名古屋大学大学院工学研究科
博士論文

関連鍵攻撃に対する公開鍵暗号要素技術の安全性に関する研究

2017 年 2 月

名古屋大学大学院工学研究科
計算理工学専攻

森田　啓

要旨

現代暗号の大きな分類として，共通鍵暗号系と公開鍵暗号系とがある．共通鍵暗号系では，通信のやりとりをする二者が同一の鍵を共有してデータの秘匿や認証などを行う．一方，公開鍵暗号系では，二者がそれぞれに秘密鍵と公開鍵のペアを持ち，公開鍵のみを公開し，秘密鍵は自身で保持した状態でデータの秘匿や認証などを行う．本論文においては，後者の公開鍵暗号系に着目し，その要素技術の中でも，非対話型鍵交換（Non-Interactive Key Exchange，略して NIKE）および署名方式について取り上げる．

NIKE は，二者が公開鍵の交換以外のやりとりをせずに共有鍵を計算することのできる公開鍵暗号要素技術である．これは，どのように二者間で秘密鍵を共有するのかという共通鍵暗号系特有の問題を解決する公開鍵暗号系の技術であり，Diffie-Hellman 鍵共有法が代表的な例の一つとして挙げられる．また，署名方式は，送りたいメッセージの偽造や改ざんを防ぐ公開鍵暗号要素技術であり，日常生活における印鑑やサインの役割を電子的に達成する技術である．Schnorr 署名方式，DSA，ElGamal 署名方式など，これまでに様々な方式が提案されてきた．

これまでに，NIKE 方式に対して Freire ら（PKC 2013）は，攻撃者の行える攻撃の種類の違いに関連して 4 つの安全性を定義し，それら安全性が実際には等価であることを示した．これにより，攻撃の種類が少なく記述の簡易な攻撃者に対して安全性を示せば十分であることが示された．加えて，Freire らは安全性証明の可能な NIKE 方式を提案した．なお，NIKE の安全性定義において攻撃者は，様々なユーザのペアが共有した鍵に関する知識のみを用いて，新たなユーザのペアの共有鍵と乱数との識別をするよう要求される．署名方式に対する通常の安全性モデルでは，複数の正当なメッセージと署名のペアの知識だけを用いて，新たなメッセージに対する署名を偽造しようとする攻撃者を考える．偽造にはその種類によって，選択的偽造や存在的偽造と呼ばれるものがあり，攻撃者の動作には，これまでに得た情報を用いて適応的に攻撃を行うか，非適応的に攻撃を行うかなどいくつかの種類が考えられる．それら偽造の種類と攻撃者の動作の違いによって，複数の安全性モデルがこれまでに考えられている．暗号技術の研究・開発において，どのような安全性モデルを考慮するかは，対象とする暗号要素技術がどのような場面で利用されるかという現実問題に関連しており，最も説得力のある安全性モデルが採用される．

近年，タンパリングやフォルトインジェクション攻撃といったサイドチャネル攻撃を考慮に入れるために，Bellare と Kohno（Eurocrypt 2003）はより強力な攻撃者を考える理論的枠組みである，関連鍵攻撃（related-key attacks，略して RKA）を

定式化した．これは，通常の安全性では考慮されなかった，秘密鍵に変更を加え，その上で計算されたアルゴリズムの出力を得ることの出来る攻撃者を考えるものである．本論文では，現実の状況を鑑みるに，これまで考えられていた通常の安全性では不十分だと考え，サイドチャネル攻撃を考慮に入れたこの RKA に着目し，NIKE 方式および署名方式の RKA 安全性について議論する．

NIKE 方式に関して本論文では，Freire らの定義した安全性に基づいて，NIKE 方式に対して 4 つの RKA 安全性定義を与える．これらの定義において，ユーザの秘密鍵を改ざんでき，その変更された秘密鍵を用いて計算された共有鍵を得られるような攻撃者を考える．ここで RKA 安全性は，攻撃者に許される秘密鍵への変更に対応する関連鍵導出関数（related-key derivation function, 略して RKD 関数）に関して定義される．この RKD 関数として，線形関数，アフィン関数，多項式などがこれまでに考えられている．本論文では，4 つの安全性定義間の含意関係および分離を示す．具体的には，本論文で RKA-CKS-light 安全性と呼ぶ安全性以外の 3 つの安全性定義は等価である一方，線形関数に関する RKA を考慮するならば，RKA-CKS-light 安全性はその他 3 つの安全性定義とは等価ではないことを示す．また本論文では，Freire らによって提案された NIKE 方式の一つが，符号付剰余群（group of signed quadratic residues）上で定義された Double Strong Diffie-Hellman (DSDH) 仮定のもと，ランダムオラクルモデルにおいて，最も強い RKA 安全性を満たすことを証明する．なお DSDH 仮定は素因数分解仮定から含意される仮定である．

署名方式に関して本論文では， Schnorr 署名方式，DSA の変型，ElGamal 署名方式の変型の 3 つのよく知られた署名方式の，RKA 安全性について考える．署名方式に対する RKA 安全性モデルでは，署名鍵を改ざんでき，その変更された署名鍵を用いて計算された署名を得られる攻撃者を考える．このとき RKD 関数は攻撃者が署名鍵に変更を加えることに相当する関数である．まず初めに，上記の 3 つの署名方式は，多項式に関する弱い RKA 安全性に関して安全であることを示す．次に，一方で，Schnorr 署名方式も DSA も ElGamal 署名方式も線形関数に関する RKA 安全を達成できないことを，具体的な攻撃を示すことで証明する．最後に，Schnorr 署名方式，DSA（の変型），および ElGamal 署名方式の変型に軽微な修正を加えると多項式に関する RKA 安全性を満たすようになることを示す．

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景と目的

現代暗号の大きな分類として，共通鍵暗号系と公開鍵暗号系とがある．共通鍵暗号系では，通信のやりとりをする二者が同一の鍵（共有鍵と呼ばれる）を共有してデータの秘匿や認証などを行う．DES（Data Encryption Standard）や AES（Advanced Encryption Standard）などに代表される，データの秘匿を目的とした暗号化方式や，メッセージ認証コードと呼ばれる認証方式が含まれる．一方，公開鍵暗号系では，二者がそれぞれに秘密鍵と公開鍵のペアを持ち，公開鍵のみを公開し，秘密鍵は自身で保持した状態でデータの秘匿や認証などを行う．RSA 暗号や ElGamal 暗号に代表される，データの秘匿を目的とした暗号化方式や，二者間でやりとりせずに鍵を共有する技術である非対話型鍵交換（Non-Interactive Key Exchange，略して NIKE）や，偽造・改ざん防止を目的とする署名方式といったものが含まれる．本論文においては，後者の公開鍵暗号系に着目し，その要素技術の中でも，NIKE および署名方式について取り上げる．

NIKE および署名方式に関しては，これまで様々な研究がなされてきた．本論文では関連鍵攻撃という，これまで考えられてきた攻撃よりも強い攻撃，つまり攻撃者のできることが多い攻撃について考え，その攻撃に対する安全性について考察する．以下では，関連鍵攻撃について記した後に，NIKE および署名方式について記す．

1.1.1　関連鍵攻撃について

これまでに考えられていた攻撃より強い（攻撃者の出来ることが多い）攻撃である関連鍵攻撃（Related-key attacks, 略して RKA）は，Bellare と Kohno [4] により定式化された．RKA は，暗号技術が実装された装置を物理的な手段で観測することにより暗号方式を破ろうとするサイドチャネル攻撃と呼ばれる攻撃技術の中でも特

に，タンパリングやフォルトインジェクション攻撃と呼ばれる現実の攻撃に対する安全性を，理論的な側面から捉えている．タンパリングやフォルトインジェクション攻撃は，ハードウェア内部に保存された秘密鍵などの情報を外部から電磁波の照射などを用いて改ざんするという能動的な攻撃である．RKA は，攻撃者がハードウェアに記録された秘密鍵を変更する状況において，関連鍵導出関数（related-key derivation function, 略して RKD 関数）と呼ばれる関数に応じて秘密鍵が変更されると捉え，そうして変更された秘密鍵を用いて実行されたアルゴリズムの出力を観察することができるような攻撃者を考慮する．RKD 関数としては例えば，線形関数，アフィン関数，多項式などを用いる．RKA は通常の攻撃より広いクラスの攻撃を考えているため，RKA に対する安全性は通常の攻撃に対する安全性よりも強い．なお，RKD 関数として恒等写像のみを考える場合には，RKA 安全性は通常の安全性と同等である．

これまでに指摘されたように [12, 27]，RKA 安全性は，デバイス自体に秘密情報を保存し暗号アルゴリズムを走らせるようなスマートカードなどのデバイスにおいて特に重要である．というのも，フォルトインジェクション攻撃などの対象となる可能性が高いためである．それゆえに，RKA を考慮しない既存の安全性定義で安全だと示された暗号要素技術が，果たして RKA 安全性の意味で安全なのかどうかという問いは，重要である．

RKA 安全性は，攻撃者が秘密鍵に変更を加える際に許される変更に対応した RKD 関数に関して定義されるが，これまでに様々な RKD 関数を対象とし，様々な暗号要素技術に対して RKA 安全性は考えられてきている [1, 3, 6, 28, 39, 41, 33, 32]．

1.1.2 非対話型鍵交換について

非対話型鍵交換（Non-Interactive Key Exchange，略して NIKE）は二者が公開鍵の交換以外のやりとりをせずに共有鍵を計算することのできる公開鍵暗号要素技術である．最も初期のよく知られた方式は Diffie と Hellman [14] によって提案された．Cash，Kiltz，および Shoup [11] は NIKE の安全性を定式化した．それは今では CKS 安全性 [23] と呼ばれる．この安全性定義において攻撃者は，様々なユーザのペアが共有した鍵に関する知識のみを用いて，新たなユーザのペアの共有鍵と乱数との識別をするよう要求される．Freire ら [23] は CKS 安全性定義の 3 つの改良版の安

全性を定式化した．それらはCKS-light，CKS-heavy，およびm-CKS-heavy安全性と書かれる．これらはオリジナルのCKS安全性と，攻撃者がアクセスできるオラクルのタイプや許されるオラクルクエリの回数などの点で異なる．彼らは，CKS安全性とその改良版の安全性が等価であることを示した．さらに，CKS-light安全なNIKE方式を設計した（安全性定義が等価であることから，その他の安全性も満たす）．このような安全性定義が提案されて以来，NIKE方式はさらに研究されてきた．PointchevalおよびSanders [35] はNIKEに対してforward安全性を定義し，多重線形写像を用いてforward安全なNIKE方式を提案した．Freire，Hesse，およびHofheinz [22] は，NIKE方式のモジュール分析（要素技術の組合せの妥当性などの分析）や他の要素技術を構築する際にNIKE方式を用いることのモジュール分析をサポートしたNIKE方式の定義を提案した．

なお，NIKE方式は様々な暗号要素技術に応用可能であり，例えば公開鍵暗号方式 [23]，検証者指定署名方式（designated verifier signature schemes） [29]，サインクリプション（signcryption） [31] や，否認可能認証 [15] などが応用例として挙げられる．以上のようにNIKEは応用範囲の広い技術であるため，これまで広く研究者コミュニティにおいて受け入れられてきた既存の安全性を満たしていることが示されたNIKE方式であっても，その方式がRKAを考慮にいれた際に安全になりうるかどうかを明らかにすることは大切である．

1.1.3 署名方式について

署名方式は，メッセージの正当性を保証するための，公開鍵暗号要素技術である．署名方式としては現在までに，Schnorr署名方式 [40], DSA [34], ElGamal署名方式 [18] など多くの方式が提案されてきた. 署名方式に関して広く受け入れられている安全性定義は，選択メッセージ攻撃（chosen message attack）に対する存在的偽造不可能性（existential unforgeability）である．この定義は，攻撃者が自身の選んだメッセージに対する署名を得られるという状況においても，新たなメッセージに対して偽造を行うこと（正当な署名を出力すること）が不可能であることを保証するものである．Schnorr署名方式および，DSAの2つの変型，そして修正版ElGamal署名方式は離散対数仮定のもと，ランダムオラクルモデルにおいてこの安全性定義を満たすことが示されている [36, 37, 38].

署名方式に対するRKAでは，攻撃者は正当なメッセージと署名のペアを複数得られるだけでなく，変更の加わった署名鍵を用いて作られた署名をも得ることができる．なお，RKD関数を用いて署名鍵に変更を加えることができる攻撃者を考える．

RKA安全な署名方式の一般的な構成法がいくつか提案されてる．Bellare，Cash，およびMiller [3]はRKA安全な暗号要素技術間の関係について研究し，特にRKA安全な擬似ランダム関数（pseudo random function, PRF）を用いて，通常の安全性をみたす署名方式をRKA安全な署名方式に変換する方法を示した．その変換は比較的シンプルであり，まず検証鍵および署名鍵を生成する前に，PRFを鍵生成に用いられる乱数に適用し，生成された署名鍵を保存する代わりに乱数を保存しておく．すると，もとの方式の署名鍵はもはや保存されておらず，メッセージに署名する際は常に再生成しなければならなくなる．これはPRFを，保存された乱数に適用し，再び鍵生成アルゴリズムを実行することでなされる．Bellare, Cash, およびMiller [3]は，この変換によってPRFのRKA安全性を署名方式にまで適応させることができることを示した．Abdallaら [1]によって提案された，q-Diffie Hellman Inversion仮定のもとでRKA安全なPRFと組合せることによって，この変換によって（通常の）署名方式を多項式に関するRKA安全な署名方式にすることができる．

Goyalら [27]は，RKA安全な署名方式を構成する同様の変換を示したが，これはcorrelated-input安全なハッシュ関数（CISハッシュ関数）に基づいている．さらに， Goyalらはq-Diffie Hellman Inversion仮定のもと安全で効率の良いCISハッシュ関数を構成した．そしてCISハッシュ関数を用いて多項式に関してRKA安全な署名方式を構成することができることを示した．しかしながら，この構成法は選択的安全性を達成するだけである．選択的安全性とは，攻撃者には署名方式の検証鍵を見る前に今後使う全てのRKD関数を決定することを要求するという安全性である．この安全性は，RKD関数を利用するごとに，これまでの情報に基づいて用いるRKD関数をその都度決定することを攻撃者に許す適応的な安全性定義（本論文で扱う）と比較すると，弱い安全性である．

non-malleableな鍵導出関数（nm-KDFs） [21]の研究に基づいて，Qinら [39]はcontinuous nm-KDFを定義した．これを利用してBellare, Cash, およびMiller [3]がPRFを用いて行った上記と同様の変換を，continuous nm-KDFを代わりに利用することで行い，標準的な仮定のもとで多項式に関してRKA安全な署名方式を構成

した．Damgård ら [13] は，秘密鍵が乱数である暗号要素技術は簡素な変換で RKA 安全になることを示したが，この変換は nm-KDF [21, 39] を秘密鍵に適用していると見なすことが出来る．しかしこの方法は必要となる計算量が多く，実行スピートの面でパフォーマンスが低い．

署名方式は必要不可欠な暗号技術であるため，様々な構成法の RKA 安全性を明らかにすることは，実用および理論の両面において興味深いことである．特に，Schnorr 署名や DSA のようによく知られた署名方式の RKA 安全性を調べることは，それらの方式が広く使われているため重要である．しかし，Bao ら [2] による否定的な結果，つまり Schnorr 署名方式と DSA はビットフリッピングに関する RKA に対して安全ではないという結果以外には，どのような RKA 安全性を持つのかについて知られていない．

1.2 貢献

NIKE 方式に関しては，3.2 章において，まず NIKE 方式に対する RKA 安全性を定義する．具体的には，RKA-CKS-light，RKA-CKS，RKA-CKS-heavy，および RKA-m-CKS-heavy security を Freire ら [23] によって提案された安全性に基づいて定義する．それから，これら安全性定義間の関係について図 1.1 内で示した章において示す : RKA-CKS，RKA-CKS-heavy，および RKA-m-CKS-heavy 安全性は全て

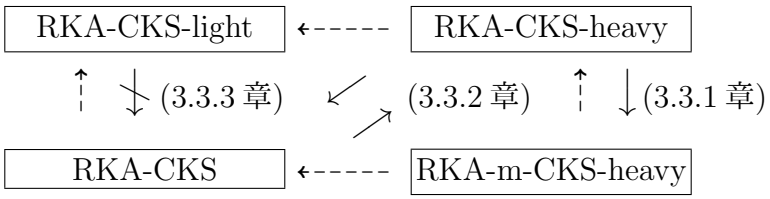

図 1.1: 本論文で示す NIKE の RKA 安全性定義間の関係．
点線での矢印は自明な含意関係である．直線での矢印は本論文で示す含意関係を表し，否定の矢印は本論文で含意関係を満たさない方式の存在を示すことを表す．

の RKD 関数に関して等価であるが，RKA-CKS-light 安全性は他の 3 つの安全性とは，線形関数に関して分離がある．対照的に，Freire らは [23] において，RKA を考慮しない NIKE の通常の安全性定義は全て等価であると示している．最後に Freire

ら [23] の提案した NIKE 方式の一つが，線形関数に関する RKA-CKS 安全である（よって， RKA-CKS-heavy 安全 RKA-m-CKS-heavy 安全でもある）ことを符号付剰余群上の Diffie-Hellman 仮定のもと，ランダムオラクルモデルにおいて証明する．この仮定は NIKE 方式の通常の安全性証明の際に Freire らが使ったものと同じ仮定であり，素因数分解仮定から導かれる．

署名方式に関しては，RKA を考慮しない既存の安全性に対して，表 1.1 のような結果が知られている．なお表において「✓」はその方式が安全であることを,「×」は安全でないことを，また「-」は証明がついていないことを示す．本論文では 4.4 章において，まず Schnorr 署名方式，DSA の変型，そして ElGamal 署名方式の変型は，RKA 署名オラクルにクエリされたメッセージを偽造の一部に用いてはならないという弱い RKA（wRKA）の安全性定義に対して安全であることを示す（表 1.2 の「✓」）．具体的には以下の結果を示す：

- Schnorr 署名方式は多項式に関する wRKA に対して安全である
- よく知られた DSA の変型 [38] は，多項式に関する wRKA に対して安全である．
- ElGamal 署名方式の変形（修正 ElGamal 署名方式 [36]）は多項式に関する wRKA に対して安全である．

次に 4.5 章において，Schnorr 署名方式，オリジナルの DSA，および ElGamal 署名方式は線形関数に関する RKA に対して脆弱であることを示す（表 1.2 の「×」）．その後，Schnorr 署名方式，DSA，および ElGamal 署名方式に基づいて,（通常の）RKA 安全な署名方式を構成する．具体的には，以下の 3 つの結果を記す（参照：表 1.3 の ✓）: どれも d-strong 離散対数（d-SDL）仮定のもとランダムオラクルモデルにて証明される．

- Schnorr 署名方式の署名アルゴリズムおよび検証アルゴリズムへ軽微な修正を加えると，多項式に関する RKA 安全な方式になる
- DSA の署名アルゴリズムおよび検証アルゴリズムへ軽微な修正を加えると，多項式に関する RKA 安全な方式になる
- ElGamal 署名方式の署名アルゴリズムおよび検証アルゴリズムへ軽微な修正を加えると，多項式に関する RKA 安全な方式になる

表 1.1: 既存の署名方式の既存の安全性（既存の結果）

	Schnorr	DSA	DSA の変型	ElGamal	修正 ElGamal
既存の安全性	✓	-	✓	×	✓

表 1.2: 既存の署名方式の RKA 安全性（本論文の結果）

	Schnorr	DSA	DSA の変型	ElGamal	修正 ElGamal
wRKA（多項式）	✓	-	✓	-	✓
RKA（線形）	×	×	-	×	-

表 1.3: 本論文で提案する署名方式の RKA 安全性（本論文の結果）

	改良 Schnorr	改良 DSA	改良 ElGamal
RKA（多項式）	✓	✓	✓

言い換えると，Schnorr 署名方式は，多項式に関する wRKA に対して安全である一方で，線形関数という弱い攻撃に対してさえ RKA 安全ではないが，さらに方式に軽微な修正を加えることで多項式に関する RKA に対して安全になる．DSA は線形関数に関する RKA 安全ではないが，[38] による DSA の変型は多項式に関する wRKA に対して安全であり，さらにこの方式に修正を加えることで多項式に関する RKA 安全性を満たす方式になる．ElGamal 署名方式は線形関数に関する RKA 安全ではないが，[36] による修正 ElGamal 署名方式は多項式に関する wRKA 安全であり，さらにこの方式に軽微な修正を加えることで多項式に関する RKA に対して安全になる．修正を加えた方式，つまり改良 Schnorr 署名方式，改良 DSA，改良 ElGamal 署名方式が多項式に関する RKA 安全であることは，d-SDL 仮定のもとランダムオラクルモデルで証明される．系として，改良 Schnorr 署名方式，改良 DSA，改良 ElGamal 署名方式は，アフィン関数に関する RKA 安全であることが，標準的な離散対数問題仮定のもとで証明される．というのも，1-SDL 仮定は離散対数仮定のことであり，1 次の多項式はアフィン関数に他ならないためである．

特筆すべきは，Schnorr 署名方式，DSA，ElGamal 署名方式に対して本論文で施した修正では，署名の際の累乗計算が 1 つ増えるだけであり，検証アルゴリズム，

署名長，および鍵長は変わらないことである．よって，continuous nm-KDF [28, 39] や RKA 安全な PRF [3, 1] に基づいた変換と比較して，本論文で提案する修正は元の Schnorr 署名方式，DSA，ElGamal 署名方式の効率を落とすことがない．さらに，RKA 安全性を達成するための上述の変換とは異なり，提案する修正では検証鍵や署名鍵に変更を加えることがない．これは，DSA のようにすでに身の回りで広く実装され利用されている方式にとっては，鍵管理と検証鍵の証明書がそのままでよいため，利点である．最後に，本論文で提案する改良 Schnorr 署名方式，改良 DSA，および改良 ElGamal 署名方式に対する安全性証明において，RKA 署名オラクルへのクエリ回数について何の制限も置いておらず，署名鍵がタンパリングされたことがわかればさらなるクエリを攻撃者から受け付けないという “self-destruct” メカニズム [20, 21] に頼ってもいないことを強調したい．

1.3 関連研究

Gennaro ら [24] は，攻撃者が暗号要素技術の鍵を任意にタンパリング出来るという仮定をして，要素技術のほとんど全ての鍵を回復する方法を示した．これは RKD 関数のどんな集合に対しても RKA 安全性が示せるわけではないことを意味している．一方で，Damgård ら [12, 13] は，攻撃者のできる RKA クエリ数に制限を設けた安全性モデルにおいて，任意の RKD 関数に関する安全性を達成できることを示した．制限漏洩・タンパリングモデル（bounded leakage and tampering model）と呼ばれるこのモデルとは対照的に，本論文では，任意回の RKA 署名オラクルクエリを許された，制限されない攻撃者を考える．non-malleable code [17] は，Dziembowski，Pietrzak，および Wichs が導入して以来研究され，RKA 安全な暗号系の構築に応用できることがわかってきた．non-malleable code だけでは RKA 安全性を達成するのに十分ではないが，[20] において導入された continuous non-malleable code を用いれば可能である．しかし，[20] で示された構成法の安全性は，self-destruct メカニズムに依っている．これはシステムの内部状態がひとたびタンパリングされたら，攻撃者がシステムにアクセスできないようにするメカニズムである．それとは対照的に，Qin ら [39] によって提案された continuous nm-KDF は self-destruct メカニズムを必要とせず，広いクラスの RKD 関数に関して RKA 安全な公開鍵暗号要素技

術を構成するのに用いることが出来る．Jafargholi および Wichs [28] は continuous nm-KDF の安全性について 2 つの要因を定義した：(I) 2 回目以降のタンパリングも常にオリジナルの秘密鍵に対して適用されるのか，それとも 2 回目以降はそれ以前に変更された秘密鍵に対してさらに適用されるのか（どちらのタイプかによって，persistent もしくは non-persistent と分類される）．(II) invalid な秘密鍵にタンパリングした際に “self-destruct” するか，それともしないか．そしてこの 2 つの要因から 4 つの安全性のレベルを考えた．また，Bellare，Cash，および Miller [3] は，任意の ID ベース暗号方式から，RKA 安全な署名が構成できることを示した．Goyal ら [27] は，Boneh-Boyen 署名方式 [10] が，ある種の多項式からなる RKD 関数のクラスに関する RKA 安全性を満たすことを示した．

注意したいのは，Bernstein ら [7] による EdDSA という署名方式と Koblitz および Menezes [30] による ECDSA$^+$ という署名方式はそれぞれ，ハッシュ関数への入力のひとつが検証鍵であるという意味において，本論文の 4.6.1 章および 4.7.1 章で提案する方式にそれぞれよく似ているということである．しかしながら，[7] および [30] の方式は，研究の異なる文脈で提案され，RKA 安全性は考慮していない．

1.4 本論文の構成

2 章において，本論文を通して必要となる記法，計算量的仮定，およびその他の基本的概念について記す．3 章において，NIKE についてとりあげ，その RKA 安全性を複数定義し，それらの関係について論じる．また具体的な NIKE 方式の RKA 安全性について論じる．4 章において，署名方式についてとりあげ，既存の RKA 安全性について紹介し，具体的な署名方式における RKA 安全性について論じる．5 章において本論文をまとめる．

第2章　準備

2.1　記法

本論文を通して，以下の記法を用いる：自然数の集合 $\mathbb{N}$ に対して，$\lambda \in \mathbb{N}$ をセキュリティパラメータとする．これは，安全性の指標となる値であり，λ 個の 1 からなる一進表記 1^λ とも書かれる．本論文においては慣習に従いその表記を使い分けるが，意味することは同一である．明示的に書かれない場合でも，アルゴリズムの入力としてセキュリティパラメータをとる．

任意の多項式の逆数よりも早く収束するならば，関数 $F : \mathbb{N} \to \mathbb{R}$ は無視可能であるという．

イベント B が起こった後に，述語 A が真であるときの確率を $\Pr[A : B]$ と書く．

安全性定義 $\mathbf{A}$ が安全性定義 $\mathbf{B}$ を含意するとき，“$\mathbf{A} \Rightarrow \mathbf{B}$” と書く：つまり，方式が安全性定義 $\mathbf{A}$ を満たすならば，その方式は安全性定義 $\mathbf{B}$ を満たす．一方で，安全性定義 $\mathbf{A}$ を満たす方式があるという仮定のもとで，安全性定義 $\mathbf{A}$ を満たすが安全性定義 $\mathbf{B}$ を満たさない方式の構成法が存在するとき，“$\mathbf{A} \not\Rightarrow \mathbf{B}$” と書く．

RSAgen を，入力として 1^λ をとり，ビット長 λ の相異なる素数 p および q に対して $n = pq$ かつ $p \equiv q \equiv 3 \pmod 4$ となる組 (n, p, q) を生成する確率的多項式時間アルゴリズムとする．ここで $p \equiv q \equiv 3 \pmod 4$ は，p, q および 3 が 4 を法として合同であることを表わす．このような n に対して，$\mathbb{Z}_n$ と書いて集合 $\{-(n-1)/2, -(n-3)/2, \ldots, 0, \ldots, (n-3)/2, (n-1)/2\}$ を表し，$\mathbb{Z}_n^*$ と書いて $\mathbb{Z}_n$ の元のうち n と互いに素となるものの集合を表わす．また，$\mathbf{QR}_n = \{x \in \mathbb{Z}_n^* \mid \exists y \in \mathbb{Z}_n^*, y^2 \equiv x \pmod n\}$ および $\mathbf{QR}_n^+ = \{|x| \mid x \in \mathbf{QR}_n\}$ とする．なお，$\mathbf{QR}_n^+$ に含まれるかどうかは効率的に確認でき，$(\mathbf{QR}_n^+, \cdot)$ は位数 $\varphi(n)/4$ の巡回群である．ここで，φ はオイラー関数である．

有限集合 S に対して，$x \xleftarrow{\$} S$ と書いて，x が S から一様ランダムに選ばれることを表わす．アルゴリズム $\mathcal{A}$ およびオラクル $\mathcal{O}$ に対して，$\mathcal{A}^{\mathcal{O}}$ と書いて，$\mathcal{A}$ が $\mathcal{O}$ に

アクセス可能であることを表わす.

2.2 計算量的仮定

本論文で用いる計算量的仮定について記す.

2.2.1 素因数分解仮定

2.1 章で記したように，RSAgen を入力として 1^λ をとり，組 (n,p,q) を出力する確率的多項式時間アルゴリズムとする．ここで p および q はビット長 λ の相異なる素数で $p \equiv q \equiv 3 \pmod 4$ を満たし $n = pq$ である．RSAgen で生成された n を素因数分解するアルゴリズム $\mathcal{A}$ の利得を，$\mathcal{A}$ が素因数分解に成功する確率で定める：

$$\begin{aligned}&\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(\lambda)\\ &\stackrel{\mathrm{def}}{=} \Pr\left[\mathcal{A}(n) = \{p,q\} : (n,p,q) \stackrel{\$}{\leftarrow} \mathsf{RSAgen}(1^\lambda)\right].\end{aligned}$$

RSAgen に対する素因数分解仮定は，任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して，$\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(\lambda)$ が無視可能となることである.

2.2.2 Double Strong Diffie-Hellman (DSDH) 仮定 [23]

$(n,p,q) \stackrel{\$}{\leftarrow} \mathsf{RSAgen}(1^\lambda)$ とし，$g \in \mathbf{QR}^+_n$ をランダムに選ばれた生成元，$X = g^x \in \mathbf{QR}^+_n, Y = g^y \in \mathbf{QR}^+_n$ をランダムに選ばれた元とする．$\hat{Y} \in \mathbf{QR}^+_n$ および $\hat{Z} \in \mathbf{QR}^+_n$ に対して，$\mathrm{DDH}_{g,X}(\hat{Y},\hat{Z})$ を，$\hat{Y}^x = \hat{Z}$ ならば 1 を出力し，そうでなければ，0 を出力するアルゴリズムとする．アルゴリズム $\mathcal{A}$ に対して，

$$\mathrm{Adv}^{\mathrm{dsdh}}_{\mathcal{A},\mathsf{RSAgen}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr\left[\mathcal{A}^{\mathcal{O}}(n,g,X,Y) = g^{xy}\right]$$

とする．ここで，$\mathcal{O} = \{\mathrm{DDH}_{g,X}(\cdot,\cdot), \mathrm{DDH}_{g,Y}(\cdot,\cdot)\}$ である．符号付剰余群上の RSAgen に対する DSDH 仮定とは，任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して，$\mathrm{Adv}^{\mathrm{dsdh}}_{\mathcal{A},\mathsf{RSAgen}}(\lambda)$ が無視可能となることである.

2.2.3 素因数分解仮定と DSDH 仮定との関係

素因数分解仮定と DSDH 仮定に関する以下の事実は符号付剰余群を考えるときに成り立つことが知られている：

命題 1 ([23]). RSAgen に対する素因数分解仮定が成り立つならば，符号付剰余剰余群に関する RSAgen に対する DSDH 仮定も成り立つ.

2.2.4 d-Strong 離散対数仮定

Goyal ら [27] によって提案された d-strong 離散対数（d-SDL）仮定を記す．d を自然数とする．d-SDL 問題とは，入力 $(g, g^x, g^{x^2}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$ を与えられて，x を計算する問題である．ここで，$\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ であり，$x \xleftarrow{\$} \mathbb{Z}_q$ である．なお本論文において，$\mathbb{Z}_q$ は $\mathbb{Z}_n$ と記述が異なることに注意が必要である.

$\mathbb{G}$ 上の d-SDL 問題を解く攻撃者 $\mathcal{A}$ に対して，利得を以下のように定める：

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{A},\mathbb{G}}(\lambda) = \Pr\left[x' = x : \begin{array}{l} g \xleftarrow{\$} \mathbb{Z}_q \\ x \xleftarrow{\$} \mathbb{Z}_q \\ x' \leftarrow \mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^d}) \end{array} \right].$$

ここで g は $\mathbb{Z}_q$ の生成元である．$\mathbb{G}$ 上の d-SDL 仮定は，利得 $\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{A},\mathbb{G}}(\lambda)$ が，どんな多項式時間アルゴリズム $\mathcal{A}$ に対しても無視可能であることを言う.

1-SDL 仮定は通常の離散対数仮定と等価であることは明らかである．d-Strong Diffie-Hellman 問題 [10] と同様に，d-SDL 問題は通常の離散対数問題より易しい問題である.

2.3 一般 Forking Lemma

Bellare および Neven [5] は，Pointcheval および Stern [36, 37] によって署名方式の安全性証明のために導入された forking lemma を一般化した.

補題 1 ([5]). 整数 $Q \geq 1$ およびサイズ $q \geq 2$ の集合 Z を固定する．IG をビット列 X を出力する入力生成器（input generator）と呼ばれる確率的アルゴリズムとする．確率的アルゴリズム $\mathcal{F}$ は入力として $X, h_1, \ldots, h_Q$ をとり，整数 J および副出

力 V を出力するとする．ここで，$h_1, \ldots, h_Q \in Z$ であり，整数 J は $0, \ldots, Q$ の範囲である．

acc は次の確率を表すものとする：

$$\mathbf{acc} = \Pr\left[\begin{array}{ll} & X \xleftarrow{\$} \mathrm{IG} \\ J \geq 1 : & h_1, \ldots, h_Q \xleftarrow{\$} Z \\ & (J, V) \xleftarrow{\$} \mathcal{F}(X, h_1, \ldots, h_Q) \end{array}\right].$$

$\mathcal{F}$ に関する forking アルゴリズム $\mathcal{B}_{\mathcal{F}}$ は，入力として X をとり，以下のように動作する確率的アルゴリズムである：

1. Pick randomness $\rho_{\mathcal{F}}$ for $\mathcal{F}$ at random
2. $h_1, \ldots, h_Q \xleftarrow{\$} Z$
3. $(I, V) \leftarrow \mathcal{F}(X, h_1, \ldots, h_Q; \rho_{\mathcal{F}})$
4. If $I = 0$ then return $(0, \perp, \perp)$
5. $h'_1, \ldots, h'_Q \xleftarrow{\$} Z$
6. $(I', V') \leftarrow \mathcal{F}(X, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_Q; \rho_{\mathcal{F}})$
7. If ($I = I'$ and $h_I \neq h'_I$) then return $(1, V, V')$
8. Else return $(0, \perp, \perp)$

$\mathcal{B}_{\mathcal{F}}$ が $(1, V, V')$ を出力する確率を

$$\mathbf{frk} = \Pr[\delta = 1 : X \xleftarrow{\$} \mathrm{IG}; (\delta, V, V') \xleftarrow{\$} \mathcal{B}_{\mathcal{F}}(X)]$$

とする．このとき，

$$\mathbf{frk} \geq \mathbf{acc} \cdot \left(\frac{\mathbf{acc}}{Q} - \frac{1}{q}\right)$$

である．

この補題は，アルゴリズム $\mathcal{F}$ が乱数 $\rho_{\mathcal{F}}$ および入力 $X, h_1, \ldots, h_Q$ に対して，ある性質をもつ値（ここでは $I \neq 0$ なる (I, V)）を無視できない確率で出力するならば，同じ乱数を用い新たな入力に対して再度アルゴリズム $\mathcal{F}$ を走らせると，無視できない確率で同様の性質を持った出力を返すことを表している．この補題は，4 章における署名方式の安全性証明において利用される．

2.4 関連鍵攻撃

関連鍵攻撃（related-key attack, RKA）においては，攻撃者がハードウェアに蓄えられた秘密鍵を電磁波を用いて改ざんし，その変更された鍵を使ったアルゴリズムの出力を得るという現実の攻撃を捉えている．これはタンパリングやフォルトインジェクション攻撃と呼ばれ，RKA 安全性は攻撃者が変更された鍵で作られたアルゴリズムの出力を得られるようになった安全性ゲームで定式化される．そこで，攻撃者の秘密鍵への変更は，関連鍵導出関数（related-key deriving function, RKD 関数） [4] と呼ばれる関数で捉えられる．

RKD 関数は関数 $\phi : \Gamma \to \Gamma$ であり，ここで Γ は公開パラメータに依存して決まる集合である．本論文では，秘密鍵空間 $\mathcal{SK}$ は Γ の部分集合であるとする．つまり，$\mathcal{SK} \subseteq \Gamma$ を満たす．また，Γ は代数構造を持つ（群，環，体など）とする．このような設定のもとでは，秘密鍵空間が Γ の真部分集合であった場合，つまり $\mathcal{SK} \subsetneq \Gamma$ であった場合，変更された秘密鍵 $\phi(sk)$ は秘密鍵空間に含まれないこともあり得る．この場合，変更された秘密鍵を用いるアルゴリズムが実行できないならば関連するオラクルの返答は $\perp$ と定義する．変更された秘密鍵を用いるアルゴリズムが実行できるならば，変更された秘密鍵が秘密鍵空間に含まれるかどうかに関わらず，関連するオラクルはそのアルゴリズムの出力を返すことにすればよい．Φ を RKD 関数のクラスとする．RKD 関数クラス Φ は，攻撃者が秘密鍵を改ざんするのに許される演算操作からなる．通常，Φ は恒等写像 id を含むと仮定され，RKA 安全性は既存の安全性を含む．なお，関数がクラス Φ に含まれるかどうかをチェックするのは容易であり，RKD 関数は効率的に計算可能だとする．

[6] に従って，3 種の RKD 関数，線形関数，アフィン関数，多項式，を考える．

線形関数. $(\Gamma, *)$ を群とする．線形関数のクラスは以下のように定義される：$\Phi^{\mathrm{lin}} = \{\phi_\Delta \mid \Delta \in \Gamma\}$. ここで，元 $k \in \Gamma$ に対して，$\phi_\Delta(k) = k * \Delta$ である．“$*$” は考える群によって加算もしくは乗算を表す演算子である．つまり，本論文では [6] の記述に従い，Γ が加法群であれば RKD 関数として考える線形関数は $\phi_\Delta(k) = k + \Delta$ と表せる関数である．

アフィン関数. Γ を有限体とする．アフィン関数のクラスは以下のように定義される：$\Phi^{\mathrm{aff}} = \{\phi_{a,b} \mid a, b \in \Gamma\}$. ここで，元 $k \in \Gamma$ に対して，$\phi_{a,b}(k) = a \cdot k + b$ で

ある．

多項式. Γ を有限体とする．多項式のクラスは以下のように定義される：$\Phi^{\mathrm{poly}(d)} = \{\phi_f \mid f \in \Gamma_d[x]\}$. ここで，$\Gamma_d[x]$ は Γ 上の最大次数 d の多項式の集合であり，元 $k \in \Gamma$ に対して，$\phi_f(k) = f(k)$ である．

RKA 安全性は，考える RKD 関数が線形関数，アフィン関数，多項式となるにつれて強い安全性であり，達成するのが難しくなる．本論文では，RKD 関数としてこのような代数的演算を持つものを考える．なお，定理 3 および定理 4 においては，$\phi_\alpha(k) = k + \alpha$ という線形関数を用いる．

第3章　非対話型鍵交換

3.1　非対話型鍵交換

Freire ら [23] に従って，以下の 3 つのアルゴリズム，アイデンティティ空間 $\mathcal{IDS}$，秘密鍵空間 $\mathcal{SK}$, および共有鍵空間 $\mathcal{SHK}$ を用いて，“non-interactive key exchange (NIKE)” 方式 **NIKE** を定義する：

$$\mathbf{NIKE} = (\mathsf{CS}, \mathsf{KG}, \mathsf{ShK}).$$

ここで，セキュリティパラメータ λ，公開パラメータの集合 $params$，アイデンティティ $\mathrm{ID} \in \mathcal{IDS}$，公開鍵 pk，秘密鍵 $sk \in \mathcal{SK}$，そして共有鍵 $K \in \mathcal{SHK}$ に対して，

$$\begin{aligned} params &\xleftarrow{\$} \mathsf{CS}(1^\lambda), \\ (pk, sk) &\xleftarrow{\$} \mathsf{KG}(params, \mathrm{ID}) \\ K_{1,2} &\leftarrow \mathsf{ShK}(\mathrm{ID}_1, pk_1, \mathrm{ID}_2, sk_2) \end{aligned}$$

である．なお，共有鍵のインデックスはアイデンティティに対応する．ShK は決定的アルゴリズムであり，CS および KG は確率的アルゴリズムである．NIKE 方式の正当性のために，以下を満たさなければならない．任意の $\lambda \in \mathbb{N}$，$\mathsf{CS}(1^\lambda)$ の出力である任意の $params$，任意の $\mathrm{ID}_1, \mathrm{ID}_2 \in \mathcal{IDS}$，$\mathsf{KG}(params, \mathrm{ID}_1)$ の出力である任意の (pk_1, sk_1)，$\mathsf{KG}(params, \mathrm{ID}_2)$ の出力である任意の (pk_2, sk_2) に対して，

$$\mathsf{ShK}(\mathrm{ID}_1, pk_1, \mathrm{ID}_2, sk_2) = \mathsf{ShK}(\mathrm{ID}_2, pk_2, \mathrm{ID}_1, sk_1)$$

である．

3.2　非対話型鍵交換の RKA 安全性

Cash ら [11] および Freire ら [23] によって提案された，NIKE に対する安全性定義に基づいて，NIKE に対する 4 つの RKA 安全性定義を導入する．まず，最も強

い RKA 安全性モデルである Φ-RKA-m-CKS-heavy 安全性を 3.2.1 章で説明する．それから，その他 3 つの RKA 安全性モデルである，Φ-RKA-CKS-heavy 安全性，Φ-RKA-CKS 安全性，および Φ-RKA-CKS-light 安全性について 3.2.2 章で説明する．

3.2.1 Φ-RKA-m-CKS-heavy

この安全性定義を，攻撃者 $\mathcal{A}$ とオラクルとの間のエクスペリメントを用いて図 3.1 のように定義する．まず，CS は入力として 1^λ をとり，公開パラメータの集合 $params$ を出力し，そしてこれは攻撃者 $\mathcal{A}$ に与えられる．それから，ランダムビット b が選ばれる．攻撃者 $\mathcal{A}$ が b' を出力するまで，オラクルは $\mathcal{A}$ からのクエリに答える．ここで，$\mathcal{A}$ がアクセスするオラクルは $\mathcal{O}$= {Reg.Hon, Reg.Cor, Extract, Hon.Rev, Cor.Rev, Test} であり[1]，動作は図 3.2 に示す．攻撃者 $\mathcal{A}$ がそれぞれのオラクルにアクセスできることは，以下のような現実の状況および攻撃と対応付けられる．

- Reg.Hon は，攻撃の対象として，攻撃者が秘密鍵を知らないユーザがいる状況を作る．クエリするごとに，このようなユーザが増える．
- Reg.Cor は，攻撃の対象として，攻撃者が秘密鍵を知っているユーザがいる状況を作る．クエリするごとに，このようなユーザが増える．
- Extract は，秘密鍵を知らないユーザに対して行い，秘密鍵を得られる攻撃．
- Hon.Rev は，秘密鍵を知らないユーザ間の共有鍵計算の結果を得られる攻撃．
- Cor.Rev は，秘密鍵を知らないユーザおよび秘密鍵を知っているユーザ間の共有鍵計算の結果を得られる攻撃．
- Test は，安全性証明特有のオラクルであり，秘密鍵を知らないユーザ間の共有鍵計算の結果もしくは乱数を得られる攻撃．攻撃者は共有鍵計算の結果なのか乱数なのかを識別する必要があり，識別できなければその方式は安全だといえる．

[1] Reg.Hon，Reg.Cor，Hon.Rev，および Cor.Rev はそれぞれ “register honest”，“register corrupt”，“honest reveal”，そして “corrupt reveal” の略である．

Φ-RKA-m-CKS-heavy
$params \xleftarrow{\$} \mathsf{CS}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$ ▷ used by the Test oracle
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(params)$

図 3.1: Φ-RKA-m-CKS-heavy 安全性エクスペリメント

では，図 3.2 のオラクルの動作の詳細を説明する．以下のオラクルの記述では，L_{H}，L_{C}，L_{E}，L_{HR}，および L_{T} はそれぞれ，オラクル Reg.Hon，Reg.Cor，Extract，Hon.Rev，および Test へのクエリを記録するリストである．

Reg.Hon オラクルは以下のように動作する：まず，入力としてユーザアイデンティティ $\mathrm{ID} \in \mathcal{IDS}$ をとる．もし ID が Reg.Cor オラクルか Reg.Hon オラクルにクエリされていたら，$\perp$ を返す．そうでなければ，ユーザの公開鍵と秘密鍵を $(pk, sk) \xleftarrow{\$} \mathsf{KG}(params, \mathrm{ID})$ によって作り，L_{H} に (ID, pk, sk) を記録することにより，アイデンティティ ID および鍵を honest として登録する．そして pk を $\mathcal{A}$ に返す．L_{H} に記録されているアイデンティティを “honest” ユーザアイデンティティと呼ぶことにする．

Reg.Cor オラクルは入力としてユーザアイデンティティ $\mathrm{ID} \in \mathcal{IDS}$ および公開鍵 pk をとる．もし ID が Reg.Hon オラクルにクエリされていたら，$\perp$ を返す．そうでなければ，L_{C} に (ID, pk) を記録することにより，それらを corrupt として登録する．なお，同じ ID が Reg.Cor オラクルにクエリされたならば，最も新しいものだけが L_{C} に記録される．L_{C} に記録されているアイデンティティを “corrupt” ユーザアイデンティティと呼ぶことにする．corrupt と登録されたアイデンティティが，後になって honest として登録されることはできず，またその逆もできないこととする．

Extract オラクルは入力として Test クエリに使われていない honest ユーザアイデンティティ ID をとり，ID をリスト L_{E} に記録し，L_{H} に記録されている対応する秘密鍵を返す．

Hon.Rev オラクルは，入力として Test オラクルにクエリされていない組 $((\mathrm{ID}_1, \mathrm{ID}_2), \phi)$ をとり，その組をリスト L_{HR} に記録し，$K_{1,2,\phi} \leftarrow \mathsf{ShK}(\mathrm{ID}_1, pk_1, \mathrm{ID}_2, \phi(sk_2))$ を返す．ここで，$\mathrm{ID}_1 \in \mathcal{IDS}$ および $\mathrm{ID}_2 \in \mathcal{IDS}$ は honest ユーザアイデンティティで，$\phi \in \Phi$ は RKD 関数である．

Cor.Rev オラクルは，入力として corrupt ユーザアイデンティティ $\mathrm{ID}_1 \in \mathcal{IDS}$，

oracle $\mathsf{Reg.Hon}$ (ID)
if $(\mathrm{ID}, *) \in L_{\mathrm{C}}$ or $(\mathrm{ID}, *, *) \in L_{\mathrm{H}}$ **then**
return $\perp$
else
$(pk, sk) \xleftarrow{\$} \mathsf{KG}(params, \mathrm{ID})$
record (ID, pk, sk) in L_{H}
return pk

---

oracle $\mathsf{Reg.Cor}$ (ID, pk)
if $(\mathrm{ID}, *, *) \in L_{\mathrm{H}}$ or $(\mathrm{ID}, *) \in L_{\mathrm{C}}$ **then**
return $\perp$
else
record (ID, pk) in L_{C}

---

oracle $\mathsf{Extract}$ (ID)
if ID is stored in L_{T} or $(\mathrm{ID}, *, *) \notin L_{\mathrm{H}}$ **then**
return $\perp$
else
record ID in L_{E}
return sk from (ID, pk, sk) in L_{H}

---

oracle $\mathsf{Hon.Rev}$ $(\mathrm{ID}_1, \mathrm{ID}_2, \phi)$
if $((\mathrm{ID}_1, \mathrm{ID}_2), \phi, *) \in L_{\mathrm{T}}$ or $(\mathrm{ID}_1, *, *) \notin L_{\mathrm{H}}$ or $(\mathrm{ID}_2, *, *) \notin L_{\mathrm{H}}$
or $((\phi = \mathrm{id})$ and $((\mathrm{ID}_2, \mathrm{ID}_1), \phi, *) \in L_{\mathrm{T}})$ **then**
return $\perp$
else
record $((\mathrm{ID}_1, \mathrm{ID}_2), \phi)$ in L_{HR}
$\mathrm{K}_{1,2,\phi} \leftarrow \mathrm{ShK}(\mathrm{ID}_1, \mathrm{pk}_1, \mathrm{ID}_2, \phi(\mathrm{sk}_2))$
return $\mathrm{K}_{1,2,\phi}$

---

oracle $\mathsf{Cor.Rev}$ $(\mathrm{ID}_1, \mathrm{ID}_2, \phi)$
if $(\mathrm{ID}_1, *) \notin L_{\mathrm{C}}$ or $(\mathrm{ID}_2, *, *) \notin L_{\mathrm{H}}$ **then**
return $\perp$
else
$\mathrm{K}_{1,2,\phi} \leftarrow \mathrm{ShK}(\mathrm{ID}_1, \mathrm{pk}_1, \mathrm{ID}_2, \phi(\mathrm{sk}_2))$
return $\mathrm{K}_{1,2,\phi}$

---

oracle Test $(\mathrm{ID}_{\mathrm{A}}, \mathrm{ID}_{\mathrm{B}}, \phi^*)$
if $((\mathrm{ID}_A, \mathrm{ID}_B), \phi^*) \in L_{\mathrm{HR}}$ or $\mathrm{ID}_A \in L_{\mathrm{E}}$ or $(\mathrm{ID}_A, *, *) \notin L_{\mathrm{H}}$ or $\mathrm{ID}_B \in L_{\mathrm{E}}$
or $(\mathrm{ID}_B, *, *) \notin L_{\mathrm{H}}$ or $((\phi^* = \mathrm{id})$ and $((\mathrm{ID}_B, \mathrm{ID}_A), \phi^*) \in L_{\mathrm{HR}})$ **then**
return $\perp$
else
if $b = 0$ **then**
$K_{A,B,\phi^*} \leftarrow \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \phi^*(sk_B))$
else if $((\mathrm{ID}_A, \mathrm{ID}_B), \phi^*, K_{A,B,\phi^*}) \in L_{\mathrm{T}}$ or $(\phi^* = \mathrm{id}$ and $((\mathrm{ID}_B, \mathrm{ID}_A), \phi^*, K_{A,B,\phi^*}) \in L_{\mathrm{T}})$ for some K_{A,B,ϕ^*} **then**
return K_{A,B,ϕ^*}
else
$K_{A,B,\phi^*} \xleftarrow{\$} \mathcal{SHK}$
record $((\mathrm{ID}_A, \mathrm{ID}_B), \phi^*, K_{A,B,\phi^*})$ in L_{T}
return K_{A,B,ϕ^*}

図 3.2: Φ-RKA-m-CKS-heavy 安全性エクスペリメント内のオラクル

honest ユーザアイデンティティ $\mathrm{ID}_2 \in \mathcal{IDS}$，および RKD 関数 $\phi \in \Phi$ をとり，$K_{1,2,\phi} \leftarrow \mathsf{ShK}(\mathrm{ID}_1, pk_1, \mathrm{ID}_2, \phi(sk_2))$ を返す．

Test オラクルは以下のように動作する：入力として組 $((\mathrm{ID}_A, \mathrm{ID}_B), \phi^*)$ をとる．ここで $\mathrm{ID}_A \in \mathcal{IDS}$ および $\mathrm{ID}_B \in \mathcal{IDS}$（$\mathrm{ID_A} \neq \mathrm{ID_B}$）は Extract にクエリされていない honest ユーザアイデンティティであり，ϕ^* は RKD 関数である．$((\mathrm{ID}_A, \mathrm{ID}_B), \phi^*)$ がリスト L_{HR} に記録されているならば，$\perp$ を返す．そうでなければ，ShK の出力もしくは乱数をビット b の値によって以下のように返し，$((\mathrm{ID}_A, \mathrm{ID}_B), \phi^*, K)$ をリスト L_{T} に記録する．ここで，K は返ってきた値である：$b = 0$ ならば，$\mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \phi^*(sk_B))$ を返す．そうでなければ（つまり $b = 1$ ならば），ランダムな鍵を返す．一貫性を保つために，$b = 1$ のエクスペリメントは $(\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{id})$ および $(\mathrm{ID}_2, \mathrm{ID}_1, \mathrm{id})$ の際には同じランダムな鍵を返すものとする．

上記が最も強い RKA 安全性の Φ-RKA-m-CKS-heavy である．このエクスペリメントにおいて，攻撃者は多項式程度の数のクエリが許される．攻撃者 $\mathcal{A}$ は安全性エクスペリメントで使われたビット b を推測しようとする．$\mathcal{A}$ が最終的に b' を出力するとき，$b' = b$ ならば $\mathcal{A}$ の勝ちである．この安全性エクスペリメントにおいて，NIKE 方式 $\mathcal{N}$ に対する攻撃者 $\mathcal{A}$ の利得を

$$\begin{aligned}&\mathrm{Adv}^{\text{Φ-rka-m-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)\\ &\stackrel{\text{def}}{=} |2 \cdot \Pr[b' = b] - 1|\end{aligned}$$

と定義する．ここで $q_H, q_C, q_E, q_{HR}, q_{CR}, q_T$ はそれぞれ，Reg.Hon，Reg.Cor，Extract，Hon.Rev，Cor.Rev，および Test オラクルへのクエリ回数である．もし

$$\mathrm{Adv}^{\text{Φ-rka-m-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)$$

が任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して無視可能であるならば，NIKE 方式 $\mathcal{N}$ が関数クラス Φ に関する RKA-m-CKS-heavy 安全（Φ-RKA-m-CKS-heavy と略す）であるという．

表 3.1: NIKE の各 RKA 安全性において攻撃者に許されたクエリの回数

	Reg.Hon.	Reg.Cor.	Extract	Hon.Rev	Cor.Rev	Test
RKA-CKS-light	2	$*$	0	0	$*$	1
RKA-CKS	$*$	$*$	0	0	$*$	$*$
RKA-CKS-heavy	$*$	$*$	$*$	$*$	$*$	1
RKA-m-CKS-heavy	$*$	$*$	$*$	$*$	$*$	$*$

注意：表中の “$*$” は攻撃者が多項式サイズの任意回のクエリが許されることを表す．また，数字は攻撃者に許された具体的なクエリ回数を表す．

3.2.2 Φ-RKA-CKS-heavy，Φ-RKA-CKS，および Φ-RKA-CKS-light

前の章で最も強い RKA 安全性のエクスペリメントを定義したが，それより弱い他の 3 つの安全性エクスペリメントを本章では定義する．表 3.1 からわかるように，それぞれの RKA 安全性は，攻撃者が許されたクエリの回数によって定義される．

RKA-CKS-heavy 安全性モデルの攻撃者は，Test クエリはただ 1 回だけ許され，他のオラクルクエリは任意回許されている．この安全性モデルでの NIKE 方式 $\mathcal{N}$ に対する攻撃者 $\mathcal{A}$ の利得を

$$\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR})$$

と書く．任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して $\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR})$ が無視可能であれば，NIKE 方式は関数クラス Φ に関する RKA-CKS-heavy（Φ-RKA-CKS-heavy と略す）安全であるという．Test オラクルクエリの回数は 1 に限定されているため，利得のパラメータからは省く．

RKA-CKS 安全性モデルの攻撃者は，任意の回数の Reg.Hon クエリ，Reg.Cor クエリ，Cor.Rev クエリ，および Test クエリが許される．しかし，RKA-m-CKS-heavy や RKA-CKS-heavy 安全性モデルと異なり，攻撃者は Extract クエリや Hon.Rev クエリをすることは許されていない．この安全性モデルでの NIKE 方式 $\mathcal{N}$ に対する攻撃者 $\mathcal{A}$ の利得を

$$\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_{CR}, q_T)$$

と書く．任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して $\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_{CR}, q_T)$ が無視可能であれば，NIKE 方式は関数クラス Φ に関する RKA-CKS（Φ-RKA-CKS と略す）安全であるという．簡単のため，どちらも 0 である Extract クエリおよび Hon.Rev クエリの回数を利得のパラメータから省く．

RKA-CKS-light 安全性のモデルの攻撃者は，2 回の Reg.Hon クエリ，任意回の Reg.Cor クエリおよび Cor.Rev クエリ，1 回の Test クエリが許されている．ただし，Extract クエリおよび Hon.Rev は許されていない．この安全性モデルでの NIKE 方式 $\mathcal{N}$ に対する攻撃者 $\mathcal{A}$ の利得を

$$\mathrm{Adv}^{\Phi\text{-rka-cks-light}}_{\mathcal{A},\mathcal{N}}(\lambda, q_C, q_{CR})$$

と書く．任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して $\mathrm{Adv}^{\Phi\text{-rka-cks-light}}_{\mathcal{A},\mathcal{N}}(\lambda, q_C, q_{CR})$ が無視可能であれば，NIKE 方式は関数クラス Φ に関する RKA-CKS-light（Φ-RKA-CKS-light と略す）安全であるという．簡単のため，定数である Reg.Hon クエリおよび Test クエリの回数は利得のパラメータから省く．

3.2.3 RKA 安全性および通常の安全性の関係

2.4 章で述べたように，RKD 関数のクラス Φ は恒等写像 id を含むことを本論文では要求している．もし Φ が恒等写像だけからなっていれば，本論文で定めた RKA 安全性定義は，Freire ら [23] の通常の安全性定義と等価である．

3.3 非対話型鍵交換の安全性定義間の関係

NIKE の通常の 4 つの安全性定義（CKS-light, CKS, CKS-heavy, m-CKS-heavy）は等価であることが証明されている [23]．本章では，図 3.3 に要約されるように，NIKE の RKA 安全性定義間の関係について示す．まず，定理 1 において，任意の RKD 関数のクラス Φ に関して，RKA-CKS-heavy は RKA-m-CKS-heavy を含意することを示す．逆方向の含意関係は明らかなため，これはつまり，Φ-RKA-CKS-heavy および Φ-RKA-m-CKS-heavy は等価であることを意味する．次に，定理 2 において，任意の RKD 関数のクラス Φ に関して，RKA-CKS および RKA-CKS-heavy は等価で

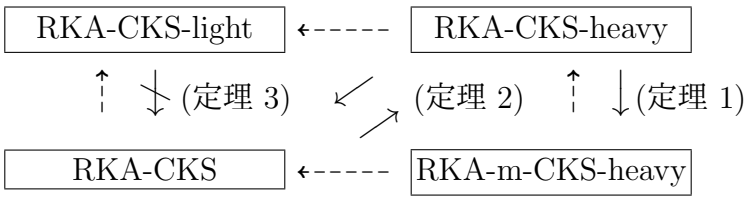

図 3.3: NIKE の RKA 安全性定義間の関係.
点線での矢印は自明な含意関係である．直線での矢印は本論文で示す含意関係を表し，否定の矢印は本論文で含意関係を満たさない方式の存在を示すことを表す.

あることを示す．最後に，定理 3 において，ある RKD 関数のクラスに関して，RKA-CKS-light は必ずしも RKA-CKS を含意しないことを示す．特に，Φ^{lin}-RKA-CKS secure 安全ではないが，Φ^{lin}-RKA-CKS-light 安全な方式を構築する．

安全性定義間の含意関係（参照：定理 1 および定理 2）は任意の RKD 関数のクラスに対して証明されるが，安全性定義間の分離（参照：定理 3）は線形関数のクラスに関してのみ証明されることを明記したい．安全性定義間の分離に関してより詳しい説明は定理 3 の証明の後ですることにする．

3.3.1 Φ-RKA-CKS-heavy ⇔ Φ-RKA-m-CKS-heavy

まず初めに，Φ-RKA-CKS-heavy 安全性は Φ-RKA-m-CKS-heavy 安全性と等価であることを証明する．これら安全性定義間の違いは，攻撃者に許された Test クエリの回数だけである．NIKE 方式に対する Φ-RKA-CKS-heavy （このモデルでは攻撃者は Test クエリは一度だけできる）の意味での攻撃者 $\mathcal{B}$ は，NIKE 方式に対する Φ-RKA-m-CKS-heavy （このモデルでは攻撃者は複数の Test を許される）の意味での攻撃者 $\mathcal{A}$ のオラクルをシミュレート出来ることを示す必要がある．ハイブリッドアーギュメント（hybrid argument，証明の一手法を指す）を用いるとこの証明はシンプルである．簡単に証明の概要を書く：同じ確率空間上で定義されるゲーム列 $G_0, G_1, \ldots, G_{q_T}$ を考える．G_0 は $b = 1$ のときの実際の Φ-RKA-m-CKS-heavy 安全性エクスペリメントであり，G_{q_T} は $b = 0$ のときの Φ-RKA-m-CKS-heavy 安全性エクスペリメントである．さらに，G_j および G_{j+1} $(0 \leq j < q_T)$ は後述するように Test クエリへの 1 つの返答が異なるだけで，その他は全く同じゲームである．任意

のゲーム G_j $(0 \leq j \leq q_T)$ において，$\mathcal{B}$ は，自身の Test オラクルおよび Hon.Rev オラクルを用いることで，$\mathcal{A}$ の Test オラクルをシミュレートできる．

定理 1.（Φ-RKA-CKS-heavy ⇔ Φ-RKA-m-CKS-heavy）任意の RKD 関数クラス Φ に対して，NIKE 方式 $\mathcal{N}$ は，Φ-RKA-m-CKS-heavy 安全のとき，またそのときに限り Φ-RKA-CKS-heavy 安全である．

証明. (⇐) 安全性定義より，NIKE 方式が Φ-RKA-m-CKS-heavy 安全ならば，その方式は Φ-RKA-CKS-heavy 安全である．
(⇒) NIKE 方式 $\mathcal{N}$ が Φ-RKA-CKS-heavy 安全ならば，その方式は Φ-RKA-m-CKS-heavy 安全であることを示す．より正確には，$\mathcal{N}$ に対する Φ-RKA-m-CKS-heavy 安全性の意味での任意の攻撃者 $\mathcal{A}$ に対して，Φ-RKA-CKS-heavy 安全性の意味で $\mathcal{N}$ を攻撃する攻撃者 $\mathcal{B}$ が存在して，以下の式を満たすことを示す：$q'_{HR} \leq q_{HR} + q_T$ に対して，

$$\begin{aligned} &\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{B},\mathcal{N}}(\lambda, q_H, q_C, q_E, q'_{HR}, q_{CR}) \\ &\geq \mathrm{Adv}^{\Phi\text{-rka-m-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)/q_T. \end{aligned} \tag{3.1}$$

$\mathcal{A}$ を走らせるアルゴリズム $\mathcal{B}$ をどのように構成するか示し，$\mathcal{A}$ の利得と $\mathcal{B}$ の利得の間の上記の不等式（式 (3.1)）を示す．

$\mathcal{A}$ に対するゲーム列 $G_0, \ldots, G_{q_T}$ を考える．ゲーム G_i において，Test クエリに答えるために，以下のベクトルが用いられる：

$$H^i = (K^{(1)}_{(\cdot,\cdot,\cdot)}, \ldots, K^{(i)}_{(\cdot,\cdot,\cdot)}, R^{(i+1)}, \ldots, R^{(q_T)}).$$

ここで，初めの i 個の要素は ShK の出力であり，後の $q_T - i$ 個の要素は乱数である．より正確には，$1 \leq j \leq i$ に対して $K^{(j)}_{(\cdot,\cdot,\cdot)}$ は $\mathcal{A}$ の j 番目の Test クエリへの答を表わし，2 つのアイデンティティおよび RKD 関数，つまり $(\mathrm{ID}, \mathrm{ID}', \phi)$ に関する ShK の出力である．ゲーム G_i および G_{i+1} は，$(i+1)$ 番目の Test クエリへの返答が異なるだけであり，これは攻撃者の視点から識別不可能であることに注意が必要である．また，最初のゲーム G_0 は $\mathcal{A}$ の $b = 1$ のときの RKA-m-CKS-heavy 安全性エクスペリメントと等価であり，最後のゲーム G_{q_T} は $b = 0$ の時の $\mathcal{A}$ の RKA-m-CKS-heavy 安全性エクスペリメントと等価である．

以下では，Φ-RKA-CKS-heavy の意味での攻撃者 $\mathcal{B}$ がどのように Φ-RKA-m-CKS-heavy の意味での攻撃者 $\mathcal{A}$ のオラクルをシミュレートするか説明する．

$\mathcal{B}$ の RKA-CKS-heavy エクスペリメントは，入力としてセキュリティパラメータ 1^λ をとり，$\mathsf{CS}(1^\lambda)$ を走らせ，公開パラメータの集合 $params$ を得て，それを $\mathcal{B}$ に与える．$b \in \{0,1\}$ が選ばれた後に，$\mathcal{B}$ が b' を出力するまで，$\mathcal{B}$ のオラクルは $\mathcal{B}$ からのクエリに答える．

まず，$\mathcal{B}$ はランダムに $i^* \in \{0,\ldots,q_T-1\}$ を選び，それから $\mathcal{B}$ は $params$ を $\mathcal{A}$ に与え，$\mathcal{A}$ のオラクルをシミュレートし $\mathcal{A}$ からのクエリに答える．$\mathcal{B}$ は，$\mathcal{A}$ から Reg.Hon，Reg.Cor，Extract，Hon.Rev，および Cor.Rev オラクルクエリを受けると，それぞれ対応する自身のオラクルにクエリし，その答をそのまま $\mathcal{A}$ に返す．$\mathcal{A}$ の Test オラクルクエリに対しては，$\mathcal{B}$ はベクトル

$$\bar{H} = (K^{(1)}_{(\cdot,\cdot,\cdot)},\ldots,K^{(i^*)}_{(\cdot,\cdot,\cdot)},\alpha,R^{(i^*+2)},\ldots,R^{(q_T)})$$

に従って答える．ここで，α は $\mathcal{B}$ の Test クエリの答えであり，$K^{(j)}_{(\cdot,\cdot,\cdot)}(1 \le j \le i^*)$ は $\mathcal{B}$ の Hon.Rev クエリの答である．初めの i^* 個の値 $K^{(j)}_{(\cdot,\cdot,\cdot)}(1 \le j \le i^*)$ を得るために，$\mathcal{B}$ は $\mathcal{A}$ からの j 番目の Test クエリを $\mathcal{B}$ 自身の Hon.Rev オラクルに渡し，その答を $\mathcal{A}$ に返す．

最後に，$\mathcal{B}$ は $\mathcal{A}$ の出力をそのまま出力し，停止する．$\mathcal{B}$ の動作手順は以上である．結果的として，$\mathcal{B}$ は自身の Hon.Rev オラクルにクエリするのは，$\mathcal{A}$ からの全ての Hon.Rev クエリに答える（q_H 回）ためと，$\mathcal{A}$ からのいくつかの Test クエリに答える（最大 q_T 回）ためである．$\mathcal{B}$ の Hon.Rev クエリの回数を q'_{HR} と書き，これは最大 q_H+q_T である．

$\mathcal{A}$ がプレイするゲームは，$\mathcal{B}$ の Test クエリの答 α が ShK の出力なのか，乱数なのかによって，以下のようになる：

α: **ShK の出力** $\mathcal{A}$ はゲーム G_{i^*+1} をプレイしている

α: **乱数** $\mathcal{A}$ はゲーム G_{i^*} をプレイしている

では，G'_0 および G'_1 をそれぞれ，RKA-CKS-heavy エクスペリメントの $b=0$ のとき（Test は ShK の出力を返す）および $b=1$ のとき (Test は乱数を返す）に $\mathcal{B}$ がプレイするゲームとする．つまり，$\mathcal{B}$ がゲーム G'_0（resp. G'_1）をプレイするとき，上

記の α は ShK の出力（resp. 乱数）である．ゆえに，$\mathcal{B}$ がエクスペリメントの初めに i^* を選び，ゲーム G'_0（resp. G'_1）をプレイしているならば，$\mathcal{A}$ はゲーム G_{i^*+1}（resp. G_{i^*}）をプレイしていることになる．

さて，$k \in \{0, \ldots, q_T\}$ に対して，$\mathcal{A}(G_k)$ を，ゲーム G_k において $\mathcal{A}$ が最終的に出力する値を表すとする．さらに，$k \in \{0, 1\}$ に対して，$\mathcal{B}(G'_k)$ を $\mathcal{B}$ が最終的にゲーム G'_k で出力する値を表すとする．$\mathcal{B}$ は i^* を $\{0, \ldots, q_T - 1\}$ よりランダムに選ぶため，$\mathcal{B}$ がゲーム G'_0 および G'_1 で 1 を出力する確率はそれぞれ，

$$\Pr[\mathcal{B}(G'_0) = 1] = \frac{1}{q_T} \sum_{i=0}^{q_T-1} \Pr[\mathcal{A}(G_{i+1}) = 1]$$

および

$$\Pr[\mathcal{B}(G'_1) = 1] = \frac{1}{q_T} \sum_{i=0}^{q_T-1} \Pr[\mathcal{A}(G_i) = 1]$$

である．それゆえに，

$$\begin{aligned}
&\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{B},\mathcal{N}}(\lambda, q_H, q_C, q_E, q'_{HR}, q_{CR}) \\
&= |\Pr[\mathcal{B}(G'_0) = 1] - \Pr[\mathcal{B}(G'_1) = 1]| \\
&= \left| \frac{1}{q_T} \sum_{i=0}^{q_T-1} \Pr[\mathcal{A}(G_{i+1}) = 1] - \frac{1}{q_T} \sum_{i=0}^{q_T-1} \Pr[\mathcal{A}(G_i) = 1] \right| \\
&= \frac{1}{q_T} |\Pr[\mathcal{A}(G_{q_T}) = 1] - \Pr[\mathcal{A}(G_0) = 1]| \\
&= \mathrm{Adv}^{\Phi\text{-rka-m-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)/q_T
\end{aligned}$$

である．これで定理 1 の証明がなされた． □

3.3.2 Φ-RKA-CKS ⇔ Φ-RKA-CKS-heavy

次に，Φ-RKA-CKS 安全性が Φ-RKA-CKS-heavy 安全性と等価であることを証明する．これら安全性定義間の違いは，攻撃者が許されたオラクルにある：Φ-RKA-CKS の意味での攻撃者は Extract クエリ，Hon.Rev クエリをともに許されず，多項式回の Test クエリを許される一方で，Φ-RKA-CKS-heavy の意味での攻撃者は多項式回の Extract クエリおよび Hon.Rev クエリ，1 回だけの Test クエリが許される．

NIKE 方式に対する Φ-RKA-CKS の意味での攻撃者 $\mathcal{B}$ が，NIKE 方式に対する Φ-RKA-CKS-heavy の意味での攻撃者 $\mathcal{A}$ のオラクルをシミュレート出来ることを示す必要がある．

証明に移る前に，その概要を簡単に説明する．識別不可能なゲーム列 H_0, H_1, H_2 を以下のように考える：H_0 は部分的に実際の $b = 0$ の時の Φ-RKA-CKS-heavy 安全性エクスペリメントに関連している．つまり，Test オラクルからの返答は ShK の出力である．H_1 は，1 回なされる Test クエリおよび，一部の Test クエリに使われたのと同じアイデンティティを用いた Hon.Rev クエリには，乱数が返される．H_2 では，Hon.Rev に対する乱数の返答を ShK の出力と入れ替える．なお，Test オラクルから返答は全て乱数である．このゲームは部分的に $b = 1$ の時の Φ-RKA-CKS-heavy 安全性エクスペリメントに関連している．どのゲーム H_j $(0 \leq j \leq 2)$ においても，$\mathcal{B}$ は自身の Test オラクルにクエリすることで，$\mathcal{A}$ の複数の Hon.Rev クエリおよび 1 つの Test クエリに答えることができる．$\mathcal{B}$ は Extract オラクルへのアクセスを持たないが，$\mathcal{A}$ からの Reg.Hon クエリを $\mathcal{B}$ の corrupt ユーザとして登録するという方法により，$\mathcal{A}$ の Extract オラクルをうまくシミュレートできる．

定理 2.（Φ-RKA-CKS $\Leftrightarrow$ Φ-RKA-CKS-heavy）任意の RKD 関数クラス Φ に対して，NIKE 方式 $\mathcal{N}$ は，Φ-RKA-CKS-heavy 安全であるとき，またそのときに限り Φ-RKA-CKS 安全である．

証明. ($\Leftarrow$) 安全性定義および定理 1 より，NIKE 方式が Φ-RKA-CKS-heavy 安全であるならば，その方式は Φ-RKA-CKS 安全であることは明らかである．
（$\Rightarrow$）NIKE 方式 $\mathcal{N}$ が Φ-RKA-CKS 安全であるならば，その方式は Φ-RKA-CKS-heavy 安全であることを示す．より正確には，q_H 回の Reg.Hon クエリ，q_C 回の Reg.Cor クエリ，q_E 回の Extract クエリ，q_{HR} 回の Hon.Rev クエリ，q_{CR} 回の Cor.Rev クエリ，および 1 回の Test クエリをする，$\mathcal{N}$ に対する Φ-RKA-CKS-heavy 安全性の意味での任意の攻撃者 $\mathcal{A}$ に対して，$\mathcal{N}$ に対する Φ-RKA-CKS 安全性の意味での攻撃者 $\mathcal{B}$ および $\mathcal{B}'$ が存在して，以下の式を満たす：$q'_H = 2, q'_C = q_C + q_H - 2, q'_{CR} \leq q_{HR} + q_{CR}$,

および $q'_T \leq q_{HR}+1$ に対して，

$$\begin{aligned}
&\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B},\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T) \\
&\quad + \mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B}',\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T - 1) \\
&\geq \frac{2}{q_H^2}\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR}).
\end{aligned}$$

ここで，q'_H, q'_C，および q'_{CR} はそれぞれ，$\mathcal{B}$ および $\mathcal{B}'$ に許された Reg.Hon クエリ，Reg.Cor クエリ，および Cor.Rev クエリの回数である．また，q'_T（resp. $q'_T - 1$）は $\mathcal{B}$（resp. $\mathcal{B}'$）に許された Test クエリの回数を表す．

まず，この不等式を証明するために，以下のゲームを考える．

G_0 (resp. G_1) $b = 0$（resp. $b = 1$）のときの Φ-RKA-CKS-heavy エクスペリメントにおける，$\mathcal{N}$ に対する $\mathcal{A}$ の通常の攻撃ゲームである．Test クエリは ShK の出力（resp. 乱数）が返される．

G_0^* （resp. G_1^*）このゲームは，エクスペリメントが 2 つのインデックス i^* および j^* を $\{1, \ldots, q_H\}$ からランダムに選び，もし $\mathcal{A}$ が Test クエリで ID_{i^*} および ID_{j^*} を使わなければ，ゲームは $\mathcal{A}$ の最終出力 b' を $b' = 0$ に置き換える，という点を除いて G_0（resp. G_1）と変わらない．ここで，ID_{i^*} および ID_{j^*} はそれぞれ，i^* 番目および j^* 番目の Reg.Hon クエリとして使われたものである．

$\mathcal{A}(G_0)$ (resp. $\mathcal{A}(G_1)$，$\mathcal{A}(G_0^*)$，および $\mathcal{A}(G_1^*)$）は，ゲーム G_0（resp. G_1，G_0^*，および G_1^*）における $\mathcal{A}$ の出力を表すとする．

ゲーム G_0^* および G_1^* における i^* および j^* の選択はランダムであるため，$\mathcal{A}$ が Test クエリに用いる Reg.Hon クエリのインデックスと確率 $1/\binom{q_H}{2} = 2/q_H(q_H-1)$ で一致する．もし，選択が誤っていれば，$\mathcal{A}$ の出力は 0 に置き換わる．結果として，

$$\Pr[\mathcal{A}(G_0^*) = 1] = \frac{2}{q_H(q_H-1)}\Pr[\mathcal{A}(G_0) = 1],$$

および

$$\Pr[\mathcal{A}(G_1^*) = 1] = \frac{2}{q_H(q_H-1)}\Pr[\mathcal{A}(G_1) = 1]$$

を得る．それゆえに，

$$\begin{aligned}
&|\Pr[\mathcal{A}(G_0^*)=1]-\Pr[\mathcal{A}(G_1^*)=1]| \\
&=\frac{2}{q_H(q_H-1)}|\Pr[\mathcal{A}(G_0)=1]-\Pr[\mathcal{A}(G_1)=1]| \\
&=\frac{2}{q_H(q_H-1)} \\
&\quad\times \mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda,q_H,q_C,q_E,q_{HR},q_{CR})
\end{aligned} \tag{3.2}$$

を得る．

続いて，G_0^* および G_1^* に基づいて，以下のゲームを考える．

H_0 このゲームは G_0^* と同じである．

H_1 このゲームは H_0 と以下の点以外変わらない．相違点は，$\mathcal{A}$ の Hon.Rev クエリ $(\mathrm{ID}_A,\mathrm{ID}_B,\phi)$ に対して，$\{\mathrm{ID}_A,\mathrm{ID}_B\}=\{\mathrm{ID}_{i^*},\mathrm{ID}_{j^*}\}$ ならば，乱数が Hon.Rev オラクルの返答として返されること．さらに，$\mathcal{A}$ の Test クエリも乱数が返されるということである．なお，H_0 と同様に，$\mathcal{A}$ が Test クエリで ID_{i^*} および ID_{j^*} を使わなければ，ゲームは $\mathcal{A}$ の出力を 0 に置き換えることに注意が必要である．

H_2 このゲームは G_1^* と同じである．

$\mathcal{A}(H_0)$（resp. $\mathcal{A}(H_1)$ および $\mathcal{A}(H_2)$）は，ゲーム H_0 (resp. H_1 および H_2) における $\mathcal{A}$ の出力を表す．ゲームの定義と三角不等式より，

$$\begin{aligned}
&|\Pr[\mathcal{A}(G_0^*)=1]-\Pr[\mathcal{A}(G_1^*)=1]| \\
&=|\Pr[\mathcal{A}(H_0)=1]-\Pr[\mathcal{A}(H_2)=1]| \\
&\le|\Pr[\mathcal{A}(H_0)=1]-\Pr[\mathcal{A}(H_1)=1]| \\
&\quad+|\Pr[\mathcal{A}(H_1)=1]-\Pr[\mathcal{A}(H_2)=1]|
\end{aligned} \tag{3.3}$$

を得る．

証明を完成するために，以下の補題 2 および補題 3 を証明する．

補題 2. Φ-RKA-CKS-heavy 安全性の意味で $\mathcal{N}$ を攻撃する任意のアルゴリズム $\mathcal{A}$ に対して，q'_H 回の Reg.Hon クエリ，q'_C 回の Reg.Cor クエリ，q'_{CR} 回の Cor.Rev クエ

リ，および q'_T 回の Test クエリを行う Φ-RKA-CKS 安全性の意味で $\mathcal{N}$ を攻撃する攻撃者が存在し，利得は以下を満たす：

$$\begin{aligned}&\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B},\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T)\\ &\geq |\Pr[\mathcal{A}(H_0)=1] - \Pr[\mathcal{A}(H_1)=1]|.\end{aligned} \tag{3.4}$$

証明. (補題 2 の証明) 内部で攻撃者 $\mathcal{A}$ を利用し，NIKE 方式 $\mathcal{N}$ を，式. (3.4) に示した利得で，RKA-CKS の意味で破る攻撃者 $\mathcal{B}$ を示す.

$\mathcal{B}$ は入力として公開パラメータの集合 $params$ をとり，以下のように動作する：

- $\mathcal{B}$ は i, j（$1 \leq i < j \leq q_H$）をランダムに選ぶ.
- $\mathcal{B}$ は $params$ を $\mathcal{A}$ に与える.
- $\mathcal{A}$ からの i 番目および j 番目の Reg.Hon クエリに対して，$\mathcal{B}$ はそれらを自身の Reg.Hon オラクルにクエリし，その返答 pk_i および pk_j それぞれを $\mathcal{A}$ に返す．$\mathcal{A}$ からの残りの Reg.Hon クエリ ID_k $(k \neq i, j)$ に対して，$\mathcal{B}$ は $(pk_k, sk_k) \xleftarrow{\$} \mathsf{KG}(params, \mathrm{ID}_k)$ を生成し，(ID_k, pk_k) を自身の Reg.Cor オラクルにクエリする．それから，$\mathcal{B}$ はその返答 pk_k を $\mathcal{A}$ に返す.
- $\mathcal{A}$ からの Reg.Cor クエリに対して，$\mathcal{B}$ はどれらを自身の Reg.Cor オラクルにクエリし，その返答を $\mathcal{A}$ に返す.
- $\mathcal{A}$ からの Extract クエリ ID_ℓ に対して，
  - ID_ℓ が $\mathcal{A}$ によって honest として登録されており，$\mathrm{ID}_\ell \notin \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は自身が生成した sk_ℓ を返す.
  - そうでなければ，$\mathcal{B}$ は 0 を出力し，停止する.
- $\mathcal{A}$ からの Hon.Rev クエリ $(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ に対して，
  - $\{\mathrm{ID}_A, \mathrm{ID}_B\} = \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ を自身の Test オラクルにクエリし，その返答を $\mathcal{A}$ に返す.
  - $\mathrm{ID}_A \notin \{\mathrm{ID}_i, \mathrm{ID}_j\}$ かつ $\mathrm{ID}_B \in \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ を自身の Cor.Rev オラクルにクエリし，その返答を $\mathcal{A}$ に返す.

– そうでなければ (i.e. $\mathrm{ID}_B \notin \{\mathrm{ID}_i, \mathrm{ID}_j\}$)，$\mathcal{B}$ は $K_{A,B,\phi} = \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \phi(sk_B))$ を計算し，それを $\mathcal{A}$ に返す．[2]

- $\mathcal{A}$ からの Cor.Rev クエリ $(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ に対して，
  - $\mathrm{ID}_B \in \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ を自身の Cor.Rev オラクルにクエリし，その返答を $\mathcal{A}$ に返す．
  - そうでなければ (i.e. $\mathrm{ID}_B \notin \{\mathrm{ID}_i, \mathrm{ID}_j\}$)，$\mathcal{B}$ は $K_{A,B,\phi} = \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \phi(sk_B))$ を計算し，それを $\mathcal{A}$ に返す．[2]
- $\mathcal{A}$ からの $Test$ クエリ $(\mathrm{ID}_1^*, \mathrm{ID}_2^*, \phi^*)$ に対して，
  - $\{\mathrm{ID}_1^*, \mathrm{ID}_2^*\} = \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $(\mathrm{ID}_1^*, \mathrm{ID}_2^*, \phi^*)$ を自身の Test オラクルにクエリし，その返答を $\mathcal{A}$ に返す．
  - そうでなければ，$\mathcal{B}$ は 0 を出力し，停止する．

$\mathcal{A}$ がビット b' を出力し停止するとき，$\mathcal{B}$ はこのビットを出力し，同様に停止する．

$\mathcal{B}$ の動作手順は以上である．結果的に，$\mathcal{B}$ の推測した i および j に対応する，$\mathcal{A}$ からの Reg.Hon クエリ ID_i および ID_j に答えるために，$\mathcal{B}$ は自身の Reg.Hon オラクルに 2 回クエリをする．よって $q'_H = 2$ である．また，$\mathcal{A}$ からの Reg.Hon クエリ（$q_H - 2$ 回）および $\mathcal{A}$ からの全ての Reg.Cor クエリに答えるために，$\mathcal{B}$ は自身の Reg.Cor オラクルにクエリする．よって $q'_C = q_C + q_H - 2$ である．$\mathcal{A}$ からの Hon.Rev クエリのいくつか（最大 q_{HR} 回）および $\mathcal{A}$ からの全ての Cor.Rev クエリに答えるために $\mathcal{B}$ か自身の Cor.Rev オラクルにクエリする．よって $q'_{CR} \leq q_{HR} + q_{CR}$ である．$\mathcal{A}$ からの Hon.Rev クエリのいくつか（最大 q_{HR} 回）および $\mathcal{A}$ からの 1 回の Test クエリに答えるために，$\mathcal{B}$ は自身の Test オラクルにクエリする．よって $q'_T \leq q_{HR} + 1$ である．Φ-RKA-CKS-heavy エクスペリメントのルールにより，$\mathcal{A}$ はちょうど 1 回の Test クエリをすることに注意．

$\mathcal{B}$ は $\mathcal{A}$ のオラクルを完全にシミュレートする．$\mathcal{B}$ のチャレンジビットが $b = 0$ ならば，$\mathcal{A}$ の Hon.Rev クエリおよび $\mathcal{A}$ の Test クエリは，H_0 のように ShK の出力で返答される．一方で，$\mathcal{B}$ のチャレンジビットが $b = 1$ ならば，$\mathcal{A}$ の $(\mathrm{ID}_i, \mathrm{ID}_j, \phi)$ および $(\mathrm{ID}_j, \mathrm{ID}_i, \phi)$ の形をした Hon.Rev クエリと，$\mathcal{A}$ の Test クエリは，H_1 のように，

[2] $\mathcal{B}$ はこのような ID_B に対応する sk_B を持っており，そのため $\mathcal{B}$ はこのステップを実行できる．

乱数で返答される．それゆえに，$\mathcal{B}$ は $\mathcal{A}$ を使うことで，自身の Test オラクルから返ってきた値が，ランダムな鍵なのか ShK アルゴリズムの出力なのか区別ができる．$\mathcal{A}$ が任意の ϕ に対して，$(\mathrm{ID}_i, \mathrm{ID}_j, \phi)$ を自身の Test オラクルにクエリする確率は，$1/\binom{q_H}{2} = 2/q_H(q_H-1)$ である．この理由は，$\mathcal{B}$ による i および j の選択が $\mathcal{A}$ の動作および $\mathcal{B}$ のチャレンジビットとは独立だからである．

G'_0（resp. G'_1）を $\mathcal{B}$ の $b=0$（resp. $b=1$）での Φ-RKA-CKS エクスペリメントとする．また，$\mathcal{B}(G'_0)$（resp. $\mathcal{B}(G'_1)$）は，ゲーム G'_0（resp. G'_1）における $\mathcal{B}$ の出力を表すとする．このとき，

$$\Pr[\mathcal{B}(G'_0)=1] = \Pr[\mathcal{A}(H_0)=1],$$

および

$$\Pr[\mathcal{B}(G'_1)=1] = \Pr[\mathcal{A}(H_1)=1]$$

を得る．

よって，

$$\begin{aligned}
&\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B},\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T) \\
&= |\Pr[\mathcal{B}(G'_0)=1] - \Pr[\mathcal{B}(G'_1)=1]| \\
&= |\Pr[\mathcal{A}(H_0)=1] - \Pr[\mathcal{A}(H_1)=1]|
\end{aligned}$$

である．そして以上で補題 2 が証明された． □

補題 3. $\mathcal{N}$ を Φ-RKA-CKS-heavy の意味で攻撃する任意のアルゴリズム $\mathcal{A}$ に対して，q'_H 回の Reg.Hon クエリ，q'_C 回の Reg.Cor クエリ，q'_{CR} 回の Cor.Rev クエリ，および q'_T-1 回の Test クエリをする $\mathcal{N}$ を Φ-RKA-CKS の意味で攻撃するアルゴリズム $\mathcal{B}'$ が存在して利得は以下を満たす：

$$\begin{aligned}
&\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B}',\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T-1) \\
&\geq |\Pr[\mathcal{A}(H_2)=1] - \Pr[\mathcal{A}(H_1)=1]|.
\end{aligned} \tag{3.5}$$

証明.（補題 3 の証明）証明は，$\mathcal{B}'$ のやる Test オラクルのシミュレーションをのぞいて，補題 2 の証明と同じである．q'_T-1 回の Test クエリをし，内部で攻撃者 $\mathcal{A}$ を利用しながら NIKE 方式 $\mathcal{N}$ を Φ-RKA-CKS の意味で攻撃する，利得が式 (3.5) に表

されるような攻撃者 $\mathcal{B}'$ を示す．ここでは，$\mathcal{B}'$ の $\mathcal{A}$ からの Test クエリへの返答のみをハイライトする．

$\mathcal{A}$ からの Test クエリに対して，$\mathcal{B}'$ はランダムな鍵を返す．

$\mathcal{A}$ からの $q_{HR}(= q'_T - 1)$ 回の Hon.Rev クエリに答えるためだけに，$\mathcal{B}'$ は自身の Test オラクルにクエリする．$\mathcal{B}'$ のチャレンジビットが $b = 0$ であれば，$\mathcal{A}$ の Hon.Rev クエリに対して，$\mathcal{B}'$ は ShK の出力を返す．そして，これは $\mathcal{B}'$ が H_2 をシミュレートしていることを意味する．$b = 1$ ならば，$\mathcal{A}$ の $(\mathrm{ID}_i, \mathrm{ID}_j, \phi)$ および $(\mathrm{ID}_j, \mathrm{ID}_i, \phi)$ という形の Hon.Rev クエリに対して，$\mathcal{B}'$ は乱数を返す．これは $\mathcal{B}'$ が H_1 をシミュレートしていることを意味する．

それゆえに，

$$\begin{aligned}
&\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B}',\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T - 1) \\
&= |\Pr[\mathcal{B}'(G'_0) = 1] - \Pr[\mathcal{B}'(G'_1) = 1]| \\
&= |\Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_1) = 1]|
\end{aligned}$$

である．以上で補題 3 が証明された． □

補題 2，補題 3，式 (3.2)，式 (3.3)，および不等式 $2/q_H(q_H - 1) \geq 2/q_H^2$ より，

$$\begin{aligned}
&\mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B},\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T) \\
&\quad + \mathrm{Adv}^{\Phi\text{-rka-cks}}_{\mathcal{B}',\mathcal{N}}(\lambda, q'_H, q'_C, q'_{CR}, q'_T - 1) \\
&\geq \frac{2}{q_H^2}\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR})
\end{aligned}$$

である．以上で定理 2 が証明された． □

3.3.3 Φ^{lin}-RKA-CKS-light $\not\Longrightarrow$ Φ^{lin}-RKA-CKS

上述の 2 つの定理によって，3 つの安全性定義 RKA-CKS，RKA-CKS-heavy，および RKA-m-CKS-heavy は RKD 関数のクラスに関わらず等価であることを示した．続いて，それら 3 つの安全性定義と RKA-CKS-light 安全性との間の分離の結果について示す．分離は，線形関数のクラスに対して示す．証明のアイディアは，Φ^{lin}-RKA-CKS モデルでは攻撃者が 2 つ以上の ShK の出力を利用できる一方で，Φ^{lin}-RKA-CKS-light モデルでは，ShK の出力を唯一つだけしか使えないという事実を利用することにある．

定理 3. (Φ^{lin}-RKA-CKS-light $\not\Rightarrow$ Φ^{lin}-RKA-CKS) NIKE 方式はセキュリティパラメータ λ を用い，r は自然数とする．Φ^{lin}-RKA-CKS-light 安全な秘密鍵空間が加法群 $\mathbb{Z}_r = \{0, \ldots, r-1\}$ であるような NIKE 方式 $\mathcal{N}$ が存在すれば，Φ^{lin}-RKA-CKS 安全ではないが，Φ^{lin}-RKA-CKS-light 安全であるような NIKE 方式 $\mathcal{N}'$ が存在する．

証明. $\mathcal{N} = (\mathsf{CS}, \mathsf{KG}, \mathsf{ShK})$ を Φ^{lin}-RKA-CKS-light 安全な NIKE 方式で，秘密鍵空間が加法群 $\mathbb{Z}_r$ であるものとする．なおこれは仮定より存在が保証されている．以下では，$\mathcal{N}$ に基づいて $\mathcal{N}'$ の構成を示し，それが Φ^{lin}-RKA-CKS-light 安全ではあるが，Φ^{lin}-RKA-CKS 安全ではないことを示す．

秘密鍵空間が $\mathbb{Z}_{2r} = \{0, \ldots, 2r-1\}$ であるような NIKE 方式 $\mathcal{N}' = (\mathsf{CS}', \mathsf{KG}', \mathsf{ShK}')$ を以下に定義する：

$\mathsf{CS}'(1^\lambda)$

1. $params \xleftarrow{\$} \mathsf{CS}(1^\lambda)$
2. return $params$

$\mathsf{KG}'(params, \mathrm{ID})$

1. $(pk, sk) \xleftarrow{\$} \mathrm{KG}(params, \mathrm{ID})$
2. $(pk', sk') \leftarrow (pk, 2 \cdot sk)$
3. return (pk', sk')

$\mathsf{ShK}'(\mathrm{ID}_1, pk'_1, \mathrm{ID}_2, sk'_2)$

1. If $sk'_2 \bmod 2 = 1$, then $sk'_2 \leftarrow sk'_2 - 1$
2. $(pk_1, sk_2) \leftarrow (pk'_1, sk'_2/2 \bmod r)$
3. $K \leftarrow \mathsf{ShK}(\mathrm{ID}_1, pk_1, \mathrm{ID}_2, sk_2)$
4. return K

ShK' の一つ目のステップにより，Φ^{lin}-RKA-CKS モデルの攻撃者は相互に関係する ShK の出力たちを得ることができる．一方で，Φ^{lin}-RKA-CKS-light モデルの攻撃者にはそれができない．というのも，Φ^{lin}-RKA-CKS-light 安全性モデルでは，攻撃者は一度しか Test クエリを行えないためである．証明を完成するために，次の補題 4 および補題 5 を証明する．

補題 4. $\mathcal{N}'$ は Φ^{lin}-RKA-CKS 安全ではない．

補題 4 の証明. $\mathcal{N}'$ に対する Φ^{lin}-RKA-CKS の意味での攻撃を示す．この安全性モデルでは，攻撃者は $(\mathrm{ID}, \mathrm{ID}', \phi)$ のような Test クエリを許される．ここで，$\alpha \in \mathbb{Z}_{2r}$ に対して，$\phi(k) = k + \alpha \bmod 2r$ である．以下のような攻撃者を考えよう．攻撃者は $(\mathrm{ID}_A, \mathrm{ID}_B, \mathrm{id})$ を Test オラクルにクエリし，共有鍵 K を得てから，今度は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi(k) = k + 1)$ を Test オラクルにクエリし，K' を得る．$K' = K$ ならば，攻撃者は 0 を出力する．こうの攻撃者はチャレンジビット b を非情に高い確率で推測できる．なぜならば，もし $b = 0$ ならば

$$\begin{aligned} K &= \mathsf{ShK}'(\mathrm{ID}_A, pk'_A, \mathrm{ID}_B, sk'_B) \\ &= \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, sk_B) \end{aligned}$$

であり，

$$\begin{aligned} K' &= \mathsf{ShK}'(\mathrm{ID}_A, pk'_A, \mathrm{ID}_B, \phi(sk'_B)) \\ &= \mathsf{ShK}'(\mathrm{ID}_A, pk'_A, \mathrm{ID}_B, sk'_B + 1) \\ &= \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, sk_B) \end{aligned}$$

であるので，$K = K'$ となるためである．一方で，$b = 1$ ならば，K と K' は乱数なので，$K = K'$ となる確率は無視可能である． □

補題 5. $\mathcal{N}'$ は Φ^{lin}-RKA-CKS-light 安全である．

補題 5 の証明. $\mathcal{N}'$ を Φ^{lin}-RKA-CKS-light 安全性の意味で攻撃する，確率的多項式時間攻撃者 $\mathcal{A}$ を考える．$\mathcal{N}$ が Φ^{lin}-RKA-CKS-light 安全であれば，

$$\mathrm{Adv}^{\Phi\text{-rka-cks-light}}_{\mathcal{A},\mathcal{N}'}(\lambda, q_C, q_{CR})$$

が無視可能であることを示す．それを証明するために，$\mathcal{A}$ をサブルーチンとして使いながら $\mathcal{N}$ を Φ^{lin}-RKA-CKS-light 安全性の意味で攻撃する攻撃者 $\mathcal{B}$ で，

$$\begin{aligned} &\mathrm{Adv}^{\Phi^{\mathrm{lin}}\text{-rka-cks-light}}_{\mathcal{B},\mathcal{N}}(\lambda, q_C, q_{CR}) \\ &\geq \mathrm{Adv}^{\Phi^{\mathrm{lin}}\text{-rka-cks-light}}_{\mathcal{A},\mathcal{N}'}(\lambda, q_C, q_{CR}) \end{aligned}$$

となるものを考える．

$\mathcal{B}$ は初めに公開パラメータの集合 $params$ を与えられると，それを $\mathcal{A}$ に与える．$\mathcal{A}$ からの Reg.Hon クエリおよび Reg.Cor クエリに対して，$\mathcal{B}$ はそれらをそれぞれ自身

の Reg.Hon オラクルおよび Reg.Cor オラクルにクエリする．そしてその返答を $\mathcal{A}$ に返す．$(\mathrm{ID}_A, \mathrm{ID}_B, \phi(k) = k + \alpha)$ のような Cor.Rev クエリ（resp. Test クエリ）に対して，$\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \hat{\phi}(k) = k + \beta)$ を自身の Cor.Rev（resp. Test）オラクルにクエリする．ここで，α が偶数ならば $\beta = \alpha/2$ であり，そうでなければ $\beta = (\alpha - 1)/2$ である．NIKE 方式 $\mathcal{N}'$ の正当な秘密鍵は偶数であるため，α の偶奇にのみ気をつければよい．sk_B を $\mathcal{B}$ の視点から見た ID_B の秘密鍵とすし，$sk'_B = 2 \cdot sk_B$ を $\mathcal{A}$ の視点から見た ID_B の秘密鍵とする．もちろん，どちらも $\mathcal{A}$ および $\mathcal{B}$ からは隠されている．上記の構成によって，$\mathcal{A}$ が偶数である α で $(\mathrm{ID}_A, \mathrm{ID}_B, \phi(k) = k + \alpha)$ を Cor.Rev オラクルにクエリする時，$\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \hat{\phi}(k) = k + \beta)$ を自身の Cor.Rev オラクルにクエリし，

$$
\begin{aligned}
K_{A,B,\hat{\phi}}\ &\mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \hat{\phi}(sk_B)) \\
&= \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, sk_B + \beta) \\
&= \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, sk_B + \alpha/2) \\
&= \mathsf{ShK}'(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, 2 \cdot sk_B + \alpha) \\
&= \mathsf{ShK}'(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, sk'_B + \alpha) \\
&= \mathsf{ShK}'(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \phi(sk'_B))
\end{aligned}
$$

を得る．そして $K_{A,B,\hat{\phi}}$ を返すのだが，これは求められていたものである．これは α が奇数の場合も確かめることが出来る．どうようにして，$\mathcal{B}$ は $\mathcal{A}$ の Test クエリに答える．上記のことから，$\mathcal{B}$ が $\mathcal{A}$ に対して Φ^{lin}-RKA-CKS-light エクスペリメントを完全にシミュレートしていることがわかる．最終的に，$\mathcal{B}$ は $\mathcal{A}$ が出力したものを出力する．このことから，

$$
\begin{aligned}
&\mathrm{Adv}^{\Phi^{\mathrm{lin}}\text{-rka-cks-light}}_{\mathcal{B},\mathcal{N}}(\lambda, q_C, q_{CR}) \\
&\geq \mathrm{Adv}^{\Phi^{\mathrm{lin}}\text{-rka-cks-light}}_{\mathcal{A},\mathcal{N}'}(\lambda, q_C, q_{CR})
\end{aligned}
$$

ということが示せ，補題 5 が証明された． □

補題 4 および補題 5 により，定理 3 が証明された． □

定理 3 における分離は，線形関数のクラスに対してのみ証明されたことに注意が必要である．RKD 関数のクラスが恒等写像のみからなるならば，RKA-CKS-light

安全性とその他の 3 つの安全性との間に分離はなく，Freire ら [23] によって示されたように，4 つの安全性定義は等価である．

3.4 RKA-CKS-heavy 安全な非対話型鍵交換

Freire ら [23] は RKA ではない通常の設定で安全だと示された NIKE 方式を提案した．本章では，その方式の一つ $\mathrm{NIKE}_{\mathrm{fac}}$ [23] を記し，その RKA 安全性について議論する．この方式はランダムオラクルとしてモデル化されるハッシュ関数 $H : \{0,1\}^* \to \{0,1\}^\lambda$ を用い，以下の 3 つのアルゴリズムからなる．アイデンティティ空間 $\mathcal{IDS}$ としては，自然な順序を有するものを用いる（i.e., 番号順，辞書順など.）:

$\mathsf{CS}(1^\lambda)$

1. $(n, p, q) \xleftarrow{\$} \mathsf{RSAgen}(1^\lambda)$
2. $g \xleftarrow{\$} \mathbf{QR}_n^+$, where $\langle g \rangle = \mathbf{QR}_n^+$
3. $params \leftarrow (H, n, g)$
4. return $params$

$\mathsf{KG}(params, \mathrm{ID})$

1. $x \xleftarrow{\$} \mathbb{Z}_{\lfloor n/4 \rfloor}$
2. $X \leftarrow g^x$
3. $pk \leftarrow X, sk \leftarrow x$
4. return (pk, sk)

$\mathsf{ShK}(\mathrm{ID}_1, pk_1, \mathrm{ID}_2, sk_2)$

1. If $\mathrm{ID}_1 = \mathrm{ID}_2$ or $pk_1 \notin \mathbf{QR}_n^+$ or $pk_2 \notin \mathbf{QR}_n^+$, then return $\perp$.
2. Else if $\begin{cases} \mathrm{ID}_1 < \mathrm{ID}_2, \text{ return } H(\mathrm{ID}_1, \mathrm{ID}_2, pk_1^{sk_2}) \\ \mathrm{ID}_2 < \mathrm{ID}_1, \text{ return } H(\mathrm{ID}_2, \mathrm{ID}_1, pk_1^{sk_2}) \end{cases}$

$\mathrm{NIKE}_{\mathrm{fac}}$ 方式は，素因数分解仮定のもとでランダムオラクルモデルにおいて（通常の） CKS-light 安全だと証明されている [23]．本章では，この方式が素因数分解仮定のもとでランダムオラクルモデルにおいて Φ^{lin}-RKA-CKS-heavy 安全であることを証明する．なお，Φ^{lin}-RKA-CKS-heavy 安全性は等価な 3 つの安全性定義のう

ち，ここでの証明をするには最も便利な定義である．$\mathbb{Z}$ 上の線形関数を RKD 関数として考える．ここで，$\mathbb{Z}$ は秘密鍵空間 $\mathbb{Z}_{\lfloor n/4 \rfloor}$ を含む．

命題 1 より，以下の定理を証明すれば十分である．

定理 4. $\mathrm{NIKE}_{\mathrm{fac}}$ は RSAgen に対する DSDH 仮定のもと，(H がランダムオラクルとしてモデル化される）ランダムオラクルモデルにおいて，Φ^{lin}-RKA-CKS-heavy 安全である．より正確には，$\mathrm{NIKE}_{\mathrm{fac}}$ を Φ^{lin}-RKA-CKS-heavy 安全性の意味で攻撃する任意の攻撃者 $\mathcal{A}$ に対して，RSAgen に関する DSDH 問題を解くアルゴリズム $\mathcal{B}$ が存在して以下の不等式を満たす：

$$\begin{aligned}&\mathrm{Adv}^{\mathrm{dsdh}}_{\mathcal{B},\mathsf{RSAgen}}(\lambda)\\ &\geq 2\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathrm{NIKE}_{\mathrm{fac}}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR})/q_H^2 - \epsilon.\end{aligned}$$

ここで ϵ は無視可能である．

証明. $\mathrm{NIKE}_{\mathrm{fac}}$ に対する RKA-CKS-heavy 安全性の証明はオリジナルの Freire ら [23] による $\mathrm{NIKE}_{\mathrm{fac}}$ の CKS-light 安全性の証明と同様である．CKS-light および RKA-CKS-heavy の間の特筆すべき違いは，RKA-CKS-heavy は CKS-light より多くのオラクル（$\mathsf{Extract}$ および $\mathsf{Hon.Rev}$）を考慮するということと，RKD 関数を考慮するということである．これから，$\mathrm{NIKE}_{\mathrm{fac}}$ を RKA-CKS-heavy の意味で攻撃する $\mathcal{A}$ を内部で用い，RSAgen に関する DSDH 問題を解く攻撃者 $\mathcal{B}$ を以下のように構築する：$\mathcal{B}$ はリスト L，L_{H}，L_{C}，および L_{T} を準備する．$\mathcal{B}$ は Test クエリで使われる 2 つのアイデンティティを予想する．これを ID_i，ID_j とし，リスト L_{H} に honest として記録する．一方で，他のアイデンティティはリスト L_{C} に corrupt として記録する．こうすることで，$\mathcal{B}$ は $\mathcal{A}$ の $\mathsf{Extract}$ オラクルをシミュレートできる．というのも，$\mathcal{A}$ から $\mathsf{Extract}$ オラクルに聞かれるであろうアイデンティティに対応する秘密鍵は，後で説明するように，$\mathcal{B}$ によって生成されているためである．$\mathcal{A}$ からの $\mathsf{Hon.Rev}$ クエリに対しては，$\mathcal{B}$ は単に乱数を返す．

より正確には，入力 $(n, g, X = g^x, Y = g^y)$ を持つ $\mathcal{B}$ は，オラクル $\mathrm{DDH}_{g,X}(\cdot,\cdot)$ および $\mathrm{DDH}_{g,Y}(\cdot,\cdot)$ にアクセスし，g^{xy} を計算しようとする．$\mathcal{B}$ は以下のように動作する：

- $\mathcal{B}$ はリスト $L = \emptyset$, $L_{\mathrm{H}} = \emptyset$，$L_{\mathrm{C}} = \emptyset$，および $L_{\mathrm{T}} = \emptyset$ を生成する．

- $\mathcal{B}$ は相異なる 2 つのインデックス $i, j \xleftarrow{\$} \{1, \ldots, q_H\}$ を選ぶ.
- $\mathcal{B}$ は $\mathcal{A}$ に公開パラメータの集合 (H, n, g) を与える．ここで，H は $\mathcal{B}$ によって管理されるランダムオラクルである.
- Reg.Hon: $\mathcal{A}$ からのクエリ ID に対して，$(\mathrm{ID}, *) \in L_{\mathrm{C}}$ ならば，$\mathcal{B}$ は $\perp$ を出力し，停止する．そうでなければ,
  - もし，それが i 番目もしくは j 番目のクエリであれば，$\mathcal{B}$ はそれぞれ $pk_i = X$ もしくは $pk_j = Y$ を返す．一般性を失うことなく，$\mathrm{ID}_i < \mathrm{ID}_j$ と仮定する.
  - そうでなければ，$\mathcal{B}$ は KG を走らせ，公開鍵/秘密鍵のペア (pk, sk) を得て，(ID, pk, sk) をリスト L_{H} に記録し，pk を返す.
- Reg.Cor: $\mathcal{A}$ からのクエリ (ID, pk) に対して，$(\mathrm{ID}, *, *) \in L_{\mathrm{H}}$ であれば，$\mathcal{B}$ は $\perp$ を出力した上で停止し，$(\mathrm{ID}, *) \in L_{\mathrm{C}}$ であれば，その L_{C} 内のエントリを消去し，代わりに (ID, pk) を L_{C} に記録する．そうでなければ,
  - $\mathrm{ID} = \mathrm{ID}_i$ または $\mathrm{ID} = \mathrm{ID}_j$ ならば，$\mathcal{B}$ は $\perp$ を返す.
  - そうでなければ，$\mathcal{B}$ は (ID, pk) をリスト L_{C} に記録する.
- Extract: $\mathcal{A}$ からのクエリである honest ユーザアイデンティティ ID に対して,
  - $\mathrm{ID} \in \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $\perp$ を出力し，停止する.
  - そうでなければ，$\mathcal{B}$ は L_{H} 内の対応する秘密鍵を返す.
- Hon.Rev: $\phi(k) = k + \alpha$ であるような $\mathcal{A}$ からのクエリ $(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ に対して,
  - $((\mathrm{ID}_A, \mathrm{ID}_B), \phi, *) \in L_{\mathrm{T}}$ または $(\phi = \mathrm{id}$ かつ $((\mathrm{ID}_B, \mathrm{ID}_A), \mathrm{id}, *) \in L_{\mathrm{T}})$ ならば，$\mathcal{B}$ は $\perp$ を出力し，停止する.
  - そうでなければ，かつ $(\mathrm{ID}_A, \mathrm{ID}_B, \perp, h, K) \in L$ ならば，$\mathcal{B}$ は以下のように動作する：
    - もし h/pk_A^{α} が $(\mathrm{ID}_A, \mathrm{ID}_B)$ に関する DH 問題の答として正しければ, つまり $pk_A^{sk_B}$ ならば $\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi, h, K)$ を L に記録し，K を返す. ここで，pk_A は ID_A の公開鍵で sk_B は ID_B の秘密鍵である．$\mathrm{ID}_B \notin \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば sk_B は $\mathcal{B}$ が生成しているため，DH 問題の正しい答かどうか $\mathcal{B}$ は容易に確認できる．また，たとえ $\mathrm{ID}_B \in \{\mathrm{ID}_i, \mathrm{ID}_j\}$

であり $\mathcal{B}$ が sk_B の値を知らないとしても，DDH オラクルを利用することで確認ができる．

- そうでなければ，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，$(\mathrm{ID}_C, \mathrm{ID}_A, \phi, \perp, K)$ を L に記録し K を返す．

– そうでなければ，$\{\mathrm{ID}_A, \mathrm{ID}_B\} = \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi, *, K)$ がリスト L 内にあるかチェックする:

- もし含まれていれば，$\mathcal{B}$ は $K_{A,B,\phi} = K$ を返す．
- そうでなければ，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，$(\mathrm{ID}_A, \mathrm{ID}_B, \phi, \perp, K)$ を L に記録し，$(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ を L_{HR} に記録し，$K_{A,B,\phi} = K$ を返す．

– そうでなければ，$\{\mathrm{ID}_A, \mathrm{ID}_B\} \neq \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば $\mathcal{B}$ は以下に説明するように $pk_A^{\phi(sk_B)}$ を計算し，また $\mathcal{B}$ が管理するハッシュオラクル H を用いて $H(\mathrm{ID}_A, \mathrm{ID}_B, pk_A^{\phi(sk_B)})$ を計算する．

– $\mathcal{B}$ は $(\mathrm{ID}_A, \mathrm{ID}_B, \phi, pk_A^{\phi(sk_B)}, K_{A,B,\phi})$ をリスト L に記録し，$(\mathrm{ID}_A, \mathrm{ID}_B, \phi)$ をリスト L_{HR} に記録し，$K_{A,B,\phi}$ を返す．

では，値 $pk_A^{\phi(sk_B)}$ と対応する共有鍵を sk_B を知らずにどのように計算するかを説明する：$\mathrm{ID}_A \notin \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は sk_A を知っているはずである．というのも，自身で KG を走らせ sk_A を得ているためである．それゆえに，RKD 関数が線形関数 $\phi_\alpha^{\mathrm{lin}}(k) = k + \alpha$ ならば，pk_A, pk_B および sk_A を知っている $\mathcal{B}$ は $pk_A^{\phi(sk_B)} = pk_A^{sk_B+\alpha} = (pk_B^{sk_A}) \cdot pk_A^{\alpha}$ を計算できる．そのため，$\mathcal{B}$ は ShK の出力も $K_{A,B,\phi} = \mathsf{ShK}(\mathrm{ID}_A, pk_A, \mathrm{ID}_B, \phi(sk_B)) = H(\mathrm{ID}_A, \mathrm{ID}_B, pk_A^{\phi(sk_B)})$ により計算できる．ここで $\mathrm{ID}_A < \mathrm{ID}_B$ と仮定した．

- Cor.Rev: $\phi(k) = k + \alpha$ であるような $\mathcal{A}$ からのクエリ $(\mathrm{ID}_C, \mathrm{ID}_A, \phi)$ に対して，$\mathcal{B}$ は $(\mathrm{ID}_C, \mathrm{ID}_A, \phi, *, K)$ がリスト L に含まれるかチェックし，以下のように動作する：

– もし含まれていれば，$\mathcal{B}$ は $K_{C,A,\phi} = K$ を返す．

– そうでなければ，$(\mathrm{ID}_C, \mathrm{ID}_A, \perp, h, K) \in L$ ならば，$\mathcal{B}$ は以下のように動作する：

- もし h/pk_C^α が $(\mathrm{ID}_C, \mathrm{ID}_A)$ に関する正しい DH 問題の答ならば，$\mathcal{B}$ は $(\mathrm{ID}_C, \mathrm{ID}_A, \phi, h, K)$ を L に記録し K を返す．正しい DH 問題の答か

どうかは，$\mathrm{ID}_A \in \{\mathrm{ID}_i, \mathrm{ID}_j\}$ かどうかによらず上記の Hon.Rev における説明と同様に確認ができる．

- そうでなければ，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，$(\mathrm{ID}_C, \mathrm{ID}_A, \phi, \perp, K)$ を L に記録し，K を返す．

– そうでなければ，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，$(\mathrm{ID}_C, \mathrm{ID}_A, \phi, \perp, K)$ を L に記録し，K を返す．

ここで ID_C は corrupt ユーザアイデンティティであり，ID_A は honest ユーザアイデンティティである．

- Test: $\mathcal{A}$ からのクエリ $(\mathrm{ID}, \mathrm{ID}', \phi^*)$ に対して，
  - $(\mathrm{ID}, \mathrm{ID}', \phi^*) \in L_{\mathrm{HR}}$ または $(\phi^* = \mathrm{id}$ かつ $((\mathrm{ID}', \mathrm{ID}), \mathrm{id}, *) \in L_{\mathrm{HR}})$ ならば，$\mathcal{B}$ は $\perp$ を返し，停止する．
  - そうでなければ，$\{\mathrm{ID}, \mathrm{ID}'\} = \{\mathrm{ID}_i, \mathrm{ID}_j\}$ ならば，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，それを返す．$\mathcal{B}$ は $((\mathrm{ID}, \mathrm{ID}'), \phi^*, K)$ を L_{T} に記録する．
  - そうでなければ，$\mathcal{B}$ は $\perp$ を出力し，停止する．
- H クエリ: $\mathcal{A}$ からのクエリ $(\mathrm{ID}, \mathrm{ID}', h)$ に対して，$\mathcal{B}$ は以下のように動作する：
  - $(\mathrm{ID}, \mathrm{ID}', *, h, K) \in L$ ならば，$\mathcal{B}$ は K を返す
  - そうでなければ，かつ $\phi(k) = k + \alpha$ に対して $(\mathrm{ID}, \mathrm{ID}', \phi, \perp, K) \in L$ ならば，$\mathcal{B}$ は以下のように動作する：
    - もし $\mathrm{ID} = \mathrm{ID}_i$（resp. $\mathrm{ID} = \mathrm{ID}_j$）ならば，$\mathcal{B}$ は h/X^α（resp. h/Y^α）が $(\mathrm{ID}, \mathrm{ID}')$ に関する正しい DDH 問題の答かどうか DDH オラクルを利用して確認する．
      - そうであれば（正しい DDH の答であれば），$\mathcal{B}$ は $(\mathrm{ID}, \mathrm{ID}', \phi, h, K)$ をリスト L に記録し K を返す．
      - そうでなければ，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，$(\mathrm{ID}, \mathrm{ID}', \perp, h, K)$ をリスト L に記録し K を返す．
    - もし $\mathrm{ID}' = \mathrm{ID}_A$（resp. $\mathrm{ID}' = \mathrm{ID}_B$）ならば，$\mathcal{B}$ は上述した Hon.Rev 内の説明と同様に動作する．

– そうでなければ，$\{\mathrm{ID}, \mathrm{ID}'\} = \{\mathrm{ID}_i, \mathrm{ID}_j\}$ かつ $h = g^{xy}$ ならば，$\mathcal{B}$ は h を DSDH 問題の解として出力する．ここで，$\mathcal{B}$ は $h = g^{xy}$ かどうかを DDH オラクルを用いてチェックできる．例えば，$\mathrm{DDH}_{g,X}(Y, h)$ とする．

– そうでなければ，$\mathcal{B}$ は $K \xleftarrow{\$} \{0,1\}^\lambda$ を選び，$(\mathrm{ID}, \mathrm{ID}', \perp, h, K)$ を L に記録し，K を返す．

$\mathcal{B}$ は $\mathcal{A}$ が guess ビットを出力するまで，$\mathcal{A}$ のオラクルを上述のようにシミュレートする．

最後に，$\mathcal{B}$ は $(\mathrm{ID}_i, \mathrm{ID}_j, \phi, h, *)$（resp. $(\mathrm{ID}_j, \mathrm{ID}_i, \phi, h, *)$）がリスト L に含まれるかどうかチェックする．ここで，$\phi(k) = k + \alpha$ である．もし含まれていれば，$\mathcal{B}$ は h/X^α（resp. h/Y^α）を DSDH 問題の解として出力する．その理由は後ほど説明する．そうでなければ（含まれていなければ），$\mathcal{B}$ は $\perp$ を出力する．以上が $\mathcal{B}$ の動作の説明である．

$\mathcal{B}$ が $\mathcal{A}$ によって Test にクエリされる正しい i と j を選ぶ限り，$\mathcal{B}$ は $\mathcal{A}$ の Extract クエリに正しく答えることが出来る．H をランダムオラクルとみなすため，$\mathcal{B}$ は $\mathcal{A}$ の Hon.Rev，Cor.Rev，および Test クエリにランダムな鍵を出力することで完璧に答えることができる．ここまでにおいて，$\mathrm{NIKE}_{\mathrm{fac}}$ を RKA-CKS-heavy の意味で攻撃する $\mathcal{A}$ のオラクルを，RSAgen に関する DSDH 問題を解く $\mathcal{B}$ がシミュレートできることを示した．

続いて，$\mathcal{B}$ は $\mathcal{A}$ を内部で用いて DSDH 問題を解けることを示す．つまり，$\mathcal{A}$ が NIKE 方式 $\mathcal{N}$ を破るならば，$\mathcal{B}$ は DSDH 問題が解けることを示す．上記の構成が示すように，$\mathcal{B}$ は $\mathcal{A}$ の Test クエリに対して乱数を返す．そのため，$\mathcal{A}$ の利得が無視可能でない（non-negligible）ためには，$\mathcal{A}$ は $\mathcal{B}$ によって選ばれた乱数と H の出力とを識別できなければならない．そうした理由から，$\mathcal{A}$ は $h = g^{x\phi^*(y)}$（または $h = g^{y\phi^*(x)}$）となる $(\mathrm{ID}_i, \mathrm{ID}_j, h)$（または $(\mathrm{ID}_j, \mathrm{ID}_i, h)$）をランダムオラクル H にクエリしている必要がある．ここで，$\phi^*(k) = k + \alpha$ である．$\mathcal{A}$ がこうしたクエリをしていたならば，値 h はリスト L に記録されており，$\mathcal{B}$ の DSDH 問題は h/X^α または h/Y^α を計算することで解くことができる．例えば，

$$\frac{h}{X^\alpha} = \frac{g^{x(y+\alpha)}}{g^{x\cdot\alpha}} = g^{xy}.$$

$\mathcal{B}$ は $h = g^{x\phi^*(y)}$（resp. $h = g^{y\phi^*(x)}$）が成り立つかどうか，オラクル $\mathrm{DDH}_{g,X}(\cdot,\cdot)$

(resp. $\mathrm{DDH}_{g,Y}(\cdot,\cdot)$) にアクセスすることで確かめることができる．つまり，例えば，$\mathcal{B}$ は $h = g^{x\phi^*(y)}$ かどうかのチェックを $\mathrm{DDH}_{g,X}(Yg^{\alpha}, h) = 1$ かどうかをチェックすることで行う．もし成り立てば，$\mathcal{B}$ は h/X^{α} を DSDH 問題の解として出力する．

$\mathcal{A}$ が任意の ϕ に対して $(\mathrm{ID}_i, \mathrm{ID}_j, \phi)$ を Test オラクルにクエリする確率は，$1/\binom{q_H}{2} = 2/q_H(q_H-1) \geq 2/q_H^2$ である．というのも，$\mathcal{B}$ による i および j の選択は，$\mathcal{A}$ の実行および $\mathcal{B}$ の入力とは独立だからである．$x, y \in \mathbb{Z}_{\varphi(n)/4}$ を用いるシミュレーションにおける g^x および g^y の分布は，$x, y \in \mathbb{Z}_{\lfloor n/4 \rfloor}$ を用いる実際の設定とは異なる．しかしながら，$\mathcal{B}$ の DSDH 問題の x および y の分布と，$\mathcal{A}$ の Φ^{lin}-RKA-CKS-heavy エクスペリメントの x および y の分布の間の統計的距離は無視可能である．これを ϵ と書いて，

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{dsdh}}_{\mathcal{B},\mathsf{RSAgen}}(\lambda) + \epsilon \\
&\geq 2\mathrm{Adv}^{\Phi\text{-rka-cks-heavy}}_{\mathcal{A},\mathcal{N}}(\lambda, q_H, q_C, q_E, q_{HR}, q_{CR})/q_H^2
\end{aligned}$$

を得る．以上で定理 4 が証明された． □

第4章　署名方式

本章では署名方式のシンタックスについて記し，署名方式の RKA 安全性について定義し，具体的な署名方式の RKA 安全性について議論する．

4.1　署名方式

署名方式 Σ は 3 つのアルゴリズム，鍵生成アルゴリズム，署名アルゴリズム，検証アルゴリズムからなる．

$$\Sigma = (\mathsf{KG}, \mathrm{Sign}, \mathrm{Verify})$$

と書き，これらアルゴリズムは以下の通りである：

$$\begin{aligned}(sk, vk) &\leftarrow \mathsf{KG}(1^\lambda),\\ \sigma &\leftarrow \mathrm{Sign}(m, sk),\\ 1/0 &\leftarrow \mathrm{Verify}(m, \sigma, vk),\end{aligned}$$

であり，sk, vk および σ はそれぞれ，署名鍵，検証鍵，署名である．任意のメッセージ m および KG で生成された任意の鍵ペア (sk, vk) に対して，以下のように正当性を満たさなければならない：

$$\mathrm{Verify}(m, \mathrm{Sign}(m, sk), vk) = 1.$$

4.2　署名の安全性

4.2.1　Φ-EUF-CM-RKA [3]

[3] に従って，選択メッセージおよび RKD 関数クラス Φ で定まる RKA に対する存在的偽造不可能性を記す．署名方式の安全性は Φ-EUF-CM-RKA と書き，以下のアルゴリズム $\mathcal{A}$ とアルゴリズム $\mathcal{B}$ との間のゲームで定式化される：

初期設定. アルゴリズム $\mathcal{B}$ は $\mathsf{KG}(1^\lambda)$ を走らせ署名鍵 sk および検証鍵 vk を得る．$\mathcal{B}$ はリスト $M \leftarrow \emptyset$ を用意する．それから，$\mathcal{B}$ は vk を $\mathcal{A}$ に渡す．

RKA 署名オラクルクエリ. $\mathcal{A}$ からの適応的なクエリ (m_i, ϕ_i) に対して，$\mathcal{B}$ は署名

$$\sigma_i \leftarrow \mathrm{Sign}(m_i, \phi_i(sk))$$

を出力する．ここで $\phi_i \in \Phi$ である．$\phi_i(sk) = sk$ ならば，$\mathcal{B}$ は m_i をリスト M に記録する．

出力. $\mathcal{A}$ は (m^*, σ^*) を出力するとする．

$\mathrm{Verify}(m^*, \sigma^*, vk) = 1$ かつ $m^* \notin M$ ならば，$\mathcal{B}$ は 1 を出力する．さもなくば，$\mathcal{B}$ は 0 を出力する．

F を，上述のゲームで $\mathcal{B}$ の出力が 1 であるイベントとする．Φ-EUF-CM-RKA 安全性に対する $\mathcal{A}$ の利得を

$$\mathrm{Adv}^{\Phi\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[F]$$

と定義する．利得 $\mathrm{Adv}^{\Phi\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda)$ がいかなる多項式時間アルゴリズム $\mathcal{A}$ に対しても無視可能であれば，署名方式 Σ は Φ-EUF-CM-RKA 安全であるという．

この安全性定義は，(m_i, ϕ_i) が RKA 署名オラクルにクエリされたものであったとしても，$\phi_i(sk) \neq sk$ であるかぎり $m^* = m_i$ として再利用してよいという意味で，強い安全性定義である．RKA は署名アルゴリズムに用いられる署名鍵に作用さるということを強調したい．つまり，検証アルゴリズムの変更については考慮せず，RKA の成功とは，元の検証鍵のもとで正当な署名を生成することである．

4.2.2 Φ-wEUF-CM-RKA

適応的選択メッセージ攻撃に対する弱存在的偽造不可能性 [26] および，RKA に対するメッセージ認証コードの弱存在的偽造不可能性 [8] に従って，本論文では上の定義よりも弱い安全性定義を提案する．上記の安全性のエクスペリメントにおける攻撃者が，これまでに RKA 署名オラクルにクエリされていない m^* に対する偽

造を生成するよう求めることで，より弱い安全性定義である Φ-wEUF-CM-RKA を得る．

いくつかのシナリオにおいてはその弱い安全性定義 Φ-wEUF-CM-RKA が安全性を保証するのに十分だという議論もあろうが，過去の研究の文脈における標準的な安全性定義は上に定めた Φ-EUF-CM-RKA に対応することを注記したい．Schnorr 署名方式は $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全だが，4.5.1 章で説明するように，Φ^{lin}-EUF-CM-RKA に関しては脆弱であることを示す．4.6.1 章に記す改良 Schnorr 署名方式は，$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全であることを示す．また，DSA の変形として知られる方式の一つ [38] は，$\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全であるが，オリジナルの DSA は 4.5.2 章で説明するように Φ^{lin}-EUF-CM-RKA に関して脆弱であることを示す．その変形 DSA が $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA に関して脆弱かどうかは知られていないが，4.7.1 章に記す改良 DSA は $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全であることが証明される．さらに修正版 ElGamal 署名方式 [36] は $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全であるが，オリジナルの ElGamal 署名方式は，4.5.3 章で説明するように，Φ^{lin}-EUF-CM-RKA に関して脆弱であることを示す．また， 4.8.1 章に記す改良 ElGamal 署名式方は，$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全であることが証明される．さらなる詳細は， 4.5 章，4.6 章，4.7 章，4.8 章を参照のこと．

しばしばフォルト攻撃と呼ばれる RKA 安全性のより強いモデルが，ラウンドベースの共通鍵暗号方式 [9, 16] および [25] で考えられてきた．これらモデルにおいては，暗号化アルゴリズムの各ラウンドにおいて，攻撃者はフォルト（入力や内部状態の変更）を注入することが可能であり，それは例えば秘密鍵の回復につながったりする．署名アルゴリズムの実行時にいつ署名鍵を変更するか選べる攻撃者を考えるような，上記と同様の拡張を署名方式においても考えることも出来る．しかし，本論文では上に記したような標準的な RKA 定義（とそれよりも弱い安全性定義）に焦点を合わせる．

4.3 具体的な署名方式

4.3.1 Schnorr 署名方式

Schnorr 署名方式は Schnorr によって 1989 年に提案され [40]，離散対数仮定のもとでランダムオラクルモデルで安全であることが証明された [36]．$\mathbb{G}$ を素数位数 q の群とする．鍵生成アルゴリズム，署名アルゴリズム，検証アルゴリズムの 3 つのアルゴリズムは以下のように定義される：

素数 q に対して $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ とする．

- KG: このアルゴリズムは，入力として 1^λ をとり，署名鍵 sk および検証鍵 vk を以下のように生成する：

  1. 生成元 $g \xleftarrow{\$} \mathbb{G}$ を選ぶ．
  2. $x \xleftarrow{\$} \mathbb{Z}_q$ を選び $y \leftarrow g^x$ とする．
  3. ハッシュ関数 $H : \{0,1\}^* \to \mathbb{Z}_q$ を選ぶ．
  4. $sk = x, vk = (g, y, H)$ を出力する．

- Sign: このアルゴリズムは，入力として $m \in \{0,1\}^*$ および署名鍵 sk をとり，署名 σ を以下のように生成する：

  1. $t \xleftarrow{\$} \mathbb{Z}_q$ を選び $r \leftarrow g^t$ とする．
  2. $h \leftarrow H(m \,\|\, r)$ とする．
  3. $s \leftarrow x \cdot h + t \bmod q$ とする．
  4. $\sigma \leftarrow (h, s)$ を出力する．

- Verify: このアルゴリズムは，入力として m，署名 σ，および検証鍵 vk をとり，署名を以下のように検証する：

  1. $r' \leftarrow g^s y^{-h}$ とする．
  2. $h' \leftarrow H(m \,\|\, r')$ とする．
  3. $h' = h$ ならば 1 を出力し，そうでなければ 0 を出力する．

4.3.2 DSA

DSA は米国の署名方式スタンダード（Digital Signature Standard）として 1994 年に提案された [34]．まずは，オリジナルの DSA を記す．

p および q を素数とする．ここで q は $p-1$ の素因数とする．DSA は以下の 3 つのアルゴリズムで定義される：

- KG: このアルゴリズムは，入力として 1^λ をとり，署名鍵 sk および検証鍵 vk を以下のように生成する：
  1. 生成元 $g \in \mathbb{Z}_p^*$ を選ぶ．
  2. $x \xleftarrow{\$} \mathbb{Z}_q^*$ を選び，$y \leftarrow g^x \bmod p$ とする．
  3. ハッシュ関数 $H : \{0,1\}^* \to \mathbb{Z}_q$ を選ぶ．
  4. $sk = x, vk = (g, y, H)$ を出力する．
- Sign: このアルゴリズムは，入力として $m \in \{0,1\}^*$ および署名鍵 sk をとり，署名 σ を以下のように生成する：
  1. $t \xleftarrow{\$} \mathbb{Z}_q^*$ を選び，$r \leftarrow (g^t \bmod p) \bmod q$ とする．
  2. $s \leftarrow t^{-1}(H(m) + x \cdot r) \bmod q$ とする．
  3. $\sigma \leftarrow (r, s)$ を出力する．
- Verify: このアルゴリズムは，入力として m，署名 $\sigma = (r, s)$，および検証鍵 $vk = (g, y, H)$ をとり，署名を以下のように検証する：
  1. $r' \leftarrow (g^{H(m)/s} y^{r/s} \bmod p) \bmod q$ とする．
  2. $r' = r$ ならば 1 を出力し，そうでなければ 0 を出力する．

DSA の変型． オリジナルの DSA の安全性は証明されていないが，Pointcheval および Vaudenay [38] は DSA の 2 つの変型は通常の安全性の意味で安全であることをランダムオラクルモデルにおいて証明した．一つ目の DSA の変型は追加で一つランダムオラクル H' を利用し，署名アルゴリズムの第一ステップで $r \leftarrow H'(g^t \bmod p)$ を計算する．二つ目の DSA の変型の主な違いは，ハッシュ関数が入力としてメッ

セージだけではなく値 r もとることである．この後 4.7 において，この二つ目の変型に軽微な修正を加えた方式を考える．

DSA で用いる $\mathbb{Z}_p^*$ から $\mathbb{Z}_q$ への写像の衝突耐性.

DSA の署名アルゴリズムのステップ 1 において，要素 $g^t \in \mathbb{Z}_p^*$ を要素 $r \in \mathbb{Z}_q$ に写像しなければならないことを注記したい．[38] において Pointcheval および Vaudenay はこの写像を $\mathbb{G}$ から $\mathbb{Z}_q$ への抽象的な関数と考えていた．ここで $\mathbb{G}$ は $\mathbb{Z}_p^*$ の部分群であり位数 q である．彼らの DSA の二つ目の変型の安全性証明をするために，Pointcheval および Vaudenay は，この関数がある種の衝突耐性を持つと仮定した．本論文では [38] と同様のアプローチをとり，この関数を $F_{p,q}$ と書き次の性質を持つと仮定する：

$F_{p,q} : \mathbb{G} \to \mathbb{Z}_q$ を $\overline{g} \mapsto \overline{g} \bmod q$ によって定まる写像とする．ここで $\overline{g} \in \mathbb{G}$ であり，$\mathbb{G}, q, p$ は DSA で用いるパラメータとする（つまり，$\mathbb{G}$ は $\mathbb{Z}_p^*$ の位数 q の部分群である）．どんな確率的多項式アルゴリズム $\mathcal{A}$ も $F_{p,q}(\overline{g}_1) = F_{p,q}(\overline{g}_2)$ となるような二つの異なる元 $\overline{g}_1, \overline{g}_2 \in \mathbb{G}$ を確率 ϵ 以上で見つけられないならば，$F_{p,q}$ は ϵ-衝突困難であるという．ϵ はセキュリティパラメータに関して無視可能であれば，単に $F_{p,q}$ は衝突困難であるという．さらに，ランダムな入力に対する $F_{p,q}$ の値が，任意に与えられた値と等しくなる確率は高々 $\sqrt{\epsilon + 1/q}$ [1]である．

4.3.3 ElGamal 署名方式

ElGamal 署名方式は [19] にて提案された．まず，オリジナルの方式を記す．

p を素数とする．ElGamal 署名方式は以下の 3 つのアルゴリズムで定義される：

- KG: このアルゴリズムは，入力として 1^λ をとり，署名鍵 sk および検証鍵 vk を以下のように生成する：

  1. 生成元 $g \in \mathbb{Z}_p^*$ を選ぶ.

[1] こうならないと仮定する．つまり，$\Pr_{\overline{g} \leftarrow \mathbb{G}}[F_{p,q}(\overline{g}) = r^*] \geq \sqrt{\epsilon + 1/q}$ となるような値 r^* が存在するとする．また，2 つのランダムに選ばれた入力 $\overline{g}, \overline{g}' \in \mathbb{G}$ を考える．$\overline{g}$ および $\overline{g}'$ は独立に選ばれるため，$\Pr[\overline{g} \neq \overline{g}' \wedge F_{p,q}(\overline{g}) = r^* \wedge F_{p,q}(\overline{g}') = r^*] \geq \Pr[F_{p,q}(\overline{g}) = r^* \wedge F_{p,q}(\overline{g}') = r^*] - \Pr[\overline{g} = \overline{g}'] \geq (\sqrt{\epsilon + 1/q})^2 - 1/q = \epsilon$ を得る．しかし，これは $F_{p,q}$ の ϵ 衝突困難性に反する．

2. $x \xleftarrow{\$} \mathbb{Z}_{p-1}$ を選び，$y \leftarrow g^x \bmod p$ とする．

3. $sk = x, vk = (g, y)$ を出力する．

- Sign: このアルゴリズムは，入力として $m \in \mathbb{Z}_{p-1}$ および署名鍵 sk をとり，署名 σ を以下のように生成する：

1. $t \xleftarrow{\$} \mathbb{Z}^*_{p-1}$ を選び，$r \leftarrow g^t \bmod p$ とする．

2. $s \leftarrow t^{-1}(m - x \cdot r) \bmod (p-1)$ とする．

3. $\sigma \leftarrow (r, s)$ を出力する．

- Verify: このアルゴリズムは，入力として m，署名 $\sigma = (r, s)$，および検証鍵 $vk = (g, y)$ をとり，署名を以下のように検証する：

1. $g^m = y^r r^s \bmod p$ ならば 1 を出力し，そうでなければ 0 を出力する．

ElGamal 署名方式の変型.

Pointcheval および Stern [36] によって存在的偽造攻撃が可能であることが指摘されているため，オリジナルの方式が安全でないことはよく知られている．一方で，ElGamal 署名方式の変型はランダムオラクルモデルにおいて通常の安全性の意味で安全である．その変型は修正 ElGamal 署名方式（modified ElGamal signature scheme）と呼ばれている [36]．ElGamal 署名方式がメッセージ m の値をそのまま使っているのに対し，修正 ElGamal 署名方式は，入力としてメッセージ m および値 r をとり λ ビット列を出力するハッシュ関数を用いている．

本論文では，この修正 ElGamal 署名方式に軽微な改良を加えた改良方式を 4.8 章において考える．

4.4 署名方式の wRKA 安全性

本節では，既存の署名方式である Schnorr 署名方式，[38] における二つ目の DSA の変型，および [36] の修正 ElGamal 署名方式がすでに $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全であることを示す．なお，$\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全性は偽造でのメッセージ m^* は RKA 署名オラクルにクエリされておらず新しくなければならない．

まず，Schnorr 署名方式に関する以下の定理を示す．

定理 5. d を正の整数とする．$\mathbb{G}$ 上の d-SDL 仮定のもと，ランダムオラクルモデルにおいて Schnorr 署名方式は $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全である．

より正確には，動作時間 $t_{\mathcal{A}}$ で，q_S 回の RKA 署名オラクルクエリをし，H への q_H 回のランダムオラクルクエリをする任意のアルゴリズム $\mathcal{A}$ に対して，動作時間 $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ であるアルゴリズム $\mathcal{B}$ が存在して，以下の式を満たす：

$$\begin{aligned}&\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda)\\ &\leq \left((q_H + q_S)\left(\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) + \frac{2q_S + 1}{q}\right)\right)^{1/2}.\end{aligned}$$

証明は 4.6.2 章の定理 8 の証明とよく似ている．二つの証明の違いを定理 8 の証明の後にハイライトする．

次に，二つ目の DSA の変型 [38] に関する以下の定理を示す．

定理 6. d を正の整数とし，写像 $F_{p,q}$ は衝突困難と仮定する．$\mathbb{G}$ 上の d-SDL 仮定のもと，二つ目の DSA の変型はランダムオラクルモデルにおいて $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全である．

より正確には，$F_{p,q}$ を ϵ 衝突困難とすると，動作時間 $t_{\mathcal{A}}$ で，q_S 回の RKA 署名オラクルクエリをし，H への q_H 回のランダムオラクルクエリをする任意のアルゴリズム $\mathcal{A}$ に対して，動作時間 $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ であるアルゴリズム $\mathcal{B}$ が存在して，以下の式を満たす：

$$\begin{aligned}&\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda)\\ &\leq \left((q_H + q_S)\left(\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) + \frac{1}{q}\right) + \right.\\ &\qquad \left. 2\left(\epsilon + \frac{q_S^2}{q} + q_H\sqrt{\epsilon + \frac{1}{q}}\right)\right)^{1/2}.\end{aligned}$$

証明は 4.7.2 章の定理 10 の証明とよく似ているため，定理 10 のあとにその違いをハイライトする．

最後に，[36] で提案された修正 ElGamal 署名方式に関する以下の定理を示す．以下の定理において，p はセキュリティパラメータ λ に対して $p > 2^{\lambda}$ かつ $p-1 = qR$ となるような長さ n の素数とする．ここで q は素数で，固定された α に対して $R \leq n^{\alpha}$ とする．

定理 7. d を正の整数とする．$\mathbb{Z}_p^*$ 上の d-SDL 仮定のもと，修正 ElGamal 署名方式はランダムオラクルモデルにおいて $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全である．

より正確には，動作時間 $t_{\mathcal{A}}$ で，q_S 回の RKA 署名オラクルクエリをし，H に対して q_H 回のランダムオラクルクエリをする任意のアルゴリズム $\mathcal{A}$ に対して，動作時間 $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ であるアルゴリズム $\mathcal{B}$ が存在して，以下の式を満たす：

$$\begin{aligned}
&\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) \\
&\leq \left(R(q_H + q_S)^2 \mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) \left(1 - \frac{1}{q}\right)^{-3} \right. \\
&\qquad \left. + \frac{2(q_S+1)(q_S+q_H)}{qR} + \frac{4q_S}{q} \right)^{1/4}
\end{aligned}$$

なお，証明は 4.8.2 章の定理 12 の証明と同様である．

4.5 署名方式に対する関連鍵攻撃

4.4 章において，wRKA 安全性について議論した．一方，本章では，Schnorr 署名方式，DSA，および ElGamal 署名方式に対して，wRKA 安全性よりも強い安全性である RKA 安全性について記す．2.4 章で言及したように，RKD 関数として用いる線形関数は署名鍵空間として用いられる群に依って，加算か乗算のどちらかで表される．

4.5.1 Schnorr 署名方式に対する関連鍵攻撃

Schnorr 署名方式は線形関数（ここでは加算）に関する RKA 安全ではないことを，シンプルで効率的な攻撃を与えることで示す．つまり，Schnorr 署名方式は Φ^{lin}-EUF-CM-RKA 安全ではないことを示す.

攻撃者 $\mathcal{A}$ は以下のように署名を偽造する：

1. 任意のメッセージ $m' \in \{0,1\}^*$ および任意の値 $b \in \mathbb{Z}_q^*$ を選ぶ.
2. $(m', \phi(x) = x - b)$ を RKA 署名オラクルにクエリし，その返答として署名 (h', s') を得る．

3. メッセージ m' および偽造 $(h', s' + b \cdot h')$ を出力する．

では，この偽造がうまくいくことを確かめる．まず，RKA 署名オラクルからの返答 (h', s') は次のようなプロセスで計算されているはずである：

- $t' \xleftarrow{\$} \mathbb{Z}_q$ を選び，$r' \leftarrow g^{t'}$ とする．
- $h' \leftarrow H(m' \,\|\, r')$ とする．
- $s' \leftarrow (x - b) \cdot h' + t' \bmod q$ とする．

このようにしてメッセージ m' に対して偽造された署名 $(h', s' + b \cdot h')$ は以下のように検証に通る．

$$\begin{aligned} r'' &= g^{s'+b\cdot h'} y^{-h'} = g^{(x-b)\cdot h'+t'+b\cdot h'} y^{-h'} \\ &= g^{(x-b)\cdot h'+t'+b\cdot h'-x\cdot h'} = g^{t'} = r'. \end{aligned}$$

4.5.2 DSA に対する関連鍵攻撃

次に，DSA が線形関数（ここでは乗算）に関する RKA 安全ではないことをシンプルで効率的な攻撃を与えることで示す．つまり，DSA は Φ^{lin}-EUF-CM-RKA 安全ではないことを示す．

攻撃者 $\mathcal{A}$ は以下のように署名を偽造する：

1. 異なる 2 つのメッセージ $m_0, m_1 \in \{0,1\}^*$ を選び，$z_0 \leftarrow H(m_0), z_1 \leftarrow H(m_1)$ とする．
2. $a \leftarrow \dfrac{z_1}{z_0} \bmod q$ とする．
3. RKA 署名オラクルに $(m_1, \phi(x) = ax)$ をクエリし，署名 $(r, s = t^{-1}(z_1 + axr))$ を得る．
4. メッセージ $m^* = m_0$ および署名 $(r^*, s^*) = (r, \dfrac{s}{a} \bmod q)$ を出力する．

たとえ a が 1 でもこの攻撃はうまくいくことに注意が必要である．

メッセージ m_0 に対する偽造された署名 $(r, \frac{s}{a} \bmod q)$ は以下のように検証に通る．まず w^* の計算は次の通り：

$$\begin{aligned} w^* = (s^*)^{-1} = \frac{a}{s} &= \frac{ta}{z_1 + axr} \\ &= \frac{ta}{a \cdot z_0 + axr} = \frac{t}{z_0 + xr}. \end{aligned}$$

それから，$u_1 = w^* z_0 \bmod q$ and $u_2 = r w^* \bmod q$ を計算し，最後に以下をチェックする．

$$\begin{aligned} r' &= (g^{H(m_0)/s^*} y^{r^*/s^*} \bmod p) \bmod q \\ &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= (g^{w^* z_0} y^{r w^*} \bmod p) \bmod q \\ &= (g^{w^* z_0 + x r w^*} \bmod p) \bmod q \\ &= (g^{w^*(z_0 + xr)} \bmod p) \bmod q \\ &= (g^t \bmod p) \bmod q \\ &= r. \end{aligned}$$

よって，$\mathcal{A}$ により出力された偽造は正当である．

4.5.3 ElGamal 署名方式に対する関連鍵攻撃

ElGamal 署名方式は通常の選択メッセージ攻撃に対して脆弱だということが知られている．[36] で示されたよく知られた攻撃は，r および s の値に左右されたメッセージに対して署名を偽造する．以下では RKA によって，任意のメッセージに対して偽造が可能であることを示す．

攻撃者 $\mathcal{A}$ は以下のように署名を偽造する．

1. 任意のメッセージ $m' \in \{0,1\}^*$ および任意の値 $a \in \mathbb{Z}_{p-1}^* \setminus \{1\}$ を選ぶ．
2. $(am', \phi(x) = ax)$ を RKA 署名オラクルにクエリし，返答として署名 (r', s') を得る．
3. メッセージ m' および偽造 $(r', s'/a)$ を出力する．

では，この偽造がうまくいくことを確かめる．まず，RKA 署名オラクルからの返答 (r', s') は次のようなプロセスで計算されているはずである：

- $t' \xleftarrow{\$} \mathbb{Z}_{p-1}^*$ を選び，$r' \leftarrow g^{t'} \bmod p$ とする．
- $s' \leftarrow (t')^{-1}(a \cdot m' - ax \cdot r') \bmod p - 1$ とする．

メッセージ m' に対して偽造された署名 $(r', s'/a)$ は以下のように検証に通る．

$$\begin{aligned} y^{r'}(r')^{s'/a} &= y^{r'}(r')^{(t')^{-1}(a \cdot m' - ax \cdot r')/a} \\ &= y^{r'}(g^{t'})^{(t')^{-1}(m' - x \cdot r')} \\ &= y^{r'} g^{(m' - x \cdot r')} \\ &= g^{m'}. \end{aligned}$$

ゆえに，偽造は正当である．trivial win を避けるために，値 a は $a \neq 1$ を満たさなければならなかったことに注意．

4.6 改良 Schnorr 署名方式とその RKA 安全性

4.5.1 章に示したように，オリジナルの Schnorr 署名方式は線形関数に関する RKA 安全ではない．本章では，軽微な修正を加えることで，多項式に関する RKA 安全な署名方式を構成できることを示す．これを改良 Schnorr 署名方式と呼ぶことにする．

4.6.1 構成法

ここでは Schnorr 署名方式に対する本論文で提案する軽微な修正を示す．なお，ハッシュ関数は，検証鍵の再計算に相当する追加の入力をとるよう修正される．$\mathbb{G}$ は素数位数 q の群とする．改良 Schnorr 署名方式は以下のように定義される：

- KG: このアルゴリズムは，入力として 1^λ をとり，署名鍵 sk および検証鍵 vk を以下のように生成する：
  1. 生成元 $g \xleftarrow{\$} \mathbb{Z}_q$ を選ぶ．
  2. $x \xleftarrow{\$} \mathbb{Z}_q$ を選び，$y \leftarrow g^x$ とする．

3. ハッシュ関数 $H : \{0,1\}^* \to \mathbb{Z}_q$ を選ぶ.

4. $sk = x$ および $vk = (g, y, H)$ を出力する.

- Sign: このアルゴリズムは，入力として $m \in \{0,1\}^*$ および署名鍵 sk をとり，署名 σ を以下のように生成する：

  1. $t \xleftarrow{\$} \mathbb{Z}_q$ を選び $r \leftarrow g^t$ とする.

  2. $\hat{y} \leftarrow g^x$ とする.

  3. $h \leftarrow H(m \,\|\, r \,\|\, \hat{y})$ を得る.

  4. $s \leftarrow x \cdot h + t \bmod q$ とする.

  5. $\sigma \leftarrow (h, s)$ を出力する.

- Verify: このアルゴリズムは，入力としてメッセージ m，署名 σ，および検証鍵 vk をとり，署名を以下のように検証する：

  1. $r' \leftarrow g^s y^{-h}$ とする.

  2. $h' \leftarrow H(m \,\|\, r' \,\|\, y)$ とする.

  3. $h' = h$ ならば 1 を出力し，そうでなければ 0 を出力する.

署名アルゴリズムの第二ステップにおける値 $\hat{y} \leftarrow g^x$ の計算は，単に検証鍵 y を $\hat{y}$ として用いることにしてはならないことに注意が必要である．つまり，署名アルゴリズムは署名の計算の度に $\hat{y} = g^x$ を計算する.

署名アルゴリズムにおいて，検証鍵を署名鍵から再計算しているため,（元の検証鍵が署名アルゴリズムで利用できるという仮定をして）再計算された検証鍵と元の検証鍵とを比較することで RKA 安全性を達成できるとも考えられるかもしれない．しかしながらこのアプローチは，署名者が保持する元の検証鍵が不変であるという仮定のもとでしかうまくいかない．この仮定は，ある意味で RKA モデルの前提，つまり攻撃者は署名者によって保持されている鍵情報に変更を加えることができるとしている点と矛盾している．それとは対照的に，本章および 4.7.1 章で記す提案方式は，保持する値に関して何の追加の仮定もおかずに RKA 安全であることが示される.

4.6.2 安全性証明

改良 Schnorr 署名方式についての以下の定理を証明する．

定理 8. d を正の整数とする．$\mathbb{G}$ 上の d-SDL 仮定のもと， 4.6.1 章の署名方式はランダムオラクルモデルにおいて $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全である．

より正確には，動作時間 $t_{\mathcal{A}}$ で，q_S 回の RKA 署名オラクルクエリをし，H への q_H 回のランダムオラクルクエリをする任意のアルゴリズム $\mathcal{A}$ に対して，動作時間 $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ であるアルゴリズム $\mathcal{B}$ が存在して，以下の式を満たす：

$$\begin{aligned} &\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) \\ &\leq \left((q_H + q_S)\left(\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) + \frac{2q_S + 1}{q}\right)\right)^{1/2}. \end{aligned} \tag{4.1}$$

証明. $\mathcal{A}$ を任意の確率的多項式時間アルゴリズム（動作時間 $t_{\mathcal{A}}$）で，改良 Schnorr 署名方式を q_S 回の RKA 署名オラクルクエリ，H に対する q_H 回のランダムオラクルクエリを行うことで $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全性の意味で攻撃する，攻撃者とする．d-SDL 問題を解くアルゴリズム $\mathcal{B}$ をどのように構成するかを示し，それから上述の $\mathcal{A}$ の利得と $\mathcal{B}$ の利得の間の不等式を示す（式 (4.1)）．

まず，一般 forking lemma を用いるために，以下を定める．IG を入力生成器とする．これは $\mathcal{G} \leftarrow (\mathbb{G}, q)$ を決定し，生成元 $g \xleftarrow{\$} \mathbb{Z}_q$ を選び，$x \xleftarrow{\$} \mathbb{Z}_q$ を選び，$y_1 \leftarrow g^x, y_2 \leftarrow g^{x^2}, \ldots, y_d \leftarrow g^{x^d}$ を計算し，$X = (\mathcal{G}, g, y_1, y_2, \ldots, y_d)$ を出力する．集合 Z を $\mathbb{Z}_q$ とする．また整数 Q を $q_H + q_S$ とする．$\mathcal{F}$ を，$\mathcal{A}$ の乱数 $\rho_{\mathcal{A}}$ からなる乱数 $\rho_{\mathcal{F}}$ および $q_H + q_S$ 個の値 $s_1, \ldots, s_{q_H+q_S} \in \mathbb{Z}_q$ を用い，入力として $X = (\mathcal{G}, g, y_1, y_2, \ldots, y_d)$ および $h_1, \ldots, h_{q_H+q_S} \in \mathbb{Z}_q$ をとり（どれも一様ランダムに選択されると仮定される），内部で $\mathcal{A}$ を以下のように走らせる：

$\mathcal{F}$ の動作．$\mathcal{F}$ は $vk \leftarrow (g, y_1, H)$ とする．ここで H は $\mathcal{F}$ が $\mathcal{A}$ にハッシュオラクルへのアクセスをゆるすことを意味する．

初期設定. $\mathcal{F}$ は以下のように動作する．

- $L_H \leftarrow \emptyset, M \leftarrow \emptyset$ とする．

- 入力 vk および一様ランダムに選ばれた乱数 $\rho_{\mathcal{A}}$ を用いて $\mathcal{A}$ を走らせる．

$\mathcal{A}$ はオラクルに対して $q_H + q_S$ 回のクエリをする：

ハッシュオラクルクエリ． i 番目（$1 \leq i \leq q_H + q_S$）のクエリがハッシュオラクルに対するもので，(m_i, r_i, y_i) という形であれば，$\mathcal{F}$ は以下のように返答する：

- ある $h \in \mathbb{Z}_q$ に対して $(m_i, r_i, y_i, h, \cdot)$ が L_H に記録されているならば（$\cdot$ は任意の値），$\mathcal{F}$ は h を返す．
- そうでなければ，$\mathcal{F}$ は $(m_i, r_i, y_i, h_i, \perp)$ をリスト L_H に記録し，h_i を返す．

RKA 署名オラクルクエリ． i 番目（$(1 \leq i \leq q_H + q_S$）のクエリが RKA 署名オラクルに対するもので，(m_i, ϕ_i) という形であれば，$\mathcal{F}$ は以下の値を計算する：

$$r_i \leftarrow g^{s_i}(g^{\phi_i(x)})^{-h_i}.$$

ここで $\phi_i(x) = a_d \cdot x^d + a_{d-1} \cdot x^{d-1} + \cdots + a_1 \cdot x + a_0$ である．

$$\begin{aligned} r_i &= g^{s_i}(g^{\phi_i(x)})^{-h_i} \\ &= g^{s_i - a_0 \cdot h_i}(y_d^{a_d} \cdot y_{d-1}^{a_{d-1}} \cdots y_1^{a_1})^{-h_i} \end{aligned}$$

であるため，アルゴリズム $\mathcal{F}$ は自身の入力を用いて r_i を計算することができる．

このとき，

- すでに $(m_i, r_i, g^{\phi_i(x)}, \cdot, \cdot)$ という形のエントリがリスト L_H にあれば，$\mathcal{F}$ は $(0, \perp)$ を出力して停止する．
- そうでなければ，$\mathcal{F}$ は $(m_i, r_i, g^{\phi_i(x)}, h_i, s_i)$ を L_H に記録する．ここで h_i は $\mathcal{F}$ の入力の一部であり，s_i は $\mathcal{F}$ の乱数 $\rho_{\mathcal{F}}$ の一部である．それから署名 $\sigma_i = (h_i, s_i)$ を $\mathcal{A}$ に返す．h_i は $\mathbb{Z}_q$ からランダムに選ばれており，s_i は $r_i = g^{s_i}(g^{\phi_i(x)})^{-h_i}$ を満たすため，上のように作ったの署名は署名アルゴリズムが作る署名と同じ確率分布で作られることに注意が必要である．

出力． $\mathcal{A}$ がメッセージ m^* および偽造署名 $\sigma^* = (h^*, s^*)$ を出力するとき，$\mathcal{F}$ は検証鍵とリスト M を用いて検証を行う．メッセージ m^* に対する偽造署名 $\sigma^* = (h^*, s^*)$

に対して，$\mathcal{A}$ はハッシュオラクルクエリ (m^*, r^*, y) をしたと仮定する．ここで

$$r^* = g^{s^*} y^{-h^*}$$

である．なお，$\mathcal{A}$ がこのハッシュオラクルクエリをすると仮定しても，一般性を失わない．なぜなら，ハッシュオラクルに (m^*, r^*, y) をクエリする攻撃者 $\mathcal{A}'$ を，ハッシュオラクルに (m^*, r^*, y) をクエリしないかもしれない $\mathcal{A}$ を用いて常に構成できるためである．攻撃者 $\mathcal{A}'$ はただ一つのハッシュオラクルクエリを増やすだけであることに注意が必要である．それゆえに，$\mathcal{F}$ は $h \in \mathbb{Z}_q$ に対して $(m^*, r^*, y, h, \cdot)$ を常にリスト L_H 内で見つける．$J \in \{1, \ldots, q_H + q_S\}$ をこの過程で見つかる $h = h_J$ となるインデックスとする．

- $h^* = h_J$ および $(m^*, r^*, y, h^*, \cdot)$ の最後の要素が $\perp$ ならば（つまり，h^* がハッシュオラクルによって計算されたが，RKA 署名オラクルでは計算されていないならば），$\mathcal{F}$ は $(J, V = (h^*, s^*))$ を出力する．この場合が意味するのは，h^* が等式 $h^* = H(m^*, r^*, y)$ を満たすため，$\mathcal{F}$ によってシミュレートされたエクスペリメントにおいて，$\sigma^* = (h^*, s^*)$ は m^* に対する正当な署名であるということである．
- そうでなければ，$\mathcal{F}$ は $(0, \perp)$ を出力する．

以上が $\mathcal{F}$ の動作の手順である．

$\mathcal{F}$ が $\mathcal{A}$ の RKA 署名オラクルクエリの返答に失敗する確率は高々 $q_S(q_H + q_S)/q$ であることがわかる．その理由は次の通りである．まず，$\mathcal{A}$ が i 番目のクエリをする段階では，s_i および h_i の値が $\mathcal{A}$ から隠されているため，$\mathcal{A}$ が $\mathcal{F}$ によって計算された隠れた値 $r_i = g^{s_i} \cdot (g^{\phi(x)})^{-h_i}$ を計算することができない（無視可能な確率で成功する場合を除く）．そのため，$\mathcal{A}$ が RKA 署名オラクルクエリをするときにリスト L_H がすでにエントリ $(m_i, r_i, g^{\phi(x)}, \cdot, \cdot)$ を含んでいる確率は高々 $(q_H + q_S)/q$ である．ここで $q_H + q_S$ は L_H のサイズを表している．このとき，$\mathcal{A}$ は高々 q_S 回の RKA 署名オラクルクエリをするため，$\mathcal{F}$ が $\mathcal{A}$ の RKA 署名オラクルクエリの返答に失敗する確率は高々 $q_S(q_H + q_S)/q$ である．なお，$\mathcal{F}$ は $\mathcal{A}$ の署名オラクルクエリの返答に失敗しない限り，ランダムオラクルモデルにおいて，$\mathcal{A}$ に対して EUF-CM-RKA エクスペリメントを完全にシミュレートしていることに注意が必要である．もし $\mathcal{A}$ が

正当な偽造ペア $(m^*, \sigma^* = (h^*, s^*))$ を出力することができ，$\mathcal{F}$ が RKA 署名オラクルクエリの返答に失敗しなければ，$\mathcal{F}$ は常に， $J \geq 1$ かつ $h^* = h_J$ となるような $(J, V = (h^*, s^*))$ を出力する．$\mathcal{F}$ がこのような (J, V) を出力する確率は **acc** であるので，

$$\mathbf{acc} \geq \mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) - \frac{q_S(q_H + q_S)}{q}$$

を得る．一般 forking lemma（補題 1）によって，forking アルゴリズム $\mathcal{B}_{\mathcal{F}}$ が forking に成功する（つまり，補題のステップ 7 で停止する）確率 **frk** について，以下の不等式を得る．

$$\mathbf{frk} \geq \mathbf{acc} \cdot \left(\frac{\mathbf{acc}}{q_H + q_S} - \frac{1}{q} \right).$$

それゆえ，

$$\mathbf{frk} \geq \frac{(\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda))^2}{q_H + q_S} - \frac{2q_S + 1}{q}$$

を得る．

次に，**frk** を d-SDL 問題を解くアルゴリズム $\mathcal{B}$ の利得に関連付ける．

$\mathcal{B}$ の動作．$\mathcal{B}$ は入力として $\mathcal{G}$ および $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$ をとり，$X = (\mathcal{G}, (g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}))$ とする．アルゴリズム $\mathcal{B}$ は，上述した $\mathcal{F}$ に関する forking の手順 $\mathcal{B}_{\mathcal{F}}$ を行う．(δ, V, V') を $\mathcal{B}_{\mathcal{F}}$ の出力とする．$\delta = 0$ ならば，$\mathcal{B}$ は停止する．そうでなければ，$V = (h^*_{(1)}, s^*_{(1)})$ かつ $V' = (h^*_{(2)}, s^*_{(2)})$ とする．ここでは，$h^*_{(1)} \neq h^*_{(2)}$ が保証されている．V および V' を使って，$r^* = g^{s^*_{(1)}} y^{-h^*_{(1)}} = g^{s^*_{(2)}} y^{-h^*_{(2)}}$ を得る．これは $y = g^{(s^*_{(1)} - s^*_{(2)})/(h^*_{(1)} - h^*_{(2)})}$ を意味する．そこで $\mathcal{B}$ は $x = (s^*_{(1)} - s^*_{(2)})/(h^*_{(1)} - h^*_{(2)}) \bmod q$ を計算し，x を出力して停止する．

以上が $\mathcal{B}$ の動作の手順であり，x は求めるべき離散対数である．言い換えると，$\mathcal{B}$ は与えられた d-SDL 問題を解けるわけである．$\mathcal{B}_{\mathcal{F}}$ が $\delta = 1$ となる組 (δ, V, V') を出力して停止するときは常に，$\mathcal{B}$ は $g^x = y$ となるような離散対数 x の計算に成功する．そのため，

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) = \mathbf{frk}$$

を得る．それから，

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) \geq \frac{\left(\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda)\right)^2}{q_H + q_S} - \frac{2q_S + 1}{q}$$

を得る．最後に，

$$\begin{aligned}&\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda)\\&\leq\left((q_H+q_S)\left(\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda)+\frac{2q_S+1}{q}\right)\right)^{1/2}\end{aligned}$$

を得る．$\mathcal{B}$ の動作時間 $t_{\mathcal{B}}$ は，$\mathcal{B}_{\mathcal{F}}$ の動作時間および x を計算する時間の和である．$\mathcal{B}_{\mathcal{F}}$ は $\mathcal{F}$ を二度走らせ，$\mathcal{F}$ は $\mathcal{A}$ を一度走らせ，また $\mathcal{F}$ は $\mathcal{A}$ からの RKA 署名オラクルクエリおよびハッシュオラクルクエリに答えるため，$\mathcal{B}$ の動作時間は $t_{\mathcal{B}}=2t_{\mathcal{A}}+\mathcal{O}(q_S+q_H)$ を満たす．以上，証明された． □

定理 5 の証明はハッシュオラクルクエリとリスト L_H の要素をそれぞれ，(m_i,r_i) と (m_j,r_j,h_{q_H+j},s_j) に変更することで得られる．また，偽造が生成された時に，$(m^*,r^*,h^*,\cdot)$ の最後の要素は $\perp$ である．というのも，$\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA エクスペリメントで要求されるように，そのメッセージは RKA 署名オラクルにクエリされていないためである．定理 8 の証明においては，偽造 $(m^*,r^*,y,h^*,\cdot)$ の最後の要素は $\perp$ である．というのも，そのメッセージ m^* は，$\phi(sk)=sk$ となる ϕ とともに RKA 署名オラクルにクエリされていないためである．つまり，$y\neq g^{\phi(x)}$ である．

なお，1-SDL 仮定は通常の離散対数仮定と等価である．そのため次の結果が導かれる．

系 9. 改良 Schnorr 署名方式は，$\mathbb{G}$ 上の離散対数仮定のもと，ランダムオラクルモデルにおいて，アフィン関数に関する RKA 安全である．

4.7 改良 DSA およびその RKA 安全性

4.5.2 章で説明したように，オリジナルの DSA は線形関数に関する RKA 安全ではない．本章において，DSA に軽微な修正を加えることで多項式に関する RKA 安全な署名方式を構成できることを示す．この方式を本論文では，改良 DSA と呼ぶことにする．

4.7.1 構成

DSA の一つの変型（[38] において “second variant” と呼ばれているもの）に基づいて，多項式に関する RKA 安全な DSA の変型を構成する．その構成のために施す軽微な修正は以下の通りである．ハッシュ関数はさらなら入力をとるように修正され，その入力は検証鍵の再計算にあたる値である．q を素数とし，p を $p-1 \bmod q = 0$ なる素数とする．また $\mathbb{G} \subseteq \mathbb{Z}_p^*$ を素数位数 q の群とする．$F_{p,q} : \mathbb{G} \to \mathbb{Z}_q$ を $\overline{g} \mapsto \overline{g} \bmod q$ により定まる写像とする．ここで $\overline{g} \in \mathbb{G}$ であり，$\mathbb{G}, q, p$ は群のパラメータである．

改良 DSA は以下のように定義される：

- KG: このアルゴリズムは，入力として 1^λ をとり，署名鍵 sk および検証鍵 vk を以下のように生成する：
  1. 生成元 $g \xleftarrow{\$} \mathbb{G}$ を選ぶ．
  2. $x \xleftarrow{\$} \mathbb{Z}_q^*$ を選び，$y \leftarrow g^x \bmod p$ とする．
  3. ハッシュ関数 $H : \{0,1\}^* \to \mathbb{Z}_q$ を選ぶ．
  4. $sk = x$ および $vk = (g, y, H)$ を出力する．
- Sign: このアルゴリズムは，入力としてメッセージ $m \in \{0,1\}^*$ および署名鍵 sk をとり，署名 σ を以下のように生成する．
  1. $t \xleftarrow{\$} \mathbb{Z}_q^*$ を選び，$r \leftarrow F_{p,q}(g^t \bmod p)$ とする．
  2. $\hat{y} \leftarrow g^x \bmod p$ とする．
  3. $s \leftarrow t^{-1}(H(m \,\|\, r \,\|\, \hat{y}) + x \cdot r) \bmod q$ とする．
  4. $\sigma \leftarrow (r, s)$ を出力する．
- Verify: このアルゴリズムは，入力としてメッセージ m，署名 σ，および検証鍵 vk をとり，署名を以下のように検証する
  1. $r' \leftarrow F_{p,q}(g^{H(m \,\|\, r \,\|\, y)/s} y^{r/s} \bmod p)$ とする．
  2. $r' = r$ ならば 1 を出力し，そうでなければ 0 を出力する．

署名アルゴリズムの第3ステップにおけるハッシュ関数の計算は，入力としてメッセージおよび値 r だけではなく，$\hat{y} = g^x$ もとることに注意が必要である．この計算は DSA の”second variant” [38] の計算とは異なる．

4.7.2 安全性証明

改良 DSA に関して以下の定理を証明する．

定理 10. d を正の整数とし，写像 $F_{p,q}$ は衝突困難とする．$\mathbb{G}$ 上の d-SDL 仮定のもと，4.7.1 章の署名方式はランダムオラクルモデルにおいて $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全である．

より正確には，$F_{p,q}$ は ϵ 衝突困難と仮定する．そのとき，動作時間 $t_{\mathcal{A}}$ で，q_S 回の RKA 署名オラクルクエリをし，H に対して q_H 回のランダムオラクルクエリをする任意の攻撃者 $\mathcal{A}$ に対して，動作時間 $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ のアルゴリズム $\mathcal{B}$ が存在して，以下の式を満たす：

$$\begin{aligned} &\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) \\ &\leq \left((q_H + q_S) \left(\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) + \frac{1}{q} \right) + \right. \\ &\qquad \left. 2 \left(\epsilon + \frac{q_S^2}{q} + q_H \sqrt{\epsilon + \frac{1}{q}} \right) \right)^{1/2}. \end{aligned} \tag{4.2}$$

証明. $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA の意味で改良 DSA を攻撃する $\mathcal{A}$ の利得と d-SDL 問題を解く $\mathcal{B}$ の利得の間の関係を表す不等式（式 (4.2)）は定理 8 の証明と同様の方法で得られる．そこで，ここでは forking アルゴリズム $\mathcal{F}$ およびアルゴリズム $\mathcal{B}$ がどのように構成されるかについて示す．

4.6.2 章で行ったように，一般 forking lemma を用いるために，まず以下を定める．IG を入力生成器とする．これは $(\mathbb{G}, q)$ を決定し，生成元 $g \xleftarrow{\$} \mathbb{G}$ を選び，$x \xleftarrow{\$} \mathbb{Z}_q$ を選び，$y_1 \leftarrow g^x, y_2 \leftarrow g^{x^2}, \ldots, y_d \leftarrow g^{x^d}$ を計算し，$X = (\mathcal{G}, g, y_1, y_2, \ldots, y_d)$ を出力する．集合 Z を $\mathbb{Z}_q$ とし，整数 Q を $q_H + q_S$ とする．$\mathcal{F}$ を，$\mathcal{A}$ の乱数 $\rho_{\mathcal{A}}$ からなる乱数 $\rho_{\mathcal{F}}$ および $\overline{r}_1, \ldots, \overline{r}_{q_H+q_S} \in \mathbb{Z}_q$ を用い，入力として $X = (\mathcal{G}, g, y_1, y_2, \ldots, y_d)$ および $\overline{h}_1, \ldots, \overline{h}_{q_H+q_S} \in \mathbb{Z}_q$ をとり（どれも一様ランダムに選択されると仮定する．なお，ここで乱数 $\overline{r}_i$ は r_i/s_i を表し， 乱数 $\overline{h}_i$ は h_i/s_i を表す），内部で $\mathcal{A}$ を以下のように走らせる：

$\mathcal{F}$ の動作．$\mathcal{F}$ は $vk \leftarrow y_1$ とする．

初期設定． $\mathcal{F}$ は以下のように動作する．

- $L_H \leftarrow \emptyset, M \leftarrow \emptyset$ とする．
- $\mathcal{A}$ を入力 $vk \leftarrow y$ および $\rho_{\mathcal{A}}$ を用いて走らせる．

$\mathcal{A}$ は自身のオラクルに $q_H + q_S$ 回のクエリをする：

ハッシュオラクルクエリ． i 番目（$1 \leq i \leq q_H + q_S$）のクエリがハッシュオラクルに対するもので，(m_i, r_i, y_i) という形であれば，$\mathcal{F}$ は以下のように返答する：

- ある $h \in \mathbb{Z}_q$ に対して $(m_i, r_i, y_i, h, \cdot)$ が L_H に記録されていれば（$\cdot$ は任意の値），$\mathcal{F}$ は h を返す．
- そうでなければ，$\mathcal{F}$ は h_i を以下のように得る：$\overline{r}_i$ および r_i より，$\mathcal{F}$ は $s_i = r_i/\overline{r}_i \bmod q$ を得る．ここで $\mathbb{Z}_q$ における乗法逆元は拡張ユークリッドの互除法により計算ができる．$\overline{h_i}$ および s_i より，$\mathcal{F}$ は $h_i = \overline{h}_i s_i \bmod q$ を得る．それから，$\mathcal{F}$ は $(m_i, r_i, y_i, h_i, \perp)$ をリスト L_H に記録し，h_i を返す．

RKA 署名オラクルクエリ i 番目（$1 \leq i \leq q_H + q_S$）のクエリが RKA 署名オラクルに対するもので，(m_i, ϕ_i) という形であれば，$\mathcal{F}$ は値 $r_i \leftarrow F_{p,q}(g^{\overline{h}_i}(g^{\phi_i(x)})^{\overline{r}_i})$ を計算する．ここで，$\phi_i(x) = a_d \cdot x^d + a_{d-1} \cdot x^{d-1} + \cdots + a_1 \cdot x + a_0$ である．

$\mathcal{F}$ は自身の入力 X を用いて $g^{\phi_i(x)}$ を計算できることに注意が必要である．$\overline{r}_i$ および r_i より，$\mathcal{F}$ は $s_i = r_i/\overline{r}_i \bmod q$ を得る．$\overline{h}_i$ および s_i より，$\mathcal{F}$ は $h_i = \overline{h}_i s_i \bmod q$ を得る．

- もしすでに $(m_i, r_i, g^{\phi_i(x)}, \cdot, \cdot)$ という形のエントリがリスト L_H にあれば，$\mathcal{F}$ は $(0, \perp)$ を出力して停止する．
- そうでなければ，$\mathcal{F}$ は $(m_i, r_i, g^{\phi_i(x)}, h_i, s_i)$ を L_H に記録し，署名 $\sigma_i = (r_i, s_i)$ を $\mathcal{A}$ に返す．

出力. $\mathcal{A}$ がメッセージ m^* および偽造署名 $\sigma^* = (r^*, s^*)$ を出力するとき，$\mathcal{F}$ は検証鍵およびリスト M を用いて検証する．メッセージ m^* に対する偽造署名 $\sigma^* = (r^*, s^*)$ に対して，一般性を失うことなく，$\mathcal{A}$ はハッシュオラクルクエリ (m^*, r^*, y) をしたと仮定する．そのため，$\mathcal{F}$ は常に，ある $h^* \in \mathbb{Z}_q$ に対して $(m^*, r^*, y, h^*, \cdot)$ をリスト L_H 内に見つけることができる．$J \in \{1, \ldots, q_H\}$ をこの過程で見つかった $h^* = h_J$ となるようなインデックスとする．$\mathcal{F}$ は $(J, V \leftarrow (r^*, s^*))$ を出力する．もし検証に通らなければ，$\mathcal{F}$ は $(0, \perp)$ を出力する．

以上が $\mathcal{F}$ の動作の手順である．

$\mathcal{A}$ の RKA 署名オラクルクエリの一つがリスト L_H 内で衝突を起こさない限り（この場合 $\mathcal{F}$ は $(0, \perp)$ を出力し停止する），$\mathcal{F}$ は $\mathcal{A}$ に対して完全なシミュレーションをしている．つまり，以下の 3 つの場合のうちどれかが起きれば，$\mathcal{F}$ は RKA 署名オラクルのシミュレートに失敗する：

1. $F_{p,q}$ の出力が，以前の RKA 署名オラクルクエリの際の，異なる入力で衝突する．つまり $g^{\overline{h}_i}(g^{\phi_i(x)})^{\overline{r}_i}) \neq g^{\overline{h}_j}(g^{\phi_j(x)})^{\overline{r}_j}$ に対して，$F_{p,q}(g^{\overline{h}_i}(g^{\phi_i(x)})^{\overline{r}_i}) = F_{p,q}(g^{\overline{h}_j}(g^{\phi_j(x)})^{\overline{r}_j})$ となる．ここで，$1 \leq j < i \leq q_H + q_S$

2. $F_{p,q}$ に渡される入力が，以前の RKA 署名オラクルクエリで作られた入力と一致する．つまり $g^{\overline{h}_i}(g^{\phi_i(x)})^{\overline{r}_i}) = g^{\overline{h}_j}(g^{\phi_j(x)})^{\overline{r}_j}$．ここで $1 \leq j < i \leq q_H + q_S$.

3. 組 $(m_i, r_i, g^{\phi_i(x)}, \ldots)$ が，$\mathcal{A}$ によってなされた以前のハッシュオラクルクエリにより作られた L_H 内の組と一致する．

1 つ目のケースは，写像 $F_{p,q}$ の ϵ 衝突困難性が破れることを意味し，このケースが起きる確率は高々 ϵ である．2 つ目のケースは，各 RKA 署名オラクルクエリごとに高々確率 q_S/q であり，それゆえに q_S^2/q で上から抑えられる．3 つ目のケースは，高々確率 $q_S q_H \sqrt{\epsilon + 1/q}$ で起こる．というのも，任意の固定値 r に対して $F_{p,q}(\xi) = r$ となる確率は $q_H \sqrt{\epsilon + 1/q}$ であるためである．ここで，$F_{p,q}$ は ϵ 衝突困難であり，$F_{p,q}$ への入力 ξ は一様ランダムである．上記の理由から，F が $(0, \perp)$ 以外を出力して停止する確率は少なくとも

$$\mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) - \left(\epsilon + \frac{q_S^2}{q} + q_H \sqrt{\epsilon + \frac{1}{q}}\right)$$

である．

$\mathcal{B}$の動作. $\mathcal{B}$は入力としてd-SDL問題$(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$をとり，$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA安全性ゲームにおいて攻撃者$\mathcal{A}$のチャレンジャをシミュレートする．$X = (\mathcal{G}, (g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}))$とする．アルゴリズム$\mathcal{B}$は前述したように，$\mathcal{F}$に関するforking procedure$\mathcal{B}_{\mathcal{F}}$を実行する．$(\delta, V, V')$を$\mathcal{B}_{\mathcal{F}}$の出力とする．$\delta = 0$ならば，$\mathcal{B}$は諦め動作終了する．そうでなければ，$V = (r^*, s^*_{(1)})$および$V' = (r^*, s^*_{(2)})$とし，ハッシュ値を計算するのに用いた対応する値を$\overline{h^*}_{(1)}$および$\overline{h^*}_{(2)}$と書く．さらに，対応するハッシュ値を$h^*_{(1)}$および$h^*_{(2)}$と書く．そしてこれらは，$h^*_{(1)} = \overline{h^*}_{(1)} s^*_{(1)} \bmod q$および$h^*_{(2)} = \overline{h^*}_{(2)} s^*_{(2)} \bmod q$である．ここで，$\overline{h^*}_{(1)} \neq \overline{h^*}_{(2)}$ということが一般forking lemmaにより保証される．そしてこれは

$$\frac{h^*_{(1)}}{s^*_{(1)}} \neq \frac{h^*_{(2)}}{s^*_{(2)}} \tag{4.3}$$

であることを意味する．VおよびV'を用いて，$r' = F_{p,q}(g^{h^*_{(1)}/s^*_{(1)}} y^{r^*/s^*_{(1)}} \bmod p) = F_{p,q}(g^{h^*_{(2)}/s^*_{(2)}} y^{r^*/s^*_{(2)}} \bmod p) = r^*$を得る．これは，少なくとも確率$1 - \epsilon$で

$$g^{h^*_{(1)}/s^*_{(1)}} y^{r^*/s^*_{(1)}} \bmod p = g^{h^*_{(2)}/s^*_{(2)}} y^{r^*/s^*_{(2)}} \bmod p \tag{4.4}$$

であることを意味する．というのも，関数$F_{p,q}$は高々確率ϵで衝突をするためである．なお，$s^*_{(2)} \neq s^*_{(1)}$を得る．というのも，そうでなければ式 (4.4) が，式 (4.3) および$\overline{h^*}_{(1)} \neq \overline{h^*}_{(2)}$であること矛盾するためである．このとき，

$$y = g^{(s^*_{(1)} h^*_{(2)} - s^*_{(2)} h^*_{(1)})/r^*(s^*_{(2)} - s^*_{(1)})} \bmod p$$

を得る．よって，$\mathcal{B}$は

$$x = \frac{s^*_{(1)} h^*_{(2)} - s^*_{(2)} h^*_{(1)}}{r^*(s^*_{(2)} - s^*_{(1)})} \bmod q$$

を得る．このxは求めるべき離散対数である．つまり，$\mathcal{B}$は与えられたd-SDL問題を解く．$\mathcal{B}$の動作時間$t_{\mathcal{B}}$は，$\mathcal{A}$を二度走らせる時間とxを計算する時間，およびRKA署名オラクルクエリとハッシュオラクルクエリに答える時間の和である．それゆえに，$t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$である． □

ここで，定理6および定理10の証明の違いについてハイライトする．定理6の証明はハッシュオラクルクエリを(m_i, r_i)に変更し，リストL_Hの要素を(m_j, r_j, h_j, s_j)

に変更することで得られる．さらに，偽造が作られ，アルゴリズム $\mathcal{F}$ が正当性をチェックする時に，$(m^*, r^*, h^*, \cdot)$ の最後の要素は $\perp$ である．というのも，$\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA 安全性の定義では，そのメッセージが RKA 署名オラクルに聞かれていてはいけないからである．また，定理 10 の証明では，$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全性の定義は $(m^*, r^*, y, h^*, \cdot)$ の最後の要素は $\perp$ であると保証する．というのも，そのメッセージ m^* は RKA 署名オラクルに $\phi(sk) = sk$ となる ϕ とともにクエリされていないためである．そのため $y \neq g^{\phi(x)}$ である．1-SDL 仮定は通常の離散対数仮定と等価である．そのため次の結果が言える．

系 11. $\mathbb{G}$ 上の離散対数仮定が成り立ち，関数 $F_{p,q}$ が衝突困難であれば，改良 DSA はランダムオラクルモデルにおいてアフィン関数に関する RKA 安全である．

4.8 改良 ElGamal 署名方式とその安全性証明

4.5.3 章で説明したように，オリジナルの ElGamal 署名方式は線形関数に関する RKA 安全ではない．本章において，ElGamal 署名方式に軽微な修正を加えることで多項式に関する RKA 安全な署名方式を構成できることを示す．この方式を本論文では，改良 ElGamal 署名方式と呼ぶことにする．

4.8.1 構成

[36] で提案された修正 ElGamal 署名方式に基づいて，多項式に関する RKA 安全な ElGamal 署名方式の変型を構成する．本論文で提案する，ElGamal 署名方式に対する改良は以下の通りである．ハッシュ関数はさらなる入力をとるように修正され，その入力は検証鍵の再計算に対応する．

p を $p > 2^\lambda$ かつ $p - 1 = qR$ となる素数とする．ここで q は指数関数的に大きな素数で，R はセキュリティパラメータ λ に関する多項式サイズの数である．

改良 ElGamal 署名方式を以下のように定義する：

- KG: このアルゴリズムは入力として 1^λ をとり，署名鍵 sk および検証鍵 vk を以下のように生成する．

1. 生成元 $g \xleftarrow{\$} \mathbb{Z}_p^*$ を選ぶ.
2. $x \xleftarrow{\$} \mathbb{Z}_{p-1}$ に対して，$y \leftarrow g^x \bmod p$ を計算する.
3. $H : \{0,1\}^* \to \mathbb{Z}_{p-1}$ を選ぶ.
4. $sk = x, vk = (g, y, H)$ を出力する.

- Sign: このアルゴリズムは入力としてメッセージ $m \in \{0,1\}^*$ および署名鍵 sk をとり，署名 σ を以下のように生成する.

1. $x \leftarrow sk$ とする.
2. $t \xleftarrow{\$} Z_{p-1}^*$ に対して，$r \leftarrow g^t \bmod p$ を計算する.
3. $\hat{y} \leftarrow g^x \bmod p$ を計算する.
4. $s \leftarrow \dfrac{H(m, r, \hat{y}) - xr}{t} \bmod (p-1)$ とする.
5. $\sigma = (r, s)$ を出力する.

- Verify: このアルゴリズムは入力としてメッセージ m，署名 σ，および検証鍵 vk をとり，署名を以下のように検証する.

1. $\sigma = (r, s), vk = (y, H)$ と要素に分ける.
2. $h \leftarrow H(m, r, y)$ を計算する.
3. $g^h = y^r r^s \pmod p$ ならば 1 を出力し，そうでなければ 0 を出力する.

4.8.2 安全性証明

改良 ElGamal 署名方式に関して以下の定理を証明する.

定理 12. d を正の整数とする．$\mathbb{Z}_p^*$ 上の d-SDL 仮定のもと，4.8.1 章の署名方式はランダムオラクルモデルにおいて $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全である.

より正確には，動作時間 $t_{\mathcal{A}}$ で q_S 回の RKA 署名オラクルクエリをし，H に対して q_H 回のランダムオラクルクエリをする任意の攻撃者 $\mathcal{A}$ に対して，動作時間 $t_{\mathcal{B}} \leq 4t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ のアルゴリズム $\mathcal{B}$ が存在して，以下の式を満たす：

$$\begin{aligned}
&\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) \\
&\leq \left(R(q_H+q_S)^2 \mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) \left(1-\frac{1}{q}\right)^{-3} \right. \\
&\qquad \left. + \frac{2(q_S+1)(q_S+q_H)}{qR} + \frac{4q_S}{q} \right)^{1/4}
\end{aligned} \tag{4.5}$$

証明. forking アルゴリズム $\mathcal{F}$ および d-SDL 問題を解くアルゴリズム $\mathcal{B}$ がどのように構成されるか示す．まず，forking lemma を用いるために，以下を定める．IG を入力生成器とする．これは p を決定し，生成元 $g \xleftarrow{\$} \mathbb{Z}_p^*$ を選び，$x \xleftarrow{\$} \mathbb{Z}_{p-1}$ を選び，$y_1 \leftarrow g^x, y_2 \leftarrow g^{x^2}, \ldots, y_d \leftarrow g^{x^d}$ を計算し，$X = (p, g, y_1, y_2, \ldots, y_d)$ を出力する．集合 Z を $\mathbb{Z}_{p-1}$ とし，整数 Q を q_H+q_S とする．$\mathcal{A}$ を，$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全性の意味で改良 ElGamal 署名方式を攻撃する攻撃者とする．forking アルゴリズム $\mathcal{F}$ は，$\mathcal{A}$ の乱数 $\rho_{\mathcal{A}}$ を含む乱数 $\rho_{\mathcal{F}}$ および $u_1, \ldots, u_{q_H+q_S} \in \mathbb{Z}_q, v_1, \ldots, v_{q_H+q_S} \in \mathbb{Z}_q^*, w_1, \ldots, w_{q_H+q_S} \in \mathbb{Z}_R^*$ を用い，入力として $X = (\mathcal{G}, y_1, y_2, \ldots, y_d)$ および $(h'_1, \ldots, h'_{q_H+q_S}) \in \mathbb{Z}_{p-1}^{q_H+q_S}$ をとり（どちらも一様ランダムに選択されると仮定する），内部で $\mathcal{A}$ を以下のように走らせる：

$\mathcal{F}$ の動作．$\mathcal{F}$ は $vk \leftarrow (g, y_1, H)$ とする．

初期化． $\mathcal{F}$ は以下のように動作する．

- $L_H \leftarrow \emptyset, M \leftarrow \emptyset$ とする．
- $\mathcal{A}$ を入力 vk および乱数 $\rho_{\mathcal{A}}$ を用いて走らせる．

$\mathcal{A}$ は自身のオラクルに $q_H + q_S$ 回のクエリをする．

ハッシュオラクルクエリ i 番目（$1 \leq i \leq q_H + q_S$）のクエリがハッシュオラクルに対するもので，(m_i, r_i, y_i) という形であれば，$\mathcal{F}$ は以下のように返答する：

- ある $h \in \mathbb{Z}_{p-1}$ に対して $(m_i, r_i, y_i, h, \cdot)$ が L_H に記録されていれば（$\cdot$ は任意の値），$\mathcal{F}$ は h を返す．
- そうでなければ，$\mathcal{F}$ は $(m_i, r_i, y_i, h'_i, \perp)$ をリスト L_H に記録し，h'_i を返す．

RKA 署名オラクルクエリ. i 番目（$1 \le i \le q_H + q_S$）のクエリが RKA 署名オラクルに対するもので，(m_i, ϕ_i) という形で $\phi_i(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$ であれば，$\mathcal{F}$ は乱数 t_i に対して $r_i = g^{t_i}$ かつ $s_i = (H(m_i, r_i, g^{\phi_i(x)}) - \phi_i(x) r_i)/t_i \bmod (p-1)$ である署名 (r_i, s_i) を以下のように出力する：$\mathcal{F}$ は $r_i \leftarrow g^{Ru_i} \cdot ((g^{x^d})^{a_d} \cdot (g^{x^{d-1}})^{a_{d-1}} \cdot \cdots \cdot (g^x)^{a_1} \cdot g^{a_0})^{Rv_i} \cdot g^{qw_i} \pmod p$ とする．なお，$\mathcal{F}$ はこれを自身の入力 $y_1 = g^x, y_2 = g^{x^2}, \ldots, y_d = g^{x^d}$ を用いて計算できる．RKA 署名オラクルをシミュレートするために，$\mathcal{F}$ は $\tilde{s}_i = -r_i (Rv_i)^{-1} \bmod q$ and $\tilde{h}_i = -u_i r_i v_i^{-1} \bmod q$ とする．これは

$$\tilde{h}_i = \phi_i(x) \cdot r_i + R \cdot (u_i + \phi_i(x) \cdot v_i) \cdot \tilde{s}_i \bmod q$$

を満たす．次に，r_i が生成元かどうかによって，どのように署名を計算するかを説明する．

- r_i が $\mathbb{Z}_p^*$ の生成元であるならば：r_i は唯一の K を用いて $r_i = g^K \bmod p$ と書け，これは q と互いに素である．上記のシミュレーションより，u_i および v_i について 2 つの等式が得られる：

$$r_i u_i + \tilde{h}_i v_i = 0 \pmod q \tag{4.6}$$

$$u_i + \phi_i(x) v_i = K R^{-1} \pmod q. \tag{4.7}$$

  - $\tilde{h}_i \neq \phi_i(x) r_i \bmod q$ ならば，$\mathcal{F}$ は唯一の解 (u_i, v_i) を得る．このとき，$\mathcal{F}$ は $\tilde{h}_i$ および $h_i' = h_i \bmod R$ を用いて新たな h_i を計算する．また，$\tilde{s}_i$ を用いて，$g^{h_i} = y^{r_i} r_i^{s_i}$ を満たす $s_i \bmod (p-1)$ を計算する．そして (r_i, s_i) を出力する．
  - $\tilde{h}_i = \phi_i(x) r_i \bmod q$ ならば，$\mathcal{F}$ は $(0, \perp)$ を出力し停止する．

- そうでなければ，r_i は $\mathbb{Z}_p^*$ の生成元ではない：$\mathcal{F}$ は $\tilde{h}_i \in \mathbb{Z}_q$ および入力 $h_i'' = h_i' \bmod R$ を用い，$h_i \in \mathbb{Z}_{p-1}$ を得る．それから，$\mathcal{F}$ は $g^{h_i} = y^{r_i} r_i^{s_i} \bmod p$ となる s_i を全数探索する．これは $h_i \in \mathbb{Z}_{p-1}$ および $\tilde{s}_i \in \mathbb{Z}_q$ を知っている $\mathcal{F}$ が計算するのは高々 R 回の試行で可能である．より詳しく言うと，$h_i \in \mathbb{Z}_{p-1}$ および $\tilde{s}_i \in \mathbb{Z}_q$ が与えられて，$\mathcal{F}$ は h_i および $s_i = \tilde{s}_i + q \cdot \eta$ $(0 \le \eta \le R-1)$ が $g^{h_i} = y^{r_i} r_i^{s_i} \bmod p$ を満たすかどうかをチェックすればよい．$\mathcal{F}$ は (r_i, s_i) を出力する．

値 h_i および s_i に対して，$h_i'' = h_i \bmod R, \tilde{h}_i = h_i \bmod q$，および $\tilde{s}_i = s_i \bmod q$ であることに注意が必要である．$\mathcal{F}$ は，が署名を計算した後は次のように動作する：

- すでに $(m_i, r_i, g^{\phi_i(x)}, \cdot, \cdot)$ という形のエントリがリスト L_H にあれば，$\mathcal{F}$ は $(0, \perp)$ を出力して停止する．
- そうでなければ，$\mathcal{F}$ は $(m_i, r_i, g^{\phi_i(x)}, h_i, s_i)$ を L_H に記録し，署名 $\sigma_i = (r_i, s_i)$ を $\mathcal{A}$ に返す．$\phi_i(sk) = sk$ ならば ($\mathcal{F}$ はこれを以下のように間接的にチェックできる．ϕ_i が多項式である限り，d-SDL 問題として与えられた $y_1, \ldots, y_d$ および RKD 関数 ϕ_i を用いて $g^{\phi_i(sk)} = y_1$ をチェックする），$\mathcal{F}$ は m_i をリスト M に記録する．

出力． $\mathcal{A}$ がメッセージ m^* および偽造署名 $\sigma^* = (r^*, s^*)$ を出力するとき，$\mathcal{F}$ はそれw検証鍵およびリスト M を用いて検証する．メッセージ m^* に対する偽造署名 $\sigma^* = (r^*, s^*)$ に対して，一般性を失うことなく，$\mathcal{A}$ はハッシュクエリ (m^*, r^*, y) をしたと仮定する．そのため，$\mathcal{F}$ は必ずリスト L_H の中に，ある $h^* \in \mathbb{Z}_{p-1}^*$ に対して $(m^*, r^*, y, h^*, \cdot)$ を見つける．$J \in \{1, \ldots, q_H\}$ を，この過程で見つかる $h^* = h_J$ を満たすインデックスとする．$\mathcal{F}$ は $(J, V \leftarrow (r^*, s^*))$ を出力する．もし検証に通らなければ，$\mathcal{F}$ は $(0, \perp)$ を出力する．

以上が $\mathcal{F}$ の動作の手順である．$\mathcal{F}$ が $\mathcal{A}$ の RKA 署名オラクルクエリに答える過程で $(0, \perp)$ を出力して停止しない限り，$\mathcal{F}$ は $\mathcal{A}$ に対して安全性モデルを完全にシミュレートしていることがわかる．1 つの RKA 署名オラクルクエリに対して $\tilde{h}_i = \phi_i(x) r_i \bmod q$ となる（つまり $\mathcal{F}$ が $(0, \perp)$ を出力して停止する）確率は，高々$1/q$ である．それゆえ，リスト L_H 内で衝突が起きる確率は高々$(q_S + q_H)/qR$ である．ゆえに $\mathcal{F}$ が $\mathcal{A}$ の RKA 署名オラクルに答えるのに失敗する確率は，高々

$$\frac{q_S(q_S + q_H)}{qR} + \frac{q_S}{q}$$

である．$\mathcal{F}$ が $\mathcal{A}$ の RKA 署名オラクルクエリに答えるのに失敗する確率を考えることで，以下の不等式を得る：

$$\mathbf{acc} \geq \mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) - \frac{q_S(q_S + q_H)}{qR} - \frac{q_S}{q}.$$

さらに，一般 forking lemma により

$$\mathbf{frk} \geq \mathbf{acc} \cdot \left(\frac{\mathbf{acc}}{q_H + q_S} - \frac{1}{qR} \right)$$

を得る．

$\mathcal{B}$ の動作手順 $\mathcal{B}$ は入力として d-SDL 問題 $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$ をとり，$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA 安全性モデルでの攻撃者 $\mathcal{A}$ のオラクルをシミュレートする．$\mathcal{B}$ は $X = (\mathcal{G}, (g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}))$ と定める．入力 X により，アルゴリズム $\mathcal{B}$ は上に示したように $\mathcal{F}$ に対応する forking procedure$\mathcal{B}_{\mathcal{F}}$ を実行する．(δ, V, V') を $\mathcal{B}_{\mathcal{F}}$ の出力とする．$\delta = 0$ ならば，$\mathcal{B}$ は停止する．そうでなければ，$V = (r^*, s^*_{(1)})$ および $V' = (r^*, s^*_{(2)})$ とし，ハッシュ値を計算するのに用いた対応する値を $(h^*)'_{(1)}$ および $(h^*)'_{(2)}$ と書く．さらに，対応するハッシュ値を $h^*_{(1)}$ および $h^*_{(2)}$ と書く．ここでは，$(h^*)'_{(1)} \neq (h^*)'_{(2)}$ が保証されている．そのため，構成より $h^*_{(1)} \neq h^*_{(2)}$ となる．V および V' を用いて，$g^{h^*_{(1)}} = (r^*)^{s^*_{(1)}} y^{r^*}$ および $g^{h^*_{(2)}} = (r^*)^{s^*_{(2)}} y^{r^*}$ を得る．これは，$g^t = r^* \pmod p$ および $g^x = y \pmod p$ を満たすある $t \in Z^*_{p-1}$ および $x \in \mathbb{Z}_{p-1}$ に対して

$$h^*_{(1)} s^*_{(2)} - h^*_{(2)} s^*_{(1)} = x r^* (s^*_{(2)} - s^*_{(1)}) \pmod{p-1} \tag{4.8}$$

および

$$h^*_{(1)} - h^*_{(2)} = t(s^*_{(1)} - s^*_{(2)}) \pmod{p-1} \tag{4.9}$$

を意味する．値 $h^*_{(1)}$ および $h^*_{(2)}$ は，ランダムオラクルの出力であり，それゆえに無視可能ではない $1 - (R/qR) = 1 - (1/q)$ という確率（高い確率）で $h^*_{(1)} - h^*_{(2)}$ は q と互いに素である．$h^*_{(1)} - h^*_{(2)}$ が q と互いに素であるとき，式 (4.9) により，値 $s^*_{(1)} - s^*_{(2)}$ もまた q と互いに素である．

- r^* が q と互いに素ならば，式 (4.8) より $\mathcal{B}$ は

  $$x = \frac{h^*_{(1)} s^*_{(2)} - h^*_{(2)} s^*_{(1)}}{r^* (s^*_{(2)} - s^*_{(1)})} \pmod q$$

  を得る．$0 \leq \eta \leq R - 1$ に対する R 個の値

  $$\left(\frac{h^*_{(1)} s^*_{(2)} - h^*_{(2)} s^*_{(1)}}{r^* (s^*_{(2)} - s^*_{(1)})} \bmod q \right) + q \cdot \eta \in \mathbb{Z}_{p-1}$$

  の中から，$\mathcal{B}$ は $g^x = y \bmod p$ となる $x \in \mathbb{Z}_{p-1}$ を見つけることができる．

- そうでなければ，$b \in \mathbb{Z}_R$ に対して $r^* = bq$ となる．よって，

$$t = \frac{h^*_{(1)} - h^*_{(2)}}{s^*_{(1)} - s^*_{(2)}} \pmod{q}$$

を得る．$\mathcal{B}$ は $g^t = bq \bmod p$ となる $t \in \mathbb{Z}^*_{p-1}$ を，$0 \leq \eta \leq R-1$ に対して

$$\left(\frac{h^*_{(1)} - h^*_{(2)}}{s^*_{(1)} - s^*_{(2)}} \bmod q \right) + q \cdot \eta \in \mathbb{Z}_{p-1}$$

の中から見つけることが出来る．

この場合，$\mathcal{B}$ は新たな生成元 $g' = y_1 g^\ell$ を用いて新たな d-SDL 問題 $X' = (\mathcal{G}', y'_1, y'_2, \ldots, y'_d) = ((p, g'), g'^{x'}, g'^{x'^2}, \ldots, g'^{x'^d})$ を生成する．ここで，ℓ および x' は $\mathbb{Z}^*_{p-1}$ からランダムに選ばれる．それから，$\mathcal{B}$ は $\mathcal{B}_{\mathcal{F}}$ を入力 X' をして再度走らせる．X' を入力として用いた $\mathcal{B}_{\mathcal{F}}$ の 2 回目の実行により，ハッシュ値の引き算が q と互いに素ならば（確率は $1 - 1/q$），X での 1 回目の $\mathcal{B}_{\mathcal{F}}$ の実行のときと同様に，$\mathcal{B}$ は x' または (t', b') を得る．$\mathcal{B}$ が (t', b') 得る時（対応する乱数を r' とする），d-SDL 問題は以下のように解ける．値 b' は少なくとも確率 $1/R$ で b に等しい．というのも，r' が q と互いに素でないとき，$r' = b'q$ は $\{0, q, 2q, \ldots, (R-1)q\}$ に含まれるためである．このとき，確率 $1/R$ で $(g^\ell y_1)^{t'} = bq = g^t \bmod p$ となる．それゆえに，$\mathcal{B}$ は $xt' = t - \ell t' \pmod{p-1}$ を得る．それに従って，t' は q と互いに素なので

$$x = \frac{t - \ell t'}{t'} \pmod{q},$$

を計算できる．確率 $1 - 1/\varphi(p-1)$ で $t - \ell t' \bmod (p-1) \neq 0$ となることに注意．ここで，φ はオイラー関数である．$\varphi(p-1) = \varphi(qR) = \varphi(q)\varphi(R) = (q-1)\varphi(R) > q$ であるため，$1 - 1/\varphi(p-1) \geq 1 - 1/q$ である．$\mathcal{B}$ は $0 \leq \eta \leq R-1$ に対して $g^x = y \bmod p$ を満たす $(x \bmod q) + q \cdot \eta$ を探す．これは高々 R 回の試行で見つかる．言い換えると，$\mathcal{B}$ は与えられた d-SDL 問題を解く．

利得を評価するために，$\mathcal{B}$ 安全性ゲームでの以下のイベントを考える：

$\mathbf{E}_1$: 1 回目の $\mathcal{B}_{\mathcal{F}}$ の forking が成功する，つまり，$\mathcal{B}_{\mathcal{F}}$ は $(1, V_1, V_2)$ を出力する．

$\mathbf{E}_{2\text{-}1}$: 1 回目の $\mathcal{B}_{\mathcal{F}}$ の実行で，r^* が q と互いに素である（このイベントはイベント $\mathbf{E}_1$ が起こったことを仮定する）．

$\mathbf{E}_{2\text{-}2}$: 1 回目の $\mathcal{B}_{\mathcal{F}}$ の実行で，r^* が q と互いに素ではない（このイベントはイベント $\mathbf{E}_1$ が起こったことを仮定する）．

$\mathbf{E}_3$: 1 回目の $\mathcal{B}_{\mathcal{F}}$ の実行で，$h^*_{(1)} - h^*_{(2)}$ が q と互いに素である（このイベントはイベント $\mathbf{E}_1$ が起こったことを仮定する）．

$\mathbf{E}_4$: 2 回目の $\mathcal{B}_{\mathcal{F}}$ の forking が成功する（このイベントは $\mathbf{E}_1$ と $\mathbf{E}_{2\text{-}2}$ が起こったことを仮定する）．

$\mathbf{E}_{5\text{-}1}$: 2 回目の $\mathcal{B}_{\mathcal{F}}$ の実行で，r^* が q と互いに素である（このイベントは $\mathbf{E}_4$ が起こったことを仮定する）．

$\mathbf{E}_{5\text{-}2}$: 2 回目の $\mathcal{B}_{\mathcal{F}}$ の実行で，r^* が q と互いに素ではない（このイベントは $\mathbf{E}_4$ が起こったことを仮定する）．

$\mathbf{E}_6$: 2 回目の $\mathcal{B}_{\mathcal{F}}$ の実行で，$h^*_{(1)} - h^*_{(2)}$ が q と互いに素である（このイベントは $\mathbf{E}_1$ と $\mathbf{E}_{2\text{-}2}$ が起こったことを仮定する）．

$\mathbf{E}_7$: b が 1 回目の実行からで，b' が 2 回目の実行からである値とすると，$b = b'$ である（このイベントは $\mathbf{E}_1$ と $\mathbf{E}_{2-2}$ と $\mathbf{E}_4$ と $\mathbf{E}_{5-2}$ が起こったことを仮定することに注意）．

$\mathbf{E}_8$: $t - \ell t' \bmod (p-1) \neq 0$（このイベントは $\mathbf{E}_7$ が起こったことを仮定する）．

今，$\mathcal{B}$ の利得は $\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) = \Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}1} \wedge \mathbf{E}_3] + \Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6 \wedge \mathbf{E}_7 \wedge \mathbf{E}_8]$ と表現できる．

まず，$\mathcal{B}$ の利得の第一項 $\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}1} \wedge \mathbf{E}_3]$ に着目する．$h^*_{(1)} - h^*_{(2)}$ は攻撃者 $\mathcal{A}$ には完全に隠されているため，$\mathbf{E}_3$ は “$\mathbf{E}_1$ かつ $\mathbf{E}_{2\text{-}1}$” からは独立していると仮定できる．ゆえに，$\mathcal{B}$ の利得の第一項に対して，$\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}1} \wedge \mathbf{E}_3] = \Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}1}] \cdot \Pr[\mathbf{E}_3]$ を得る．すでに議論されたように，$h^*_{(1)}$ および $h^*_{(2)}$ は $\mathbb{Z}_{p-1}$ からランダムに選ばれるため，$\Pr[\mathbf{E}_3] \geq 1 - 1/q$ である．続いて，$\mathcal{B}$ の利得の第二項に着目する．条件付き確

率を考えることで，第二項は

$$\begin{aligned}
&\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6 \wedge \mathbf{E}_7 \wedge \mathbf{E}_8] \\
&= \Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6] \\
&\times \Pr[\mathbf{E}_7 \mid \mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6] \\
&\times \Pr[\mathbf{E}_8 \mid \mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6 \wedge \mathbf{E}_7]
\end{aligned}$$

と計算できる．今，$\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2-1}]$ を P と書くことにする．すると，定義より $\Pr[\mathbf{E}_1] = \mathbf{frk}$ であるため，$\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2}] = \mathbf{frk} - P$ である．ここで $\mathbf{frk}$ は $\mathcal{B}_{\mathcal{F}}$ が forking に成功する確率である．2 回の実行における $\mathcal{B}_{\mathcal{F}}$ への入力は真に独立で分布は同一であるため，1 回目の実行に係るイベントおよび 2 回目の実行に係るイベントは独立であり，同じ確率で起こる．それゆえに，

$$\begin{aligned}
&\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6] \\
&= \Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3]^2 \\
&= (\Pr[\mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2}] \cdot \Pr[\mathbf{E}_3])^2 \\
&= (\mathbf{frk} - P)^2 \cdot (1 - 1/q)^2
\end{aligned}$$

を得る．先に述べたように，$\Pr[\mathbf{E}_7 \mid \mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6] \geq 1/R$ であり，$\Pr[\mathbf{E}_8 \mid \mathbf{E}_1 \wedge \mathbf{E}_{2\text{-}2} \wedge \mathbf{E}_3 \wedge \mathbf{E}_4 \wedge \mathbf{E}_{5\text{-}2} \wedge \mathbf{E}_6 \wedge \mathbf{E}_7] \geq 1/(\varphi(p-1)) \geq 1 - 1/q$ である．よって，$\mathcal{B}$ の利得の第二項は，$(\mathbf{frk} - P)^2 \cdot (1 - 1/q)^3 (1/R)$ により下から抑えられる．

最終的に，$\mathcal{B}$ の利得は $P \cdot (1 - 1/q) + (\mathbf{frk} - P)^2 \cdot (1 - 1/q)^3 (1/R)$ により下から抑えられる．そしてこれは $P = 0$ のとき最小となる．最小となるとき，

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) \geq \mathbf{frk}^2 \left(1 - \frac{1}{q}\right)^3 \frac{1}{R}$$

を得る．そのため，最終的には，

$$\begin{aligned}
&\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) \\
&\leq \left(R(q_H + q_S)^2 \mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) \left(1 - \frac{1}{q}\right)^{-3} \right. \\
&\qquad \left. + \frac{2(q_S + 1)(q_S + q_H)}{qR} + \frac{4q_S}{q} \right)^{1/4}
\end{aligned}$$

を得る．$\mathcal{B}$ の動作時間 $t_{\mathcal{B}}$ は，最大でも $\mathcal{A}$ を 4 回走らせる時間，x を計算する時間，RKA 署名オラクルクエリおよびハッシュオラクルクエリに答える時間の和である．よって，$t_{\mathcal{B}} \leq 4t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ である． □

1-SDL 仮定が通常の離散対数仮定と等価であるという事実により，以下の結果が導かれる．

系 13. $\mathbb{Z}_p^*$ 上の離散対数仮定が成り立つならば，改良 ElGamal 署名方式はランダムオラクルモデルにおいて，アフィン関数に関する RKA 安全である．

第5章　結論

5.1　まとめ

スマートカードなど，暗号技術を搭載した小型のデバイスの普及は生活を便利にする一方，電磁波などによる物理的な攻撃技術の現在の発達によって，小型デバイスから個人情報が漏洩する危険性が増している．これまでの暗号研究では，ハードウェア内部に保存された秘密鍵は安全であると考える安全性解析が主だったが，タンパリングやフォルトインジェクション攻撃といった秘密鍵への改ざんを行うような，より能動的な攻撃をも考慮に入れる必要性が出ている．そこで，本論文ではそうした攻撃を理論的に扱う枠組みである RKA を考慮に入れ，既存の暗号技術の安全性についてさらなる調査を行った．

本論文では 3 章において，RKA-CKS-light，RKA-CKS，RKA-CKS-heavy，および RKA-m-CKS-heavy と書く NIKE の RKA 安全性を提案し，それらの間の関係を示した．特に，RKA-CKS，RKA-CKS-heavy，および RKA-m-CKS-heavy は等価であり，RKA-CKS-light はその他から分離があることを示した．なお，この分離は，RKD 関数が線形関数の時に示した．この分離が生じる理由としては，RKA-CKS-light 安全性とその他の安全性との以下のような違いが挙げられる：RKA-CKS-light の攻撃者は，2 人の honest ユーザの KG からの出力をただ 1 つだけ得ることが許されている．一方，RKA-m-CKS-heavy，RKA-CKS-heavy，そして RKA-CKS の攻撃者は，2 人の honest ユーザの KG からの出力を複数得ることが許されている．この分離の結果が示唆するのは，RKA-CKS-light 安全性を示すだけでは不十分であるということである．Freire らが示したように，RKA を考慮に入れない安全性定義においては，4 つの安全性定義が等価であったため，CKS-light 安全性を示しさえすれば十分であった．しかし，RKA を考慮に入れると，RKA-CKS-light 安全性が示されたとしても，その他の安全性定義の意味で安全であることは必ずしも保証されない．現実世界の攻撃者がタンパリングやフォルトインジェクション攻撃といった能動的な

攻撃を行いつつある状況においては，より高い安全性を担保する RKA-CKS-heavy 安全（RKA-CKS 安全性および RKA-m-CKS-heavy 安全性も満たす）な NIKE 方式が望ましいことが示された．

そこで本論文では，$\mathrm{NIKE}_{\mathrm{fac}}$ 方式の RKA 安全性を解析し，符号付剰余群上の DSDH 仮定のもとランダムオラクルモデルにおいて，その方式が Φ^{lin}-RKA-CKS-heavy 安全であることを証明した．そしてこれは，その方式が同様の仮定のもとランダムオラクルモデルにおいて Φ^{lin}-RKA-CKS-light 安全，Φ^{lin}-RKA-CKS 安全，Φ^{lin}-RKA-m-CKS-heavy 安全であることも意味する．

4 章において，Schnorr 署名方式，DSA，および ElGamal 署名方式の RKA 安全性を解析した．特に，Schnorr 署名方式，[38] における第二の DSA の変型，および [36] における修正 ElGamal 署名方式は多項式に関する弱 RKA 安全（$\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA）であるが，Schnorr 署名方式，オリジナルの DSA，およびオリジナルの ElGamal 署名方式（ElGamal 署名方式はすでに通常の安全性の意味でも安全ではないが）は線形関数に関する RKA 安全性（Φ^{lin}-EUF-CM-RKA）という，RKA 安全性の中でも比較的弱い安全性でさえ満たさないことを示した．しかしながら，各署名方式にシンプルな修正を加えた，改良 Schnorr 署名方式，改良 DSA，改良 ElGamal 署名方式が多項式に関する RKA 安全（$\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA）であることをランダムオラクルモデルにおいて証明した．多項式に関する RKA 安全性は，d-SDL 仮定のもとで証明される．興味深いことに，$d = 1$ の場合を考えることで，提案する改良署名方式は，通常の離散対数仮定のもとランダムオラクルモデルにおいてアフィン関数に関する RKA 安全であることがわかる．さらに，改良のために Schnorr 署名方式，第二の DSA の変型，修正 ElGamal 署名方式に加えたシンプルな修正は，元々の方式の公開鍵や秘密鍵に変更を加えることはなく，ただ署名アルゴリズムで 1 つ累乗計算を増やすだけで，その他の計算コストや署名長が増えることはない．この結果により，現在でも広く利用されている署名方式の署名鍵や，すでに配布された検証鍵を再生成するという労力をかけずに，署名者および検証者が用いるアルゴリズムに手を加えるだけで高い安全性を満たす RKA 安全な署名方式を利用可能であることが示された．

5.2 今後の課題

$\mathrm{NIKE}_{\mathrm{fac}}$ 方式の RKA 安全性を解析するにあたっては，線形関数に関する RKA 安全性のみを考え，アフィン関数や多項式などのより広い RKD 関数のクラスは考慮していない（図 5.1 の定理 3 の分離）．どのくらい RKD 関数のクラスを広げられるのかについては今後の課題である．なお，$\mathrm{NIKE}_{\mathrm{fac}}$ 方式は，利用する符号付剰余群の位数が攻撃者には未知であったため，線形関数に関する RKA 安全性しか証明できなかったと予想される．そのため，$\mathrm{NIKE}_{\mathrm{fac}}$ 方式に変更を加え，位数が既知であるような素数位数の群を用いれば，アフィン関数に関する RKA 安全性が示されるのではないかと考えられる．

既存の署名方式の RKA 安全性に関しては，表 5.1 および表 5.2 内で「-」で示されるように，まだ証明のついていない課題が残っている．特に，第二の DSA の変型および修正 ElGamal 署名方式が Φ^{lin}-EUF-CM-RKA に関して脆弱かどうかは知られていないため，これは今後の課題である（図 5.2 内の下の段の「-」）．また，本論文で提案した改良署名方式は Bao ら [2] によって示されたビットフリッピング攻撃については安全性を証明できないため，こうした攻撃に対しても安全性証明のつく方式を構成することが今後の課題である．

本論文では，NIKE および署名方式のみを扱ったが，同様の議論を暗号化方式やサインクリプションなどの，公開鍵暗号系の他の要素技術に対しても応用し，既存の公開鍵暗号要素技術の RKA 安全性を広く調査することが望まれる．

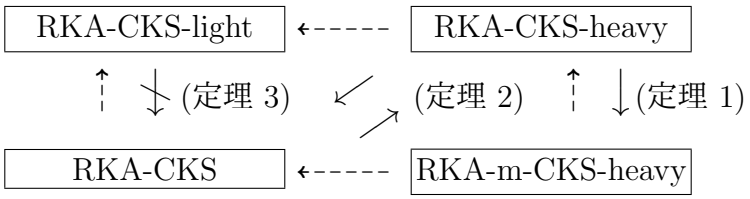

図 5.1: （図 3.3 の再掲）NIKE の RKA 安全性定義間の関係．
点線での矢印は自明な含意関係である．直線での矢印は本論文で示す含意関係を表し，否定の矢印は本論文で含意関係を満たさない方式の存在を示すことを表す．

表 5.1: （表 1.1 の再掲）既存の署名方式の既存の安全性（既存の結果）

	Schnorr	DSA	DSA の変型	ElGamal	修正 ElGamal
既存の安全性	✓	-	✓	×	✓

表 5.2: （表 1.2 の再掲）既存の署名方式の RKA 安全性（本論文の結果）

	Schnorr	DSA	DSA の変型	ElGamal	修正 ElGamal
wRKA（多項式）	✓	-	✓	-	✓
RKA（線形）	×	×	-	×	-

表 5.3: （表 1.3 の再掲）本論文で提案する署名方式の RKA 安全性（本論文の結果）

	改良 Schnorr	改良 DSA	改良 ElGamal
RKA（多項式）	✓	✓	✓

参考文献

[1] M. Abdalla, F. Benhamouda, A. Passelègue, and K.G. Paterson, "Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier," CRYPTO 2014, LNCS, vol.8616, pp.77–94, Springer, 2014.

[2] F. Bao, R.H. Deng, Y. Han, A.B. Jeng, A.D. Narasimhalu, and T. Ngair, "Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults," Security Protocols 1997, LNCS, vol.1361, pp.115–124, Springer, 1997.

[3] M. Bellare, D. Cash, and R. Miller, "Cryptography Secure against Related-Key Attacks and Tampering," ASIACRYPT 2011, pp.486–503, 2011.

[4] M. Bellare and T. Kohno, "A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications," EUROCRYPT 2003, pp.491–506, 2003.

[5] M. Bellare and G. Neven, "Multi-signatures in the plain public-Key model and a general forking lemma," CCS 2006, pp.390–399, ACM, 2006.

[6] M. Bellare, K.G. Paterson, and S. Thomson, "RKA Security beyond the Linear Barrier: IBE, Encryption and Signatures," ASIACRYPT 2012, LNCS, vol.7658, pp.331–348, Springer, 2012.

[7] D.J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, "High-Speed High-Security Signatures," CHES 2011, LNCS, vol.6917, pp.124–142, Springer, 2011.

[8] R. Bhattacharyya and A. Roy, "Secure Message Authentication Against Related-Key Attack," FSE 2013, LNCS, vol.8424, pp.305–324, Springer, 2013.

[9] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," CRYPTO '97, LNCS, vol.1294, pp.513–525, Springer, 1997.

[10] D. Boneh and X. Boyen, "Short Signatures Without Random Oracles," EUROCRYPT 2004, LNCS, vol.3027, pp.56–73, Springer, 2004.

[11] D. Cash, E. Kiltz, and V. Shoup, "The Twin Diffie-Hellman Problem and Applications," EUROCRYPT 2008, LNCS, vol.4965, pp.127–145, Springer, 2008.

[12] I. Damgård, S. Faust, P. Mukherjee, and D. Venturi, "Bounded Tamper Resilience: How to Go beyond the Algebraic Barrier," ASIACRYPT 2013, LNCS, vol.8270, pp.140–160, Springer, 2013.

[13] I. Damgård, S. Faust, P. Mukherjee, and D. Venturi, "The Chaining Lemma and Its Application," ICITS 2015, LNCS, vol.9063, pp.181–196, Springer, 2015.

[14] W. Diffie and M.E. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, vol.22, no.6, pp.644–654, 1976.

[15] Y. Dodis, J. Katz, A.D. Smith, and S. Walfish, "Composability and On-Line Deniability of Authentication," TCC 2009, LNCS, vol.5444, pp.146–162, Springer, 2009.

[16] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on A.E.S," ACNS 2003, LNCS, vol.2846, pp.293–306, Springer, 2003.

[17] S. Dziembowski, K. Pietrzak, and D. Wichs, "Non-Malleable Codes," ICS 2010, pp.434–452, Tsinghua University Press, 2010.

[18] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," CRYPTO '84, LNCS, vol.196, pp.10–18, Springer, 1984.

[19] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, vol.31, no.4, pp.469–472, 1985.

[20] S. Faust, P. Mukherjee, J.B. Nielsen, and D. Venturi, "Continuous Non-malleable Codes," TCC 2014, LNCS, vol.8349, pp.465–488, Springer, 2014.

[21] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs, "Efficient Non-malleable Codes and Key-Derivation for Poly-size Tampering Circuits," EUROCRYPT 2014, LNCS, vol.8441, pp.111–128, Springer, 2014.

[22] E.S.V. Freire, J. Hesse, and D. Hofheinz, "Universally Composable Non-Interactive Key Exchange," SCN 2014, Lecture Notes in Computer Science, vol.8642, pp.1–20, Springer, 2014.

[23] E.S.V. Freire, D. Hofheinz, E. Kiltz, and K.G. Paterson, "Non-Interactive Key Exchange," PKC 2013, LNCS, vol.7778, pp.254–271, Springer, 2013.

[24] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering," TCC 2004, LNCS, vol.2951, pp.258–277, Springer, 2004.

[25] C. Giraud, "DFA on AES," AES 2004, LNCS, vol.3373, pp.27–41, Springer, 2004.

[26] S. Goldwasser, S. Micali, and R.L. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM J. Comput., vol.17, no.2, pp.281–308, 1988.

[27] V. Goyal, A. O'Neill, and V. Rao, "Correlated-Input Secure Hash Functions," TCC 2011, LNCS, vol.6597, pp.182–200, Springer, 2011.

[28] Z. Jafargholi and D. Wichs, "Tamper Detection and Continuous Non-malleable Codes," TCC 2015, LNCS, vol.9014, pp.451–480, Springer, 2015.

[29] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated Verifier Proofs and Their Applications," EUROCRYPT '96, pp.143–154, 1996.

[30] N. Koblitz and A.J. Menezes, "The random oracle model: a twenty-year retrospective," Des. Codes Cryptography, vol.77, no.2-3, pp.587–610, 2015.

[31] T. Matsuda, K. Matsuura, and J.C.N. Schuldt, "Efficient Constructions of Signcryption Schemes and Signcryption Composability," INDOCRYPT 2009, LNCS, vol.5922, pp.321–342, Springer, 2009.

[32] H. Morita, J. Schuldt, T. Matsuda, G. Hanaoka, and T. Iwata, "On the Security of Schnorr Signatures, DSA, and ElGamal Signatures against Related-Key Attacks," IEICE Transactions, 2017. To appear.

[33] H. Morita, J.C.N. Schuldt, T. Matsuda, G. Hanaoka, and T. Iwata, "On the Security of the Schnorr Signature Scheme and DSA Against Related-Key Attacks," ICISC 2015, LNCS, vol.9558, pp.20–35, Springer, 2015.

[34] "National Institute of Standards and Technology (NIST), FIPS Publication 186: Digital Signature Standards (DSS)," 1994.

[35] D. Pointcheval and O. Sanders, "Forward Secure Non-Interactive Key Exchange," SCN 2014, Lecture Notes in Computer Science, vol.8642, pp.21–39, Springer, 2014.

[36] D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes," EUROCRYPT '96, pp.387–398, 1996.

[37] D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures," J. Cryptology, vol.13, no.3, pp.361–396, 2000.

[38] D. Pointcheval and S. Vaudenay, "On Provable Security for Digital Signature Algorithms." Technical Report, Ecole Normale Superieure, LIENS., 1996.

[39] B. Qin, S. Liu, T.H. Yuen, R.H. Deng, and K. Chen, "Continuous Non-malleable Key Derivation and Its Application to Related-Key Security," PKC 2015, LNCS, vol.9020, pp.557–578, Springer, 2015.

[40] C. Schnorr, "Efficient Identification and Signatures for Smart Cards," CRYPTO '89, LNCS, vol.435, pp.239–252, Springer, 1989.

[41] H. Wee, "Public Key Encryption against Related Key Attacks," PKC 2012, LNCS, vol.7293, pp.262–279, Springer, 2012.

研究業績

1. 本論文を構成する学術論文誌論文

1. H. Morita, J. Schuldt, T. Matsuda, G. Hanaoka, and T. Iwata, “On the Security of Schnorr Signatures, DSA, and ElGamal Signatures against Related-Key Attacks”, IEICE Transactions, 100-A(1): 73–90, 2017.（第 4 章）

2. H. Morita, J. Schuldt, T. Matsuda, G. Hanaoka, and T. Iwata, “On the Security of Non-Interactive Key Exchange against Related-Key Attacks”, IEICE Transactions, accepted, 2017.（第 3 章）

2. その他の研究業績

国際会議（査読付き）

1. Kazuhiko Minematsu, Stefan Lucks, Hiraku Morita, and Tetsu Iwata. Attacks and Security Proofs of EAX-Prime. In Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11–13, 2013. Revised Selected Papers, pages 327–347, 2013.

2. Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the Security of the Schnorr Signature Scheme and DSA Against Related-Key Attacks. In Information Security and Cryptology - 18th International Conference, ICISC 2015, Korea, November 25-27, 2015. Vol. 9558 of LNCS, pp.20–35. Springer, 2015.

国内研究会（査読なし）

1. 小林 隼人, 森田 啓, 岩田 哲. OKH 認証暗号化方式に対する攻撃. 2013 年暗号と情報セキュリティシンポジウム, SCIS 2013, 1B2-2, 2013/1

2. 森田 啓, 松田 隆宏, 花岡 悟一郎, 岩田 哲. 関連鍵攻撃に対する Schnorr 署名方式の安全性について. 2015 年暗号と情報セキュリティシンポジウム, SCIS 2015, 2E3-2, 2015/1.

3. 森田 啓, Jacob Schuldt, 松田 隆宏, 花岡 悟一郎, 岩田 哲. 関連鍵攻撃に対する Non-Interactive Key Exchange の安全性について. 2016 年暗号と情報セキュリティシンポジウム, SCIS 2016, 4E1-4, 2016/1.

謝辞

本博士論文は，筆者が名古屋大学大学院工学研究科計算理工学専攻博士課程後期課程在学中に基盤計算科学講座先端情報環境グループにおいて行った研究をまとめたものです．

本研究に関して，アイディア段階から終始ご指導ご鞭撻を頂きました本学の岩田哲准教授に心より感謝致します．

本論文をご精読頂き多くのコメントを頂きました本学の河口信夫教授，楫勇一教授，藤井俊彰教授，産業技術総合研究所の花岡悟一郎研究グループ長に深く感謝致します．

本研究の遂行および論文執筆に当たり，多くの助言を頂きました産業技術総合研究所の松田隆宏研究員，Jacob C.N. Schuldt 研究員の協力なくしては本論文を書き上げることは不可能であったことを記すとともに，深く感謝申し上げます．

研究を進めるにあたり，様々な視点から質問をしてくれた研究室の後輩に感謝しております．また，事務手続きを一手に引き受けて下さいました秘書の岩瀬いずみさんに感謝しております．

最後になりますが，どんな時も温かく見守り心の支えになってくれた両親と，応援してくれた家族に感謝の意を表して謝辞といたします．