート 2：従来型の講義形式だけの授業に比べて，暗号プロトコル自動検証ツール ProVerif を用いた授業は暗号プロトコルの理解が高まったと思う．

アンケート 3：暗号プロトコル自動検証ツール ProVerif の使い方は習得しやすい．

アンケートの回答者数は 93 人になり，その集計結果は図 2 のようになった．この結果より，学生は ProVerif を用いることで暗号技術を効果的に学ぶことができた．しかしながら，ProVerif の使い方を習得することは学生にとって難しいものであった．

5.2　教員による授業の評価

レポートの評点分布を図 3 に示す．本演習は信州大学工学部電子情報システム工学科 3 年生 124 名を対象に行った．本学科は 2 年次より電子，通信，情報の 3 プログラムに分かれ専門性の異なるカリキュラムでの教育を行っている．本演習は通信，情報プログラムの学生に開講されているがプログラム間で既習科目が異なるため図 3 では情報プログラム 99 名の評点のみを対象とした．A より順に高評価とし，C までが合格点とした．ただし D の 7 名にはレポート未提出，欠席を含んでいる．図 3 から分かるように提出された

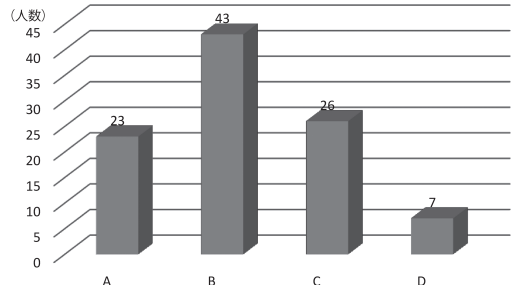


図 3　学生の理解度

レポートは充実した内容のものが多く，本演習の学習効果は著者等の予想を上回るものであったと考える．

4.2 節で述べたように，2 週目のグループ演習は著者等の予想よりはるかに有効で，グループ内で活発に相談をしながらプロトコルの設計と安全性検証を行っていた．当初の予定では学習者等が自ら暗号プロトコルの仕様について検討して設計を行うことを期待していたが，多くのグループが例示したプロトコルについて Web サイトや書籍等の詳細な解説を参考にして ProVerif での形式モデル化，安全性検証を行っていた．しかし ProVerif による安全性検証を行うため，暗号プロトコルについての解説内容を誤って理解していた場合には形式モデル化したプロトコルに対する具体的な攻撃手順を ProVerif によって導出できるため問題点を直観的に理解し易く，効果的に暗号技術についての学習を行えたと考える．これは 5.1 節のアンケート結果，「座学の講義よりも本演習による暗号技術について理解しやすい」との結果と一致する．アンケート対象の受講者が暗号プロトコルに関する授業を受けるのは本科目が最初の機会であり，本アンケートの結果はあくまでも受講者の主観による回答である．しかし 4 章で述べたように，本演習は 2 週連続で行い，1 週目には ProVerif による検証を行いながらも基本的には座学での解説を行った．1 週目の最後に行う基礎演習と，2 週目に行うグループ演習の際に，1 週目の座学や補助教材，受講者が Web で調べてきた解説サイト等の内容を，受講者が実際に形式記述を行い ProVerif で検証を行ってみて，ツールの探索によって具体的な攻撃手順が導出されることで，分かった気になっていた事に気付き，その対策を受講者自身で考えたり，教員や TA に質問することで解決できるため，ツールによる検証が受講者にとって暗号プロトコルの理解の一助になり，従来より受講者の暗号プロトコルの理解が深まっていると考える．

グループ演習の際に教員及び TA に対する質問の多くは ProVerif の文法，記述方法に関するものであった．これは 1 週目の解説及び演習で暗号技術に関する理解が深まっていたからであると考えるが，複雑なプロトコルの形式モデルを記述しようとすると構文エラー等が頻繁に起こっていた．これは 5.1 節のアンケート結果，「ProVerif は難しい」との結果と一致する．教員及び TA が演習時に受講者より受けた質問内容，および提出されたレポートより，受講者が「ProVerif は難しい」と感じる原因は，教員の想定よりも受講者が積極的に演習に取り組んでいたことに起因する．本演習ではあらかじめ教員が用意した暗号技術の形式化モデルを利用してプロトコルの実行部分のみを記述するように指導していたが，受講者が自主的に形式化モデルの記述にも取り組んでいたため，事前の想定よりも高度な質問が多く寄せられていた．本演習では ProVerif に習熟した教員及び TA3 名の合計 4 名が指導を行っていたのですぐに対応し，円滑に演習を行うことが可能であったが，本演習教材を公開し，他の教員に利用してもらうためには改善が必要である．

6　CAI 教材化計画とその課題

5.1 節の議論により本研究の教育効果は確認された．ただし，これは著者等のような暗号理論，形式検証技術に関する専門知識を有する者が指導した場合に限られる．2 章で述べたように本研究の目的は暗号技術に関する初学者向きの教材不足，さらには指導者不足を解決するためのインタラクティブな教材を開発することにあった．本演習の暗号技術の内容自体に関する教育効果は高いと考えるため，ProVerif の習熟を容易にする，あるいは ProVerif の習熟度が必ずしも高くなくても，ユーザインタフェースの工夫等で暗号プロトコルの形式化モデル記述および安全性検証を行える CAI 教材を開発することが望ましいと考える．

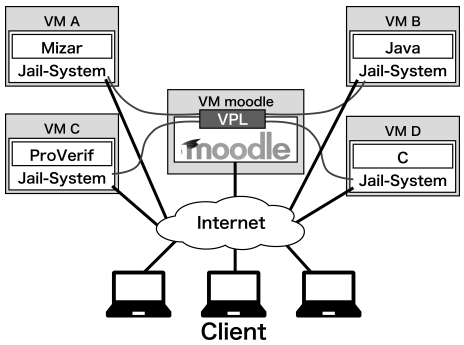


図 4 教育支援システムの動作環境概念図

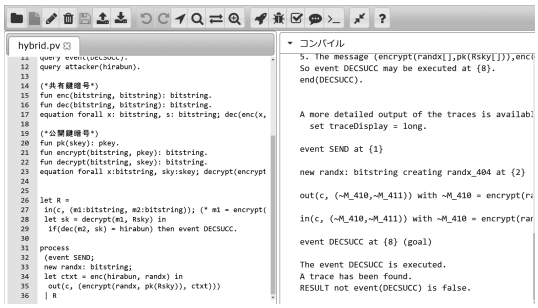


図 5 moodle 上での ProVerif の動作例

6.1 CAI 化方針

著者等は科研費研究課題 [17] において数学証明問題学習用 CAI 教材 [18] 開発に参加した．[17] では証明問題の成否判定に形式的定理証明システムの一つである Mizar [1] を用いる CAI 教材を開発した．Mizar を用いた証明問題 CAI 教材システムはフリーの LMS として有名な moodle [13] 用のアドインとして開発した．この moodle 用 Mizar アドインを用いれば moodle 上で教師権限を持つユーザは比較的容易に証明問題を作成することが可能であった．本研究では moodle 用のプログラミング言語教育用アドインである VPL [7] を利用して moodle 上で ProVerif を実行する環境を構築することとした．VPL は標準で C 言語，Perl 等をはじめとする多数のプログラミング言語をサポートしており，学習者は moodle 環境の Web ブラウザ上でプログラムコードの編集を行い，そのコンパイルや作成した実行形式ファイルの実行は専用の jail サーバ上に隔離して行うことができる．また，標準でサポートされていないプログラミング言語であっても，Linux 上で動く，入力ファイルはテキスト形式である，出力はテキスト形式ファイルまた標準出力である等の条件を満たせば jail サーバ上に導入して VPL 環境で利用可能である．ProVerif や Mizar はプログラミング言語ではなくあくまで検証ツールではあるが，前述の条件を満たしているので導入を試みたところ問題なく VPL 環境で利用可能であった．さらに，jail サーバ上でのプログラムの実行方法をシェルスクリプトにより指定することが可能であるのでユーザによる出題方法のカスタマイズは比較的自由にすることが可能であろう．

そこで，以下では現在開発中の moodle VPL 環境上での ProVerif による暗号技術教材の概要を紹介する．

著者等は，以前に VM 上に moodle と VPL を利用した C 言語プログラミング教育支援システムを開発した [14]．更に，開発した教育支援システムに ProVerif を導入した [12]．図 4 は，開発した教育支援システムの動作環境概念図である．

VM 上に教育支援システムを開発することより，システム管理者は，仮想マシンイメージを仮想化ソフトウェア上に転送し，IP アドレス等を環境に合わせて設定することにより，比較的容易にシステムを利用可能になる．仮想マシンイメージは，ホスト OS を問わずに利用することが可能であり，Windows や Mac，Linux が稼働しているコンピュータ上にも導入することが可能である．moodle VPL 環境上では，学生は ProVerif をインストールする必要がなく，ブラウザのみで動作するため，タブレットやスマートフォンで利用可能である．また，学生は moodle 上のテキストエディタに ProVerif のコードを記述するので，ファイルのアップロードの必要がない．教師は生徒の ProVerif コードを採点する際，コードをダウンロードすることなく，moodle 上で ProVerif コードの確認が可能である．図 5 は moodle 上で ProVerif を動作させた例である．

6.2 暗号・形式検証の知識を有しない教員による実践

本演習は暗号と形式検証技術の専門家である著者等が教材開発，演習指導を行ったが，他機関で本教材を

採用して暗号プロトコルの教育を実践する場合，指導者が必ずしも暗号や形式検証に関する知識を有するとは限らない．しかし著者等は，暗号や形式検証の専門家でなくとも，ICT 技術教育に携わる教員にとって，本演習の実践がそれほど困難でないように教材開発を行っている．本演習はあくまでセキュリティや暗号技術の専門家では無い．ICT 技術者向けのコースとして教材開発を行っているため，現在準備している，共通鍵暗号，公開鍵暗号，暗号学的ハッシュ関数，メッセージ認証子，ディジタル署名の 5 つの形式モデルがあれば十分に演習を行えるものと想定している．著者等が用意し，実際に本科目で利用しているサンプルコードには，それだけで独習することが可能となるように詳しい解説をコメントとして記載しており，何らかのプログラミング言語を習熟している者であれば，このコメントのみで基本的な学習は可能である．また，教員が本教材で予め準備されていない高度な暗号技術要素についての形式モデルを作成する事も，その暗号技術要素に関する知識があれば十分可能である．さらに，様々な形式モデルが必要になったとしても多くのものは既存の論文，ProVerif 配布物に同梱のサンプルコード，NICT が公開している暗号プロトコル評価ポータルサイト [10] 等を参考にすることが可能である．

6.3 今後の課題

前節で紹介した moodle VPL 環境上での ProVerif による暗号技術教材を開発，改善することにより，本研究の問題点の一つである ProVerif の習熟が必ずしも容易ではないことは解決可能であると考える．

しかし，本研究の主たる目的は教育効果の高い暗号技術に関する CAI 教材の開発である．5.1 節で述べたように，暗号技術および ProVerif に関する専門知識を有するものが指導を行う場合には本教材は高い教育効果を発揮すると期待できる．指導者が専門家でない場合，あるいは CAI で独習する場合であっても高い教育効果を有する教材を作成する方針を模索することが今後の課題である．そのためにはまず教育効果を客観的に調べる方法を検討する必要がある．ProVerif の習熟度が必ずしも高くなくても教育効果を高めるために，CAI 教材では学生が記述できる部分をプロセス実行部に制限する，あるいは Blockly [9] 等のビジュアルプログラミング教材の利用等を利用することにより，受講者自身に形式モデルを記述させることなく，教員が準備した形式モデルのみを利用して暗号プロトコル設計演習を行えるような教材開発方法を検討している．

7 まとめ

本研究では，学生に対して形式検証ツールを用いた暗号教育を行った．図 2 や図 3 から形式検証ツールを用いた暗号教育は従来型の座学だけの授業に比べて効果的であることが判明した．よって，今後，形式検証ツールを用いた暗号教育支援環境を構築し，暗号技術教材を開発することで学習者が e-Learning 上で暗号技術をより簡単に学べるようにしたい．

謝辞 本研究の一部は JSPS 科研費 JP17K00182, JP18K02917 の助成を受けたものです．本研究成果は，CELLOS (Cryptographic protocol Evaluation toward Long-Lived Outstanding Security) の活動に基づいて得られたものです．

参 考 文 献

[1] ASSOCIATION OF MIZAR USERS: Mizar Project, http://mizar.org/. アクセス日：2019 年 10 月 21 日.
[2] Backes, M. and Maffei, M.: Security, Core Lecture, Universitt des Saarlandes, Germany, https://www.infsec.cs.uni-saarland.de/teaching/10WS/Security/. アクセス日：2019 年 10 月 21 日.
[3] Blanchet, B.: ProVerif:Cryptographic protocol verifier in the formal model, http://prosecco.gforge.inria.fr/personal/bblanche/proverif/. アクセス日：2019 年 10 月 21 日.
[4] Blanchet, B., Smyth, B., Cheval, V., and Sylvestre, M.: ProVerif 2.00: Automatic Cryptographic Protocol Verifier,User Manual and Tutorial, https://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf. アクセス日：2019 年 10 月 21 日.
[5] Chothia, T.: Modelling and Analysis of Security Protocols, http://www.cs.bham.ac.uk/~tpc/cwi/Teaching/index.html. アクセス日：2019 年 10 月 21 日.
[6] CRYPTREC Cryptography Research and Evaluation Committees: 電子政府推奨暗号リスト, https://www.cryptrec.go.jp/list.html. アクセス日：2019 年

10 月 21 日.
[7] del Pino, J. C. R.: VPL Virtual programming lab, https://vpl.dis.ulpgc.es/. アクセス日：2019 年 10 月 21 日.
[8] Dolev, D. and Yao, A. C.: On the security of public-key protocols, *IEEE Transactions on Information Theory*, Vol. 29, No. 2 (1983), pp. 198–208.
[9] Google LLC: Google for Education Blockly, https://developers.google.com/blockly/. アクセス日：2019 年 10 月 21 日.
[10] 情報通信研究機構: Cryptographic Protocol Verification Portal, http://crypto-protocol.nict.go.jp/. アクセス日：2019 年 10 月 21 日.
[11] Laud, P.: Cryptographic protocols, Tartu University Estonia, http://kodu.ut.ee/~peeter_l/teaching/krprot09s/. アクセス日：2019 年 10 月 21 日.
[12] Miyamoto, T., Shimura, S., Watanabe, T., Futa, H. O. Y., and Murakami, Y.: e-Learning System for Cryptography on Moodle, *Internet Conference 2018(IC2018), Poster session*, 2018.
[13] Moodle HQ: Moodle Project, https://moodle.org/. アクセス日：2019 年 10 月 21 日.
[14] 中村雅宏，渡邊樹，金田益典，岡崎裕之，村上恭通: moodle を用いたプログラミング教育支援システム，第 *40* 回情報理論とその応用シンポジウム *(SITA2017)*，ポスターセッション，2017.
[15] Okazaki, H., Futa, Y., and Arai, K.: Suitable Symbolic Models for Cryptographic Verification of Secure Protocols in ProVerif, *The International Symposium on Information Theory and Its Applications (ISITA2018: October 28-31, 2018, Singapore)*, 2018, pp. 326–330.
[16] Pomorova, O. and Lysenko, S.: Formal and Intelligent Methods for Security and Resilience: Education and Training Issues, *Information & Security: An International Journal*, Vol. 35(2016), pp. 133–150.
[17] 師玉康成 (代表): MIZAR 数学ライブラリの構築と大学数学向け高度遠隔教育用コンテンツ開発，https://kaken.nii.ac.jp/ja/grant/KAKENHI-PROJECT-22300285/. アクセス日：2019 年 10 月 21 日.
[18] Takaya, I., Okazaki, H., Yamazaki, H., Kawamoto, P., Wasaki, K., and Shidama, Y.: Content Development for Distance Education in Advanced University Mathematics Using Mizar, *2013 International Conference on Foundations of Computer Science (FCS'13: July 22-25, 2013, USA)*, 2013, pp. 312–326.

A ProVerif

ProVerif [3] は Dolev–Yao モデル [8] に基づく暗号プロトコルの形式的安全性検証ツールである．ProVerif では暗号プロトコルを形式モデルで記述し，そのプロトコルの秘匿，認証をはじめとする安全性要件を検証することが可能である．実際の暗号プロトコルでは秘密鍵暗号は AES，暗号学的ハッシュ関数は SHA–256 というように具体的な暗号スィーツを選択して構成するが，ProVerif による形式検証では個々の要素技術は公開鍵暗号モデル，電子署名モデルというように理想化された形式モデルを用いて暗号プロトコルを構成し，その安全性を ProVerif の検証器がサポートするクエリを用いて検証する．このとき，個々の暗号技術は理想化されている，即ちそれ自身が破られることは無いとの仮定の下で安全性検証を行うことに注意されたい．実際の攻撃においても個々の暗号スィーツを破ることはまれであり，ほとんどの場合，暗号スィーツの不適切な利用，組み合わせ方の不備や安全でないデータの利用が脆弱性の原因となっている．本演習の目的は要素技術それぞれの安全性に関する機能と，適切な利用方法および組み合わせ方を学習することを目的としている．

A.1 ProVerif による検証概要

ProVerif の検証ファイル (*.pv) は，宣言部，クエリ，実行部で構成される．暗号プロトコルは攻撃者による傍受が可能である公開通信路を介したプロセス間の通信手順として形式化する．宣言部ではプロトコルの構成に必要な項 (変数) や関数，およびプロセスの定義を行う．暗号技術は宣言した項と変数に関する書き換えルールを定義することによって形式モデル化する．

実行部では宣言部で定義したプロセスの呼び出し方法を記述することで，検証対象となる暗号プロトコルを定義する．プロトコル実行時に攻撃者には公開通信路より入手可能な項と，`private` 属性を付されていない関数を利用したあらゆる操作が許される．検証器はクエリで指定された攻撃が成功する手順があるか否かを，攻撃者がとり得る操作を網羅的に探索することによって判定し，もし攻撃が発見されればその具体的な手順を表示する．

A.2 宣言部

宣言部では暗号プロトコルで利用する項 (変数) や通信路，関数の定義を行う．これらはそれぞれ `privite`，`public` の属性を付することが可能であり，攻撃者に

よる利用の可否を指定できる．さらに項や関数に関する書き換えルールを定義することによって，暗号技術のモデル化を行う．書き換えルールによる暗号技術の具体的な形式モデルについては 4.1 節で解説する．宣言部ではさらに実行部で実行するプロセスの定義も行うが詳細は 4.1 節で解説する．

A.3 実行部

実行部では検証対象となる暗号プロトコルの実行方法を定義する．暗号プロトコルは並列に実行されたプロセスが指定された通信路を介してデータの送受信を行うことにより実行される．本節では宣言部で定義したプロセスもしくは実行部に直接記述されたメインプロセスの呼び出し方法を解説する．

並列実行するプロセスを以下のように | で区切って記述する．

$$\texttt{Initiator} \mid \texttt{responder}_2 \mid \texttt{responder}_1 \mid \cdots$$

!P は プロセス P を以下のように実行回数を制限することなく並列実行する．

$$\texttt{P} \mid \texttt{P} \mid \texttt{P} \mid \ldots$$

各プロセスは通信路を介して項を送受信するが，public 属性を付された通信路を通過する項は攻撃者に傍受されるものとする．詳細なシンタックスは [4] を参照されたい．

A.4 クエリ

ProVerif は暗号プロトコルの秘匿性，認証等の安全性要件を検証するためのクエリをサポートしている．ProVerif のクエリはあくまで検証技術に基づいたものであり，暗号学的な安全性要件を必ずしも直接検証可能であるわけではない．したがって，暗号学的に意味のある検証を行うためには工夫が必要であり教材として利用する場合には直観的に理解することが困難である．著者等は ProVerif での検証と暗号学における安全性検証とのギャップを埋めるための ProVerif における形式化方法 [15] を提案しており，本演習の教材ではその成果を利用している．詳細は 4.1 節を参照されたい．本節では本演習で利用した基本的なクエリの紹介を行う．

`[query attacker(X)]`

攻撃者が秘密の項 X を完全解読するか否かを判定する．`not attacker (X)` の判定結果が `true` であれば，項 `X` はプロトコル終了まで秘密裏に保たれる．

`[query event(X(v)) ==> event(Y(v))]`

プロセス内に設定したイベントの到達可能性を判定する．`event(X) ==> event(Y)` の検証結果が `true` であるならば，`X` が実行されるには必ず事前にイベント `Y` が実行される．検証器はイベント `Y` を経ずにイベント `X` に到達する方法を探索することにより判定を行う．本クエリはイベント間の依存関係の判定を行っており，ユーザ認証等の安全性要件の判定に利用される．オプションとして `event(X(v)) ==> event(Y(v))` のように同一データを利用したイベントの依存関係も判定可能である．例えば， `v` をユーザ ID とした場合，サーバによるユーザ `v` の認証が成功 (イベント `X(v)`) するのであれば，事前に必ずユーザ `v` によるユーザ証明 (イベント `Y(v)`) が実行されているというように対応する項を限定した検証を行うことが可能である．さらに `inj-event(X(v)) ==> inj-event(Y(v))` とすることで，イベントの 1 対 1 対応性の検証，即ち再送攻撃耐性の判定も行える．また，`query event(Z)` とすると単独のイベント `Z` の到達可能性の判定も可能である．

岡崎裕之

1999 年京都工芸繊維大学工芸学部電子情報工学科卒業．2004 年京都工芸繊維大学大学院工芸科学研究科博士後期課程修了．2005 年信州大学大学院助手．2007 年より信州大学大学院助教．博士 (工学)．情報処理安全確保支援士 (登録番号 018816)．情報セキュリティ，特に暗号理論，形式検証などの研究に従事．

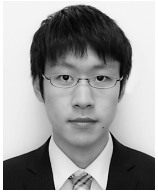
紫村彰吾

2018 年信州大学工学部 電子情報工学科卒業．同年 4 月より信州大学大学院 総合理工学研究科修士課程に所属．形式検証ツール ProVerif を用い

た暗号教育の研究に従事している.

宮本 樹

2014 年大阪電気通信大学情報通信工学部通信工学科卒業. 2018 年同大学院博士前期課程工学研究科電子通信工学専攻修了. 在学中, 組織間通信に適した暗号, moodle と VPL を用いた e-ラーニングシステムの研究開発に従事.



渡邊 樹

2017 年大阪電気通信大学情報通信工学部通信工学科卒業. 在学中, moodle と VPL を用いた e-ラーニングシステムの研究開発に従事. 現在, 日本インフォメーション株式会社管理本部総務部経営企画.

布田裕一

1996 年京都工芸繊維大学工芸学部電子情報工学科卒. 1998 年同大学大学院工芸科学研究科博士前期課程修了. 2012 年信州大学大学院総合工学系研究科博士課程修了. 博士 (工学). 1998 年松下電器産業株式会社 (現パナソニック株式会社) 入社. 2013 年北陸先端科学技術大学院大学特任准教授. 2016 年より東京工科大学コンピュータサイエンス学部准教授. 暗号技術, 暗号方式の形式検証, ネットワークセキュリティなどの研究に従事.

村上恭通

1989 年京都工芸繊維大学工芸学部電子工学科卒. 1991 年同学大学院工芸科学研究科博士前期課程修了. 1997 年京都工芸繊維大学工芸科学研究科博士後期課程社会人入学. 2000 年同単位取得退学. 2001 年京都工芸繊維大学博士 (工学). 1991 年村田機械 (株) 入社. 情報機器事業部所属, 2002 年 R&D センター所属. 2003 年大阪電気通信大学工学部通信工学科講師. 2005 年同学情報通信工学部通信工学科講師. 2011 年准教授. 2016 年教授. 暗号と情報セキュリティの研究に従事.