暗号アルゴリズム
「ECDHS (Elliptic Curve Diffie-Hellman Scheme) in SEC 1」
詳細評価 (攻撃評価) レポート

２００１年1月１２日

目次

1．ECDHS の概要

DHS(Diffie Hellman Scheme) は, Whitfield Diffie と Martin E. Hellman の公開鍵暗号の発見を記す論文において 1976 年に提案された鍵共有方式である。その論文では，有限体乗法群を用いたものであった。また，楕円曲線暗号は, Neal Koblitz と Victor Miller により独立に提案された。DHS の楕円曲線版 ECDHS (Elliptic Curve Diffie Hellman Scheme)もその時に認識され，その後，(EC)DHS の安全性の根拠である(EC)DLP (Discrete Logarithm Problem) と共に, (EC)DHS に対し，数多くの暗号，数学研究者により解(読)法が提出されてきた。

本 ECDHS は，SEC 1（暗号技術仕様書，自己評価書）では，“ 楕円曲線プリミティブ ”と“ 鍵導出関数プリミティブ ”を用いた“ 楕円 Diffie-Hellman スキーム ”の構成となっている。その細かい内容については付録１を参照のこと。この構成は，極めて基本的であり，妥当なものである。現時点での問題点は，特になく，自己評価も詳細に行われている。また，各研究者に SEC 1 に対する評価を依頼し, web 上で公開もしている。しかし, ECDHS に影響を与え得る各研究は，現在進展中であるので，その動向に注目しなければならない。
また，(EC)DHS を基にした方式がどのような条件を満たせば実用的で信頼できるシステムを構成し得るか，その条件の探索やそれへの対策も，現在の課題である。

本稿においては，まず, (EC)DLP そのものに対する攻撃法を見てみる。有限体乗法群を用いた DLP に対しては，数体篩を用いた準指数時間の解法アルゴリズムが存在するのに対し，一般的な ECDLP に対しては，指数時間の解法アルゴリズム（Pollard-ρ法, Pollard-λ法）しか知られていない。また, Pohlig-Hellman 法という generic 攻撃法もある。
有限体乗法群 DLP の時に有効であった指数計算法の適用が困難であることは，Miller によって，既に指摘されていた。しかし, ECDLP の特殊な場合には，準指数時間の解法 MOV 法, FR 法と，多項式時間の解法 SSSA 法がある。また，標数２のある楕円曲線の場合に有効な Weil Descent 法も提案されている。これら攻撃法に対する安全性を検討評価する。

次に ECDLP を直接攻撃するのでない ECDHS への攻撃法を見る。一般に DLP と DHP の同値性を確立しようという試みは，Maurer，Wolf らにより行われている。特に ECDLP と ECDHP の同値性に関しては, Boneh と Lipton による報告もある。DDH 仮定というものに本質的に暗号の安全性が依存していることが認識されてきたため，ECDLP，ECDHS と DDH 仮定との間の関係性が注目されている。これら最新研究と本スキームの安全性の関係について検討する。

その他，小部分群攻撃法に対する安全性，Man-in-the-middle 攻撃法に対する安全性，鍵導

出関数（key derivation function）への攻撃法に対する安全性, 秘密鍵生成に用いる擬似乱数生成法の安全性が評価検討される。

2. ECDHS の安全性評価

2.1 ECDLP に対する攻撃法

2.1.1 Generic 攻撃法

2.1.1.1 Pollard-ρ法, Pollard-λ法に対する安全性

【自己評価書 SEC 1 における記述（和訳引用，文中引用文献略)】

一般に，ECDLP に対して，準指数時間アルゴリズムは知られていない。現時点での最良の一般的アルゴリズムは，Pollard-ρ法，Pollard-λ法に基づいている。Pollard-ρ法は，約 $\sqrt{\pi n/2}$ ステップを要し, Pollard-λ法は約 $2\sqrt{n}$ ステップを要する。両方法とも効率的にパラレル化できる。

Gallant-Lambert-Vanstone と Wiener-Zuccherato は最近 Pollard-ρ法を $\sqrt{2}$ 倍早くできることを示した。それゆえ，このスピードアップを考慮に入れると，Pollard-ρ法の期待実行時間スピードアップは $\sqrt{\pi n/4}$ となる。

彼らは，$\mathbf{F}_{2^e}$ 上定義された楕円曲線 Eの $\mathbf{F}_{2^{ed}}$ 有理点群の時に，更にスピードアップできることを示した。この場合，Pollard-ρ法は $\sqrt{2d}$ 倍高速化できることを彼らは示した。例えば，Koblitz 曲線 E: $y^2 + xy = x^3 + x^2 + 1$ の $\mathbf{F}_{2^{163}}$ 有理点群は$\#E(\mathbf{F}_{2^{163}})=2n$, n は 162 ビットの素数である。Pollard-ρ法の効率化により，$E(\mathbf{F}_{2^{163}})$の ECDLP は，同様の位数のランダム曲線 ECDLP を解くのに 2^{81} ステップ要するのに対し，約 2^{77} ステップで解けることを示した。

下図は，ECDLP の困難さを示している。改良 Pollard-ρ法をソフトウェアで実装した時の一般の曲線に対して ECDLP を解く計算量の MIPS 年での見積りも含まれている。下図の付加説明として, Odlyzko は, 2004 年に世界中のコンピューターの 0.1%を 1 年動員した時, その計算量を 10^8 MIPS 年と見積っており，それは，2014 年には 10^{10} 乃至 10^{11} MIPS 年とも見積っている。下図は, ANSI X9.62 から抜粋したものである。その見積もりがどのようにして得られたかの更なる詳細については，それを参照のこと。

n のサイズ (ビット)	$\sqrt{\pi n/4}$	MIPS 年
160	2^{80}	8.5×10^{11}
186	2^{93}	7.0×10^{15}
234	2^{117}	1.2×10^{23}
354	2^{177}	1.3×10^{41}
426	2^{213}	9.2×10^{51}

ECDLP の困難さについては, van Oorschot と Wiener の 1994 年の論文で示されている。

van Oorschot と Wiener は，ECDLP を解く特殊仕様ハードウェア構築の可能性について，詳細に調べた。彼らは，$n \approx 2^{120}$ となる楕円曲線 E に対して，ECDLP を 35 日で解く 325,000 個のプロセッサを持つマシンが 1000 万ドルでできると見積った。現時点では，$n \approx 2^{160}$ となるような大きい n については，ハードウェア改良による攻撃は，完全に不可能と思われる。

【考察】

Pollard-ρ法, Pollard-λ法は，現在最も汎用的攻撃法であり，その可能性が見出された時には，いかなる楕円暗号も（少なくとも，現在の）意味をなさない。それゆえ，漸近的計算時間だけでなく実計算時間も詳細に検討される必要があり，本評価書においても詳細にレポートがなされていると思われる。上記，評価書の記述は妥当なものであり，当面は，Pollard-ρ法, Pollard-λ法の現在の形での適用は不可能であることがわかる。

2.1.1.2 Pohlig-Hellman 法に対する安全性

【技術仕様書 SEC 1 における記述（和訳引用，文中引用文献略)】

“ 楕円曲線パラメータ生成プリミティブ ”に暗号に用いる楕円曲線群の位数 n が素数になるように定められている。

【考察】

Pohlig-Hellman 法は，n が小因数を含む時に，秘密鍵の部分情報を導出する解読法であり，特に群位数が素数の時には回避できるので，十分妥当な配慮である。

また，本攻撃法は，Pollard-ρ法，Pollard-λ法と同じく任意の可換群に対して，理論上適用可能なので, generic な攻撃法と言われている。

SEC 1仕様記述中， **3.1.1.1 Actions** 2， **3.1.1.2.1 Actions** 5， **3.1.2.1 Actions** 3, **3.1.2.2.1 Actions** 6 において，配慮がなされている（付録1 参照）。

2.1.2　Non-generic 攻撃法

2.1.2.1　MOV 法, FR 法に対する安全性

【自己評価書 SEC 1 における記述（和訳引用，文中引用文献略)】

楕円曲線の $\mathbf{F}_q$有理点群で，小さいBに対して，nが$q^B - 1$を割り切る時に，Menezesと OkamotoとVanstoneによる攻撃法やFreyとRuckによる攻撃法が適用される。これらの攻撃法はその楕円曲線のECDLPを$\mathbf{F}_q$の小次数拡大体の通常のDLPに効率的に変換する。
（中略）
これら（暗号的に）弱いクラスの曲線はこのドキュメントにおいては除外されている。

“楕円曲線ドメインパラメータ生成プリミティブ”に暗号に用いる楕円曲線群の位数 n が $q^B \equiv 1 \pmod{n}$ （$1 \leqq B < 20$）とならないように定められている。

【考察】

十分妥当な配慮がなされている。

仮に，$B = 20$, $\log_2 q \approx 160$で$q^B \equiv 1 \pmod{n}$が成立したとしても，現時点では，セキュリティパラメータ$B \times \log_2 q \approx 3200$ の通常のDLPは十分困難なものである。また，$q^B \equiv 1 \pmod{n}$ （$1 \leqq B < 20$）とならないかどうかのチェックの計算時間も通常は問題にならないものである。ランダムに選んだほとんどの楕円曲線に対して，この攻撃法が適用できないことも知られている。

SEC 1仕様記述中，**3.1.1.1 Actions** 2， **3.1.1.2.1 Actions** 8， **3.1.2.1 Actions** 3，**3.1.2.2.1 Actions** 9　において，配慮がなされている（付録1 参照）。

2.1.2.2　SSSA法に対する安全性

【自己評価書における記述（和訳引用，文中引用文献略）】

楕円曲線の$\mathbf{F}_q$有理点群で，$\#E(\mathbf{F}_q)= q$となる時，Semaev, Smart，佐藤と荒木による攻撃法が適用できる。この攻撃法はその楕円曲線群を$\mathbf{F}_q$の加法群に効率的に写像する。
（中略）
これら（暗号的に）弱いクラスの曲線はこのドキュメントにおいては除外されている。

“楕円曲線パラメータ生成プリミティブ”に暗号に用いる楕円曲線群の位数 n とそのコファクターの位数 h が $nh \neq q$ となるように定められている。

【考察】

2.1.2.1 のMOV法，FR法が，理論上どの楕円曲線に対しても適用可能である（実際的には，ほとんどの場合，無効）のに対し，本攻撃法は，理論上も特殊な楕円曲線にのみ，適用可能である。また，2.1.2.1 が準指数時間攻撃を導くのに対し，本攻撃法は，多項式時間での攻撃を導く。それらの意味で本攻撃法は，より特殊性に依存した強力な方法と言えよう。その高速アルゴリズムの故に，解読法だけでなく，暗号構成とも関係を持っていることが知られている。

以上のことから，現時点では，ほぼ唯一の可能性を排除すればよく，SEC 1においても十分妥当な配慮がなされていると言える。

SEC 1仕様記述中，**3.1.1.1 Actions** 2，**3.1.1.2.1 Actions** 8，**3.1.2.1 Actions** 3，**3.1.2.2.1 Actions** 9　において，配慮がなされている（付録1 参照）。

2.1.2.3　Weil descent法(Gaudry-Hess-Smart attack) に対する安全性

【自己評価書 SEC 1 における記述（和訳引用，文中引用文献略）】

楕円曲線の $\mathbf{F}_{2^m}$ 有理点群で，mが合成数の時，Weil descentに基づいた攻撃法が適用される可能性がある。GalbraithとSmartの論文, GaudryとHessとSmartの論文を参照。
これら（暗号的に）弱いクラスの曲線はこのドキュメントにおいては除外されている。

“楕円曲線パラメータ生成プリミティブ”に m が素数 113, 131, 163, 193, 233, 239, 283, 409, 571 のいずれかであるように定められている。

【考察】

Weil descentに基づいた攻撃法は, 1998年のECC98でGerhard Freyの講演“How to disguise an elliptic curve”に触発されて，上記のGalbraithとSmartの論文, GaudryとHessとSmartの論文が発表されたことで，広く知られるようになった攻撃法である。その攻撃法は発見されてまだ間もなく，その適用可能性の完全な理解にはまだ至っていないので，今後の研究動向が注目されるトピックとなっている。

mが合成数の時に引っかかる可能性が高まることがわかるので，上記記述は妥当な配慮と思われる。実際, MenezesとQuの論文では, mが$160 \leqq m \leqq 600$の素数となる場合には, Gaudry-Hess-Smartによる方法が有効でないと述べられている。しかし，その論文でも強調されているように，これはmが$160 \leqq m \leqq 600$の素数となる場合でも, Weil descent に基づいた攻撃法の適用可能性を完全に否定するものではない。

今後の研究動向が注目される。

SEC 1仕様記述中， **3.1.2.1 Actions** 1， **3.1.2.2.1 Actions** 1　において，配慮がなされている（付録1 参照）。

2.2　ECDLP 以外の ECDHS に対する攻撃法

2.2.1　ECDHPの安全性とECDDH仮定

【自己評価書 SEC 1 における記述（和訳引用，文中引用文献略）】

両楕円曲線 Diffie-Hellman プリミティブ（基本スキームとコファクタースキーム）に対する第一のセキュリティ要件は，“ U と V の公開鍵しか知らない攻撃者は，（Uと Vが）共有するフィールド要素を計算できない ”ということである。この要件は，楕円曲線 Diffie-Hellman 問題(ECDHP)が困難であるということと同値である。ECDHP は次のように述べられる。
E を $\mathbf{F}_q$上定義された楕円曲線, $G \in E(\mathbf{F}_q)$が大きい素数位数 n を持つとする。ECDHP とは, E, Gと Gのスカラ倍点 $Q_1=d_1G$, $Q_2=d_2G$が与えられた時に, d_1d_2Gを計算せよ，という問題である。
ECDHP は ECDLP と密接に関連している。明らかに，もし，ECDLP が容易であれば，ECDHP も容易である。Boneh と Lipton は，実際的な目的のためには，逆もまた正しいことを示した。即ち，もし実際に, ECDLP に対する最良のアルゴリズムが，指数時間を要すれば, ECDLP と ECDHP は同値であることを示した。
Diffie-Hellman プリミティブをベースにした多くのスキームは，実際，共有されたフィールド要素が攻撃者によって推測困難というだけでなく，その要素が攻撃者から全くランダムに見えるという強い仮定に基づいている。この要件や，それと ECDHP との関連に関する議論は，D.Boneh “ The decision Diffie-Hellman problem ”, D.Boneh, R.Vankatesan “ Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes ”を参照のこと。

【考察】

ECDLP と ECDHP（Elliptic Curve Diffie-Hellman Problem）との関係，ECDHP と ECDDH（Elliptic Curve Decision Diffie-Hellman）仮定との関係を扱った本記述は，最新研究成果の一部の紹介であり，現時点で本 ECDHS の実装そのものに要件を課しているわけでない。SEC 1 の上記記述は，妥当なものである。

先に, ECDLP と ECDHP との関係の研究が ECDHS に与える影響を補足する。
上記記述にあるように，まだその同値性は，無条件に示されたわけではない。実運用する者に対して，もし，その同値性が無条件に示された楕円曲線が存在するならば，それを選択的に用いたいと思うであろう。その点に関して，実際，一般の DLP, DHP に対して, den Boer

と Maurer による研究により，群位数が素数 p の時に, p-1 か p+1 が小さい素因数しか持たない時には，無条件にこの同値性が成立することが，以前から知られていた（Boneh と Lipton の上記成果はこれら研究の延長線上にある)。しかし，現在，上記条件の楕円曲線を探索すること（同値性の証明をつけること）や, Maurer や Wolf により示された他の結果は, 計算時間の点で，アルゴリズムとして，実際的に広くは使えない。
また, *Crypto*'94 で, Maurer は次のように述べている。

“ 位数が素数 p であり, p-1 か p+1 が小さい素因数しか持たない群 G を用いることは，例え（G の）離散対数問題を解く有効な方法が知られていないとしても，いい方法かどうかは疑わしいと思われる。そのような G に対する有効な離散対数アルゴリズムの探索を，オープンプロブレムとして，指摘する。”

次に, ECDHP と ECDDH 仮定の関係の研究が ECDHS に与える影響を補足する。
“ その要素が攻撃者から全くランダムに見えるという強い仮定 ” は ECDDH（Elliptic Curve Decision Diffie-Hellman）仮定と呼ばれるものである。(EC)DHP の困難さと (EC)DDH 仮定の差を探る研究は，現在の話題であり，有限体乗法群の場合は部分的結果が知られている。その楕円曲線群版は, Boneh の ANTSⅢでの講演においてオープンプロブレムとして，指摘されている。

以上の今後の研究動向とその影響が注目される。

2.2.2　小部分群攻撃法に対する安全性

【自己評価書 SEC 1 の記述（和訳引用，文中引用文献略）】

これまでの楕円Diffie-Hellmanプリミティブに関する議論は，標準的なスキームとコファクタースキーム双方に適切なものであった。このセクションの残りで２つのプリミティブの違いについて説明する。
ECDHPに対する直接的な攻撃だけが，Diffie-Hellmanプリミティブを用いたスキームに攻撃する唯一の方法でない。そのプリミティブを用いた多くのスキームは，小部分群攻撃法を受ける可能性がある。小部分群攻撃法とは，攻撃者が，上記プリミティブを用いる時に，Vの公開鍵を小位数点に置き換えて，Uに予測可能なフィールド要素を計算させることによって行われる。鍵共有スキームにおいて，この攻撃法の影響は深刻なものである。例えば，結果的に，U と V によって共有されるセッション鍵の強度を落としたり，U のスタティックな秘密鍵の落とすことになる可能性がある。
ここでは，本攻撃法に対する２つの防御法を提示する。１つ目は，Vの公開鍵を検証し，標準的なDiffie-Hellmanプリミティブを用いるものである（Vの公開鍵を検証することで，Vの公開鍵が位数nであることをチェックし，本攻撃法を防ぐ）。２つ目は，Vの公開鍵を部分検証し，コファクターDiffie-Hellmanプリミティブを用いるものである（小部分群内の点を使って，コファクターDiffie-Hellmanプリミティブを行うと，無限遠点が計算されるので，その場合は，その鍵はリジェクトされる）。
上記防御法のどちらを用いるのが適当かということは，状況によって決まる。既存の標準的なDiffie-Hellmanプリミティブとの互換性要求によって決まったり（その場合は，１つ目の防御法なら互換性を保つが，２つ目は保たない）システム効率要求の大小によって決まったりする（２つ目の防御法は，通常１つ目より効率がよい）。

【考察】

十分妥当な記述である。コファクターDiffie-Hellmanプリミティブは，IEEE P1363等にも定められており広く知られた構成法と思われる。

2.2.3　Man-in-the-middle 攻撃法に対する安全性

【自己評価書 SEC 1 の記述（和訳引用，文中引用文献略）】

もし，楕円Diffie-Hellmanスキームが，注意深く適用されなければ，Q_UやQ_Vが交換された時に，それらを変更することで，攻撃が可能になる。その結果として，スキームは，implicit key authentication や，known key security というようなセキュリティゴールを達成することができなくなる。このような能動的な攻撃法に対してのいくつかの防御法が一般に使われる。例えば，Q_U,Q_V を署名付きメッセージの中に入れるとか，Q_U,Q_Vを認証するというようなことである。

【考察】

本攻撃法への防御は，ECDHS 適用状況に応じて，その都度，考慮されねばならないものである。SEC 1においては，幅広く仕様書として，適用するために，細かい議論にふれてはいないが，実運用上は，大事な課題である。SEC 1の記述自体は，妥当なものである。

2.2.4　鍵導出関数（key derivation function）への攻撃法に対する安全性

【自己評価書 SEC 1 の記述（和訳引用，文中引用文献略）】

楕円Diffie-Hellmanスキームにおいて用いられる鍵導出関数は，スキームの安全性をたもつために，いくつかの性質を有さなければならない。例えば，もし鍵導出関数の出力のいくつかのビットを，攻撃者が予測することができたり，鍵導出関数の出力間にある相関関係が見出せたりしたならば，攻撃者は共有されたデータに関する情報を得ることが可能になってしまう。

【考察】

鍵導出関数への攻撃法は，ハッシュ関数それ自体に対してと，鍵導出関数構成に対してのものが考えられる。

ハッシュ関数は, SEC 1においては, SHA-1がサポートされている。ANSI,NISTでSHA-2の標準化が終われば, SHA-2のサポートも予定されていると記述されている。SHA-1は2^{61}以下の長さのオクテットストリングを長さ20のオクテットストリングに変換するものである。実際のハッシュ関数の衝突確率や，それら衝突確率に対する要求条件，証明可能なスキーム構築における役割分析などは現在なお継続的に研究されており，その動向に注目する必要がある。

鍵導出関数構成は, ANSI-X9.63-KDFがサポートされている。今後，他の鍵導出関数のサポートも考慮に入っているとの記述もある。鍵導出関数構成は，適用状況に応じた*Sharedinfo*の与え方など，適用システム全体のセキュリティ評価と連動して，慎重に選択されるべきものである。

SEC 1の鍵導出関数に関する記述ならびにサポート状況は，現時点で妥当なものである。上述した通り，今後の研究動向に対する注目が必要である。

2.2.5　秘密鍵生成に用いる擬似乱数生成法の安全性

【自己評価書 SEC 1 の記述（和訳引用，文中引用文献略）】

鍵生成は，楕円Diffie-Hellmanスキームの鍵配置プロシジャー(key deployment procedure)の中で行われる。安全な乱数または擬似乱数生成法は，例えば，*V*の鍵生成において，予測可能な秘密鍵を選んでしまうのを防ぐために必要とされる。安全でない乱数または擬似乱数生成法は，多分暗号システムに対して，暗号攻撃を招く最もありがちな原因である。

【考察】

安全な擬似乱数生成の必要性は，改めて言うまでもない。例えば，FIPS-PUB 186-2“ DSS ”では，SHA-1または，DESを基にした擬似乱数生成法を推奨している。

直接ECDHSと関係はないが，不用意な擬似乱数生成法の使用が安全性を損なう例として，“ DSS ”の擬似乱数生成法に線形合同法(Linear congruential generators)を用いた時には，安全性が損なわれることが，*Crypto*'97でBellareらによって報告されている。上記FIPS-PUBの推奨法は，現時点では，もちろん問題はないが，擬似乱数生成は，暗号の要といえるものであるので，実装する上では，細心の配慮が必要とされる。

3. まとめ

本 ECDHS 詳細評価レポートにおいては，実装法に対する攻撃（フォールトに基づく攻撃法，電力解析攻撃法，タイミング解析攻撃法）を除いたECDHSに対する主々の攻撃法を，評価した。現時点では，特に問題点はない。ただ，今後の研究動向に注目し，適切に反映させていく必要がある。

付録1.　ECDHSの仕様

ECDHSは, SEC 1中以下の部分の記述によっている（章, 節番号はSEC 1に従う）。

- **Elliptic Curve Primitives (3.1, 3.2, 3.3 節)**
- **Key Derivation Primitives (3.5, 3.6 節)**
- **Elliptic Curve Diffie-Hellman Scheme (6.1 節)**

上位２つのプリミティブを組み合わせて, 6.1節の仕様に従い, ECDHSは実行される。

Elliptic Curve Primitives は以下のように構成されている。

- **Elliptic Curve Domain Parameters (3.1 節)**
  - **・Elliptic Curve Domain Parameters over F_p (3.1.1 節)**
    Generation Primitives (3.1.1.1 節) と Validation (3.1.1.2 節)
  - **・Elliptic Curve Domain Parameters over F_p (3.1.2 節)**
    Generation Primitives (3.1.2.1 節) と Validation (3.1.2.2 節)
- **Elliptic Curve Key Pairs (3.2 節)**
  - **・Elliptic Curve Key Pair Generation Primitive (3.2.1 節)**
  - **・Validation of Elliptic Curve Public Keys (3.2.2 節)**
  - **・Partial Validation of Elliptic Curve Public Keys (3.2.3 節)**
- **Elliptic Curve Diffie-Hellman primitives (3.3 節)**
  - **・Elliptic Curve Diffie-Hellman Primitive (3.3.1 節)**
  - **・Elliptic Curve Cofactor Diffie-Hellman Primitive (3.3.1 節)**

Key Derivation Primitives は, SHA-1 ベースの ANSI-X9.63-KDF がサポートされている。

Elliptic Curve Diffie-Hellman Scheme は以下のように構成されている。

- **Scheme Setup (6.1.1 節)**
- **Key Deployment (6.1.2 節)**
- **Key Agreement Operation (6.1.3 節)**

ECDHSの仕様を, SEC 1より原文のまま, 抜粋する（章, 節番号もSEC 1に従う）。

Elliptic Curve Primitives

3.1 Elliptic Curve Domain Parameters

3.1.1 Elliptic Curve Domain Parameters over $\mathbf{F}_p$

Elliptic curve domain parameters over $\mathbf{F}_p$ are a sextuple:

$$T = (p, a, b, G, n, h)$$

consisting of an integer p specifying the finite field $\mathbf{F}_p$, two elements $a, b \in \mathbf{F}_p$ specifying an elliptic curve $E(\mathbf{F}_p)$defined by the equation:

$$E : y^2 = x^3 + a.x + b \pmod{p},$$

a base point $G = (x_G, y_G)$ on $E(\mathbf{F}_p)$, a prime n which is the order of G, and an integer h which is the cofactor $h = \#E(\mathbf{F}_p)/n$.

Elliptic curve domain parameters over $\mathbf{F}_p$ precisely specify an elliptic curve and base point. This is necessary to precisely define public-key cryptographic schemes based on ECC.

Section 3.1.1.1 describes how to generate elliptic curve domain parameters over $\mathbf{F}_p$, and Section 3.1.1.2 describes how to validate elliptic curve domain parameters over $\mathbf{F}_p$.

3.1.1.1 Elliptic Curve Domain Parameters over $\mathbf{F}_p$ Generation Primitive

Input: The approximate security level in bits required from the elliptic curve domain parameters — this must be an integer $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$.

Output: Elliptic curve domain parameters over $\mathbf{F}_p$:

$$T = (p, a, b, G, n, h)$$

such that taking logarithms on the associated elliptic curve is believed to require approximately 2^t operations.

Actions: Generate elliptic curve domain parameters over $\mathbf{F}_p$ as follows:

1. Select a prime p such that $\lceil \log_2 p \rceil = 2t$ if $t \neq 256$ and such that $\lceil \log_2 p \rceil = 521$ if $t = 256$ to determine the finite field $\mathbf{F}_p$.
2. Select elements $a, b \in \mathbf{F}_p$ to determine the elliptic curve $E(\mathbf{F}_p)$ defined by the equation:

   $$E : y^2 = x^3 + a.x + b \pmod{p},$$

   a base point $G = (x_G, y_G)$ on $E(\mathbf{F}_p)$,a prime n which is the order of G, and an integer h which is the cofactor $h = \#E(\mathbf{F}_p)/n$, subject to the following constraints:
   - $4.a^3 + 27.b^2 \not\equiv 0 \pmod{p}$.

- $\#E(\mathbf{F}_p) \neq p$.
- $p^B \not\equiv 1 \pmod{n}$ for any $1 \leqq B < 20$.
- $h \leqq 4$.

3. Output $T = (p, a, b, G, n, h)$.

This primitive allows any of the known curve selection methods to be used — for example the methods based on complex multiplication and the methods based on general point counting algorithms. However to foster interoperability it is strongly recommended that implementers use one of the elliptic curve domain parameters over F p specified in SEC 2 . See Appendix B for further discussion.

3.1.1.2 Validation of Elliptic Curve Domain Parameters over $\mathbf{F}_p$

There are four acceptable methods for an entity U to receive an assurance that elliptic curve domain parameters over $\mathbf{F}_p$ are valid. Only one of the methods must be supplied, although in many cases greater security may be obtained by carrying out more than one of the methods.

The four acceptable methods are:

1. U performs validation of the elliptic curve domain parameters over $\mathbf{F}_p$ itself using the validation primitive described in Section 3.1.1.2.1.
2. U generates the elliptic curve domain parameters over $\mathbf{F}_p$ itself using a trusted system using the primitive specified in Section 3.1.1.1.
3. U receives assurance in an authentic manner that a party trusted with respect to U's use of the elliptic curve domain parameters over $\mathbf{F}_p$ has performed validation of the parameters using the validation primitive described in Section 3.1.1.2.1.
4. U receives assurance in an authentic manner that a party trusted with respect to U's use of the elliptic curve domain parameters over $\mathbf{F}_p$ generated the parameters using a trusted system using the primitive specified in Section 3.1.1.1.

Usually when U accepts another party's assurance that elliptic curve domain parameters are valid, the other party is a CA.

3.1.1.2.1 Elliptic Curve Domain Parameters over $\mathbf{F}_p$ Validation Primitive

Input: Elliptic curve domain parameters over $\mathbf{F}_p$.

$$T = (p, a, b, G, n, h),$$

along with an integer $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$ which is the approximate security level in bits required from the elliptic curve domain parameters.

Output: An indication of whether the elliptic curve domain parameters are valid or not — either 'valid' or 'invalid'.

Actions: Validate the elliptic curve domain parameters over $\mathbf{F}_p$ as follows:

1. Check that p is an odd prime such that $\lceil \log_2 p \rceil = 2t$ if $t \neq 256$ or such that $\lceil \log_2 p \rceil = 521$ if $t = 256$.
2. Check that a, b, x_G and y_G are integers in the interval $[0, p-1]$.
3. Check that $4.a^3 + 27.b^2 \not\equiv 0 \pmod{p}$.
4. Check that $y^2 = x^3 + a.x + b \pmod{p}$.
5. Check that n is prime.
6. Check that $h \leqq 4$, and that $h = \lfloor (\sqrt{p}+1)^2 / n \rfloor$.
7. Check that $nG = O$.
8. Check that $p^B \not\equiv 1 \pmod{n}$ for any $1 \leqq B < 20$, and that $nh \neq p$.
9. If any of the checks fail, output 'invalid', otherwise output 'valid'.

Step 8 above excludes the known weak classes of curves which are susceptible to either the Menezes-Okamoto-Vanstone attack, or the Frey-Ruck attack, or the Semaev-Smart-Satoh-Araki attack. See Appendix B for further discussion.

If the elliptic curve domain parameters have been generated verifiably at random using SHA-1 as de-scribed in ANSI X9.62, it may also be checked that a and b have been correctly derived from the random seed.

3.1.2 Elliptic Curve Domain Parameters over $\mathbf{F}_{2^m}$

Elliptic curve domain parameters over $\mathbf{F}_{2^m}$ are a septuple:

$$T = (m, f(x), a, b, G, n, h)$$

consisting of an integer m specifying the finite field $\mathbf{F}_{2^m}$, an irreducible binary polynomial $f(x)$ of degree m specifying the representation of $\mathbf{F}_{2^m}$, two elements $a, b \in \mathbf{F}_{2^m}$ specifying the elliptic curve $E(\mathbf{F}_{2^m})$ defined by the equation:

$$y^2 + x.y = x^3 + a.x + b \quad \text{in } \mathbf{F}_{2^m},$$

a base point $G = (x_G, y_G)$ on $E(\mathbf{F}_{2^m})$,a prime n which is the order of G, and an integer h which is the cofactor $h = \#E(\mathbf{F}_{2^m})/n$.

Elliptic curve domain parameters over $\mathbf{F}_{2^m}$ precisely specify an elliptic curve and base point. This is necessary to precisely define public-key cryptographic schemes based on ECC.

Section 3.1.2.1 describes how to generate elliptic curve domain parameters over $\mathbf{F}_{2^m}$, and Section 3.1.2.2 describes how to validate elliptic curve domain parameters over $\mathbf{F}_{2^m}$.

3.1.2.1 Elliptic Curve Domain Parameters over $\mathbf{F}_{2^m}$ Generation Primitive

Input: The approximate security level in bits required from the elliptic curve domain parameters — this must be an integer $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$.

Output: Elliptic curve domain parameters over $\mathbf{F}_{2^m}$:

$$T = (\mathrm{m}, f(x), a, b, G, n, h)$$

such that taking logarithms on the associated elliptic curve is believed to require approximately 2^t operations.

Actions: Generate elliptic curve domain parameters over $\mathbf{F}_{2^m}$ as follows:

1. Let t' denote the smallest integer greater than t in the set {64, 80, 96, 112, 128, 192, 256, 512 }. Select $m \in$ { 113, 131, 163, 193, 233, 239, 283, 409, 571 } such that $2t < m < 2t'$ to determine the finite field $\mathbf{F}_{2^m}$.
2. Select a binary irreducible polynomial $f(x)$ of degree m from Table 1 in Section 2.1.2 to determine the representation of $\mathbf{F}_{2^m}$.
3. Select elements $a, b \in \mathbf{F}_{2^m}$ to determine the elliptic curve $E(\mathbf{F}_{2^m})$ defined by the equation:

   $$E : y^2 + x.y = x^3 + a.x + b \quad \text{in } \mathbf{F}_{2^m},$$

   a base point $G = (x_G, y_G)$ on $E(\mathbf{F}_{2^m})$, a prime n which is the order of G, and an integer h which is the cofactor $h = \#E(\mathbf{F}_{2^m})/n$, subject to the following constraints:
   - $b \neq 0$ in $\mathbf{F}_{2^m}$.
   - $\#E(\mathbf{F}_{2^m}) \neq 2^m$.
   - $2^{mB} \not\equiv 1 \pmod{n}$ for any $1 \leqq B < 20$.
   - $h \leqq 4$.
4. Output $T = (\mathrm{m}, f(x), a, b, G, n, h)$.

This primitive also allows any of the known curve selection methods to be used. However to foster interoperability it is strongly recommended that implementers use one of the recommended elliptic curve domain parameters over $\mathbf{F}_{2^m}$ specified in SEC 2. See Appendix B for further discussion.

3.1.2.2 Validation of Elliptic Curve Domain Parameters over $\mathbf{F}_{2^m}$

There are four acceptable methods for an entity U to receive an assurance that elliptic curve domain parameters over $\mathbf{F}_{2^m}$ are valid. Only one of the methods must be supplied, although in many cases greater security may be obtained by carrying out more than one of the methods.

The four acceptable methods are:

1. *U* performs validation of the elliptic curve domain parameters over $\mathbf{F}_{2^m}$ itself using the validation primitive described in Section 3.1.2.2.1.
2. *U* generates the elliptic curve domain parameters over $\mathbf{F}_{2^m}$ itself using a trusted system using the primitive specified in Section 3.1.2.1.
3. *U* receives assurance in an authentic manner that a party trusted with respect to *U*'s use of the elliptic curve domain parameters over $\mathbf{F}_{2^m}$ has performed validation of the parameters using the validation primitive described in Section 3.1.1.2.1.
4. *U* receives assurance in an authentic manner that a party trusted with respect to *U*'s use of the elliptic curve domain parameters over $\mathbf{F}_{2^m}$ generated the parameters using a trusted system using the primitive specified in Section 3.1.2.1.

3.1.2.2.1 Elliptic Curve Domain Parameters over $\mathbf{F}_{2^m}$ Validation Primitive

Input: Elliptic curve domain parameters over $\mathbf{F}_{2^m}$:

$$T = (\mathrm{m}, f(x), a, b, G, n, h)$$

along with an integer $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$ which is the approximate security level in bits required from the elliptic curve domain parameters.

Output: An indication of whether the elliptic curve domain parameters are valid or not — either 'valid' or 'invalid'.

Actions: Validate the elliptic curve domain parameters over $\mathbf{F}_{2^m}$ as follows:

1. Let *t'* denote the smallest integer greater than *t* in the set { 64, 80, 96, 112, 128, 192, 256, 512 } .Check that *m* is an integer in the set { 113, 131, 163, 193, 233, 239, 283, 409, 571 } such that $2t < m < 2t'$.
2. Check that $f(x)$ is a binary irreducible polynomial of degree *m* which is listed in Table 1 in Section 2.1.2.
3. Check that *a*, *b*, x_G and y_G are binary polynomials of degree $m - 1$ or less.
4. Check that $b \neq 0$ in $\mathbf{F}_{2^m}$.
5. Check that $y_G^2 + x_G.y_G = x_G^3 + a.x_G + b$ in $\mathbf{F}_{2^m}$.
6. Check that *n* is prime.
7. Check that $h \leqq 4$, and that $h = \lfloor(\sqrt{2^m} + 1)^2/n\rfloor$.
8. Check that $nG = O$.
9. Check that $2^{mB} \not\equiv 1 \pmod{n}$ for any $1 \leqq B < 20$, and that $nh \neq 2^m$.
10. If any of the checks fail, output 'invalid', otherwise output 'valid'.

Steps 1 and 9 above excludes the known weak classes of curves which are susceptible to either the Menezes-Okamoto-Vanstone attack, or the Frey-Ruck attack, or the Semaev-Smart-Satoh-Araki attack, or to attacks based on the Weil descent. See Appendix B for further discussion.

If the elliptic curve domain parameters have been generated verifiably at random using SHA-1 as described in ANSI X9.62, it may also be checked that a and b have been correctly derived from the random seed.

3.2 Elliptic Curve Key Pairs

3.2.1 Elliptic Curve Key Pair Generation Primitive

Input: Valid elliptic curve domain parameters $T = (p, a, b, G, n, h)$ or $(m, f(x), a, b, G, n, h)$.

Output: An elliptic curve key pair (d, Q) associated with T.

Actions: Generate an elliptic curve key pair as follows:

1. Randomly or pseudorandomly select an integer d in the interval $[1, n - 1]$.
2. Calculate $Q = dG$.
3. Output (d, Q).

3.2.2 Validation of Elliptic Curve Public Keys

There are four acceptable methods for an entity U to receive an assurance that an elliptic curve public key is valid. Only one of the methods must be supplied, although in many cases greater security may be obtained by carrying out more than one of the methods.

The four acceptable methods are:

1. U performs validation of the elliptic curve public key itself using the public key validation primitive described in Section 3.2.2.1.
2. U generates the elliptic curve public key itself using a trusted system.
3. U receives assurance in an authentic manner that a party trusted with respect to U's use of the elliptic curve public key has performed validation of the public key using the public key validation primitive described in Section 3.2.2.1.
4. U receives assurance in an authentic manner that a party trusted with respect to U's use of the elliptic curve public key generated the public key using a trusted system.

Usually when U accepts another party's assurance that an elliptic curve public key is valid, the other party is a CA who validated the public key during the certification process. Occasionally U may also receive assurance from another party other than a CA. For example, in the Station-to-Station protocol described in ANSI X9.63 , U receives an ephemeral public key from V. V is trusted with respect to U's use of the public key because U is attempting to establish a key with V and U only combines the public key with its own ephemeral key pair. It is therefore acceptable in this circumstance for U to accept assurance from V that the public key is valid because the public key is received in a signed message.

3.2.2.1 Elliptic Curve Public Key Validation Primitive

Input: Valid elliptic curve domain parameters $T = (p, a, b, G, n, h)$ or $(m, f(x), a, b, G, n, h)$, and an elliptic curve public key $Q = (x_Q, y_Q)$ associated with T.
Output: An indication of whether the elliptic curve public key is valid or not —either 'valid' or 'invalid'.
Actions: Validate the elliptic curve public key as follows:

1. Check that $Q \neq O$.
2. If T represents elliptic curve domain parameters over $\mathbf{F}_p$, check that x_Q and y_Q are integers in the range $[1, p - 1]$, and that:
$$y_Q^2 = x_Q^3 + a.x_Q + b \pmod{p}.$$
3. If T represents elliptic curve domain parameters over $\mathbf{F}_{2^m}$, check that x_Q and y_Q are binary polynomials of degree at most $m - 1$, and that:
$$y_Q^2 + x_Q.y_Q = x_Q^3 + a.x_Q^2 + b \text{ in } \mathbf{F}_{2^m},$$
4. Check that $nQ = O$.
5. If any of the checks fail, output 'invalid', otherwise output 'valid'.

3.2.3 Partial Validation of Elliptic Curve Public Keys

There are four acceptable methods for an entity U to receive an assurance that an elliptic curve public key is partially valid. Only one of the methods must be supplied, although in many cases greater security may be obtained by carrying out more than one of the methods.
The four acceptable methods are:

1. U performs partial validation of the elliptic curve public key itself using the public key partial validation primitive described in Section 3.2.3.1.

2. U generates the elliptic curve public key itself using a trusted system.
3. U receives assurance in an authentic manner that a party trusted with respect to U's use of the elliptic curve public key has performed partial validation of the public key using the public key partial validation primitive described in Section 3.2.3.1.
4. U receives assurance in an authentic manner that a party trusted with respect to U's use of the elliptic curve public key generated the public key using a trusted system.

3.2.3.1 Elliptic Curve Public Key Partial Validation Primitive

Input: Valid elliptic curve domain parameters $T = (p, a, b, G, n, h)$ or $(m, f(x), a, b, G, n, h)$, and an elliptic curve public key $Q = (x_Q, y_Q)$ associated with T.
Output: An indication of whether the elliptic curve public key is partially valid or not — either 'valid' or 'invalid'.
Actions: Partially validate the elliptic curve public key as follows:

1. Check that $Q \neq O$.
2. If T represents elliptic curve domain parameters over $\mathbf{F}_p$, check that x_Q and y_Q are integers in the range $[1, p - 1]$, and that:
$$y_Q^2 = x_Q^3 + a.x_Q + b \pmod{p}.$$
3. If T represents elliptic curve domain parameters over $\mathbf{F}_{2^m}$, check that x_Q and y_Q are binary polynomials of degree at most $m - 1$, and that:
$$y_Q^2 + x_Q.y_Q = x_Q^3 + a.x_Q^2 + b \text{ in } \mathbf{F}_{2^m}.$$
4. If any of the checks fail, output 'invalid', otherwise output 'valid'.

3.3 Elliptic Curve Diffie-Hellman Primitives

3.3.1 Elliptic Curve Diffie-Hellman Primitive

Input: The elliptic curve Diffie-Hellman primitive takes as input:

1. Valid elliptic curve domain parameters $T = (p, a, b, G, n, h)$ or $(m, f(x), a, b, G, n, h)$.
2. An elliptic curve private key d_U associated with T owned by U.
3. An elliptic curve public key Q_V associated with T purportedly owned by V.

The public key Q_V should be valid.
Output: A shared secret field element z, or 'invalid'.

Actions: Calculate a shared secret value as follows:

1. Compute the elliptic curve point $P = (x_P, y_P) = d_U Q_V$.
2. Check that $P \neq O$. If $P = O$, output 'invalid' and stop.
3. Output $z = xP$ as the shared secret field element.

3.3.2 Elliptic Curve Cofactor Diffie-Hellman Primitive

Input: The elliptic curve cofactor Diffie-Hellman primitive takes as input:

1. Valid elliptic curve domain parameters $T = (p, a, b, G, n, h)$ or $(m, f(x), a, b, G, n, h)$.
2. An elliptic curve private key d_U associated with T owned by U.
3. An elliptic curve public key Q_V associated with T purportedly owned by V.
4. The public key Q_V should at a minimum be partially valid.

Output: A shared secret field element z, or 'invalid'.

Actions: Calculate a shared secret value as follows:

1. Compute the elliptic curve point $P = (x_P, y_P) = d_U Q_V$
2. Check that $P \neq O$. If $P = O$, output 'invalid' and stop.
3. Output $z = xP$ as the shared secret field element.

Key Derivation Primitives

3.5 Hash Functions

Setup: Select one of the approved hash functions. Let *Hash* denote the hash function chosen, *hashlen* denote the length in octets of hash values computed using *Hash*, and *hashmaxlen* denote the maximum length in octets of messages that can be hashed using *Hash*.

Input: The input to the hash function is an octet string *M*.

Output: The hash value *H* which is an octet string of length *hashlen* octets, or 'invalid'.

Actions: Calculate the hash value *H* as follows:

1. Check that *M* is less than *hashmaxlen* octets long — i.e. check that:
$$\| M \| < hashmaxlen.$$
If $\| M \| \geqq hashmaxlen$, output 'invalid' and stop.
2. Convert *M* to a bit string $\overline{M}$ using the conversion routine specified in Section 2.3.2.
3. Calculate the hash value $\overline{H}$ corresponding to $\overline{M}$ using the selected hash function:
$$\overline{H} = Hash(\overline{M}).$$
4. Convert $\overline{H}$ to an octet string *H* using the conversion routine specified in Section 2.3.1.
5. Output *H*.

3.6 Key Derivation Functions

3.6.1 ANSI X9.63 Key Derivation Function

Setup: Select one of the approved hash functions listed in Section 3.5. Let *Hash* denote the hash function chosen, *hashlen* denote the length in octets of hash values computed using *Hash*, and *hashmaxlen* denote the maximum length in octets of messages that can be hashed using *Hash*.

Input: The input to the key derivation function is:

1. An octet string *Z* which is the shared secret value.
2. An integer *keydatalen* which is the length in octets of the keying data to be generated.
3. (Optional) An octet string *SharedInfo* which consists of some data shared by the entities intended to share the shared secret value *Z*.

Output: The keying data K which is an octet string of length *keydatalen* octets, or 'invalid'.

Actions: Calculate the keying data K as follows:

1. Check that $\| Z \| + \| SharedInfo \| + 4 < hashmaxlen$. If $\| Z \| + \| SharedInfo \| + 4 \geqq hashmaxlen$, output 'invalid' and stop.
2. Check that $keydatalen < hashlen \times (2^{32} - 1)$. If $keydatalen \geqq hashlen \times (2^{32} - 1)$, output 'invalid' and stop.
3. Initiate a 4 octet, big-endian octet string *Counter* as 00000001_{16} .
4. For $i = 1$ to $\lceil keydatalen / hashlen \rceil$, do the following:
   4.1. Compute:
   $$K_i = Hash(Z \| Counter \| [SharedInfo])$$
   using the selected hash function from the list of approved hash functions in Section 3.5.
   4.2. Increment *Counter*.
   4.3. Increment i.
5. Set K to be the leftmost *keydatalen* octets of:
   $$K_1 \| K_2 \| \ldots \| K_{\lceil keydatalen / hashlen \rceil}$$.
6. Output K.

Elliptic Curve Diffie-Hellman Scheme

6.1 Elliptic Curve Diffie-Hellman Scheme

6.1.1 Scheme Setup

1. U and V should establish which of the key derivation functions supported in Section 3.6 to use, and select any options involved in the operation of the key derivation function. Let KDF denote the key derivation function chosen.
2. U and V should establish whether to use the 'standard' elliptic curve Diffie-Hellman primitive specified in Section 3.3.1, or the elliptic curve cofactor Diffie-Hellman primitive specified in Section 3.3.2.
3. U and V should establish at the desired security level elliptic curve domain parameters $T = (p, a, b, G, n, h)$ or $(m, f(x), a, b, G, n, h)$. The elliptic curve domain parameters T should be generated using the primitive specified in Section 3.1.1.1 or the primitive specified in Section 3.1.2.1. Both U and V should receive an assurance that the elliptic curve domain parameters T are valid using one of the methods specified in Section 3.1.1.2 or Section 3.1.2.2.

6.1.2 Key Deployment

1. U should establish an elliptic curve key pair (d_U, Q_U) associated with the elliptic curve domain parameters T established during the setup procedure. The key pair should be generated using the primitive specified in Section 3.2.1.
2. V should establish an elliptic curve key pair (d_V, Q_V) associated with the elliptic curve domain parameters T established during the setup procedure. The key pair should be generated using the primitive specified in Section 3.2.1.
3. U and V should exchange their public keys Q_U and Q_V.
4. If the 'standard' elliptic curve Diffie-Hellman primitive is being used, U should receive an assurance that Q_V is valid using one of the methods specified in Section 3.2.2, and if the elliptic curve cofactor Diffie-Hellman primitive is being used, U should receive an assurance that Q_V is at least partially valid using one of the methods specified in Section 3.2.2 or Section 3.2.3.5.
5. If the 'standard' elliptic curve Diffie-Hellman primitive is being used, V should receive an assurance that Q_U is valid using one of the methods specified in Section 3.2.2, and if the elliptic curve cofactor Diffie-Hellman primitive is being

used, V should receive an assurance that Q_U is at least partially valid using one of the methods specified in Section 3.2.2 or Section 3.2.3.

6.1.3 Key Agreement Operation

U and V should perform the key agreement operation described in this section to establish keying data using the elliptic curve Diffie-Hellman scheme. For clarity U's use of the operation is described. V's use of the operation is analogous, but with the roles of U and V reversed. U should establish keying data with V using the keys and parameters established during the setup procedure and the key deployment procedure as follows:

Input: The input to the key agreement operation is:

1. An integer *keydatalen* which is the number of octets of keying data required.
2. (Optional) An octet string *SharedInfo* which consists of some data shared by U and V.

Output: An octet string K which is the keying data of length *keydatalen* octets, or 'invalid'.

Actions: Establish keying data as follows:

1. Use one of the Diffie-Hellman primitives specified in Section 3.3 to derive a shared secret field element $z \in \mathbf{F}_q$ from U's secret key d_U established during the key deployment procedure and V's public key Q_V obtained during the key deployment procedure. If the Diffie-Hellman primitive outputs 'invalid', output 'invalid' and stop. Decide whether to use the 'standard' elliptic curve Diffie-Hellman primitive or the elliptic curve cofactor Diffie-Hellman primitive according to the convention established during the setup procedure.
2. Convert $z \in \mathbf{F}_q$ to an octet string Z using the conversion routine specified in Section 2.3.5.
3. Use the key derivation function *KDF* established during the setup procedure to generate keying data K of length *keydatalen* octets from Z and [*SharedInfo*]. If the key derivation function outputs 'invalid', output 'invalid' and stop.
4. Output K.