

ソフトウェアライブラリ間の暗号技術相互運用性に関する調査

荒巻 佑樹¹ 金岡 晃¹

概要：開発者にとって暗号技術関連部分をアプリケーションに導入する際、容易に実装することが難しいことが指摘されている。この問題を解決するには、ライブラリが提供する API のユーザビリティ向上と各言語でのライブラリごとにデータフォーマットを揃えることが重要となる。API のユーザビリティ向上においてはサンプルコードが充実し、マニュアルもわかりやすくするといった工夫により改善されつつあるが、各言語でのライブラリごとの暗号の入力・出力時の各データフォーマットを揃えているのかという点では、詳細に調査されていない。それを本研究では、各言語でのライブラリごとの暗号実装時の入力・出力のデータフォーマットを調査することで相互運用が可能かを検証する。

Survey on Cryptographic Technology Interoperability between Software Libraries

YUKI ARAMAKI¹ AKIRA KANAOKA¹

1. はじめに

ソフトウェア開発者がさまざまなソフトウェアを開発するにあたり、通信が発生するソフトウェアや情報を保管するソフトウェア、利用者の情報を扱うソフトウェアなどは、セキュリティ技術を導入してそれらの情報を必要に応じて守らなければならない。暗号技術はそこで利用される代表的な技術であるが暗号技術を適切にソフトウェア開発者が利用することの難しさが指摘されている [1], [2], [3], [4]。

この問題を解決するには、暗号技術がソフトウェアライブラリとして提供されることに加え、そのライブラリのユーザビリティ向上と各言語とライブラリ間でのデータフォーマットの共通化などの相互運用性の確保が重要となる。ライブラリとその API のユーザビリティ向上においては、サンプルコードの充実とマニュアルの可読性の高さの確保などの工夫により改善が図られてきている [2]。しかし、各言語でのライブラリごとの暗号の入力・出力時の各データフォーマットを揃えているかの点は十分に調査がされていない。また、各ライブラリごとのデータフォーマット調査の前の段階である、各言語での代表的な暗号技術ライブラリの包括的な調査と利用可能な暗号技術の調査も十分にされているとは言い難い。

そこで本研究では、データフォーマット共通化といった相互運用性の確保に向けて、最初に各言語での代表的な暗号技術ライブラリの洗い出しを行い、次にそれぞれの暗号

技術ライブラリで利用可能な暗号技術を仕様面から調査し、その整理を行う。さらに、それらの内容を基にデータフォーマットで利用されているフォーマットの種類について考察をし、それらに起きうる問題点を指摘する。

調査として対象となる言語を 10 個選び、それぞれの言語における暗号技術ライブラリを調査し、リストアップした。またそれぞれの暗号技術ライブラリの詳細を仕様を基に調査し、利用可能な方式などを整理して、表として示した。その結果、いくつかの新たな視点が得られた。まず、各暗号技術ライブラリでの暗号方式の表記法の不統一であった。これにより送受信側で同一と思い込みつつ違う内容の暗号文が交わされる可能性を指摘した。同様に、パディングについても設定ミスが発生しうる可能性を指摘した。さらに、それらの内容を基にデータフォーマットで利用されているフォーマットの種類について考察をし、それらに起きうる問題点を指摘した。

今後は、これらの問題点の現実的なリスクを実際のテスト実装を基に調査をし明確化することが求められるだろう。また、今後の暗号技術ライブラリのために、相互運用テストが可能なテストデータ群を揃えてテストスイートを構築することもこういった相互運用性向上の一助となろう。

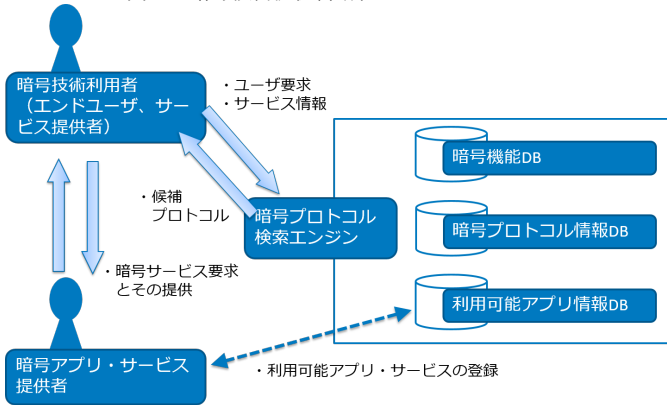
2. 関連研究

2.1 暗号技術提供自動化フレームワーク

金岡により暗号技術の提供を自動化するためのフレームワークが提案された (図 1)[5]。金岡の研究では、提供者側は開発時に安全性を検討することなく開発可能であること

¹ 東邦大学
Toho University

図 1 暗号技術提供自動化フレームワーク



や、利用者側も自身で安全性の確認をしなくて良いなど双方に利点があるとされている。提案されたフレームワークは、「暗号プロトコルの記述言語」「提案言語で記述されたプロトコルのデータベース」「暗号プロトコル検索エンジン」の3つのコンポーネントから構成されている。

2.2 暗号プロトコルの調査

前島らの研究 [6] では、暗号プロトコルの自動適用に必要なフレームワークの一部である暗号機能データベースと暗号プロトコル情報データベースに着目した。暗号機能データベースは、プロトコルの表現のために暗号プロトコルが提供しうる機能を細分化した情報が提供される役割で、暗号プロトコル情報データベースは、研究者により提案されてきた暗号プロトコルや、標準化がすすんでいる暗号プロトコルが網羅的に含まれるデータベースのことである。そして、暗号機能データベースと暗号プロトコルデータベースと利用可能アプリ情報データベースをまとめたものを「暗号プロトコル大辞典」と呼び前島らの研究ではその実現を目的とした。暗号プロトコル大辞典の実現のために、書籍を中心に暗号プロトコルを調査した。ここでの暗号プロトコルとは共通鍵暗号や公開鍵暗号、ハッシュ関数、デジタル署名などがある。また各プログラミング言語で書かれた暗号ライブラリで利用可能なプロトコルがあるか調査した。そして調査に基づき、暗号プロトコルを分類し、分類に基づきプロトコルを整理し、暗号プロトコル大辞典の基礎を作り上げた。

2.3 暗号技術の効果的な利用

暗号技術の適切な利用の難しさと、それ故に発生するリスクなどについてはいくつかの研究がされてきた [1], [2], [3], [4]。最近ではそれらを踏まえ、開発時に補助をする方法が提案されている。Nguyen らによるアプローチは統合開発環境のプラグインとしてサポートをする機能の提案となっている [7]。彼らの手法では、暗号だけではなく TLS の設定などいくつかの視点からの実装サポー

トがされている。暗号に関していえば、AES のパディングや共通鍵暗号への初期ベクトル設定など、限定された用途に適用が可能となっている。

Ma らの研究では、暗号技術を適用するときに通して発生される欠陥に注目し、それらに対し固定のパッチを準備して対応する方法を提案した [8]。そこで対象とされている暗号技術は共通鍵暗号でのモード利用や初期ベクトルの設定、鍵の保護や乱数の選択など、暗号技術としては重要であるが基本的な部分が対象となっており、さまざまな暗号技術全般を対象としたものではなかった。

3. 調査方法

本研究では、データフォーマット共通化といった相互運用性の確保に向けて、最初に各言語での代表的な暗号技術ライブラリの洗い出しを行い、次にそれぞれの暗号技術ライブラリで利用可能な暗号技術を仕様面から調査し、その整理を行う。さらに、それらの内容を基にデータフォーマットで利用されているフォーマットの種類について考察をし、それらに起きうる問題点を指摘する。

3.1 各言語における暗号技術ライブラリの調査

最初の調査では、暗号化・復号やデジタル署名、ハッシュ関数といった暗号プロトコルに対し、オープンソースや無料利用など広く利用可能なライブラリを調査し、ライブラリリストを作成した。調査対象としたプログラミング言語は Java、JavaScript、C、C++、C#、Ruby、Perl、Python、PHP、Visual Basic .NET の 10 言語とした。

暗号技術ライブラリの洗い出しでは、Web の検索を用い、「暗号」「ライブラリ」というキーワードに各言語名称を加えて検索を行い、結果に表れたライブラリ候補をまずリストアップした。検索に利用したキーワードはそのほかにも「Cryptographic」「Crypto」「Library」を用いた。その後、ライブラリ候補の公式 Web ページを調査し、公式 Web ページやそれに類するページがあった場合にまずそれが暗号技術ライブラリかどうかの判断を行い、調査対象のライブラリとしての採用を判定した。

3.2 ドキュメントからのライブラリの対応暗号技術調査

広く利用されているライブラリの提供は、Web 上で行われている。そのライブラリは本体だけでなく利用可能な機能やその利用法、ライブラリ自体の導入方法が記載されたドキュメントが合わせて整備されている。そのため、ライブラリで利用可能な暗号技術とデータフォーマットを調査するために、ドキュメントの記載内容を調査することとした。ドキュメントとは、そのライブラリの説明書であり、特徴や関数や対応暗号技術等が書かれている。参照例の crypto-js のドキュメント (図??) では、ハッシュ関数や暗号の入力・出力や対応技術暗号や OpenSSL との相互運

表 1 今回調査したライブラリの内訳

言語	ライブラリ個数
Java	17 個
JavaScript	9 個
C	9 個
C++	6 個
C#	4 個
Ruby	3 個
Perl	2 個
Python	3 個
PHP	5 個
Visual Basic	2 個

用性、エンコーダーの記載がされている。

暗号技術は、その名称利用が固定化されていない方式が数多く存在する。たとえば、米国の標準技術暗号である AES や DES はほぼ名称がこれらで固定されている一方で、DES を 3 段にして使うトリプル DES 方式に関しては TripleDES と記載するケースや 3DES と記載するケース、DESede と記載するケースが混在している。また RC4 については開発当時に存在した米国の輸出規制に影響を受け、正式名称ではなく機能が同等で名目上別名称になっている Arcfour といった表記がされている。また、使用上の記載として暗号技術の鍵長も名称に含めているケースも存在する。たとえば AES では鍵のサイズを 128 ビット、192 ビット、256 ビットなど変更することができる。ライブラリによっては、利用暗号を設定したあとに別パラメータで鍵長を設定するものもあれば、利用暗号の設定時に「AES256」として 256 ビット鍵の AES を指定させるものもある。こういった鍵長の面でも名称表記の揺れが発生している。

本調査では、著者の知識をもとに可能な限りこういった表記の揺れは統一化した。

3.3 データフォーマット調査

暗号ライブラリであるライブラリのドキュメントから暗号実装時の入力・出力のデータフォーマットの型を調査しリスト化した。

4. 調査結果

4.1 各言語の暗号技術ライブラリの洗い出し

表 1 はその結果得られた調査対象ライブラリの個数である。ただし、Java 言語ではライブラリだけではなくプロバイダも含んだ数となっている。

Java 言語の調査結果については表 2,3,4,5,6、JavaScript 言語の調査結果は表 7, 8, 9, 10、C 言語の調査結果は表 11, 12, 14, 13、C++言語の調査結果は表 15, 16, 17, 18、C#言語の調査結果は表 19, 20, 21, 22、Ruby 言語と Perl 言語の調査結果は表 23, 24, 25, 26、Perl 言語の調査結果は表 ??, ??, ??, ??, Python 言語の調査結果は表 27, 28, 29, 30,

PHP 言語の調査結果は表 31, 32, 33, 34、VisualBasic 言語の調査結果は表 35, 36, 37, 38, にそれぞれ示す。

4.2 調査結果の考察

いずれの言語にも暗号技術ライブラリが存在することが今回の調査で判明した。また、多くの言語で代表的な暗号ライブラリが存在し、そのライブラリにより高い網羅性の実現されていることがわかった。特に OpenSSL は C 言語が基礎となっているものの、各言語への変換が準備されており、各言語での暗号技術利用に大きな影響を与えていることが見える。

利用されている暗号技術については、公開鍵暗号系であれば RSA や DSA の利用、共通鍵暗号系であれば AES、ハッシュ関数での SHA-1 または SHA-256、MAC における HMAC といった、標準技術がまず搭載されている。これは利用者のニーズを考えると当然と言えよう。一方で、分野に特化して多くの方式を充実させてあるライブラリや、今後重要視されている chacha20 や Poly1305 などの方式を積極的に採用しているライブラリなど、それぞれのライブラリの性格をうかがわせるものとなった。

表記法についてはそれぞれのライブラリ間で違いがあり、この部分も相互運用性に関わる点であることが今回明らかになった。

AES や RSA といった暗号方式では、入力データが規定サイズと揃わない場合に残りのデータ部分を埋める手法である Padding (パディング) が行われる。厳密にはこのパディングにも仕様があり、このパディングも送受信側で一致しなければ暗号の適切な利用とはならない。ライブラリによってはこのパディング手法が設定できないものや、こういったパディングが初期設定されているかの情報が確認できないものもあり、パディングの部分が相互運用にかかわる大きな問題点になる可能性が浮き彫りになった。

4.3 データフォーマットに関する考察

データフォーマットについては、ライブラリ間での差異が大きく出る結果となった。OpenSSL などではそれぞれの暗号化データや鍵データの出力方法を指定するパラメータや API が明確に存在していた一方で、存在していないものも多く存在していた。それらの場合は暗号技術ライブラリではなく一般の API により文字列出力するなどの対応がされるようであった。これらのことから、暗号技術ライブラリの開発組織でのデータフォーマットの扱いに大きな差があり、ここでも相互運用の大きな問題が発生する可能性が見て取れた。

5. まとめ

本論文では、代表的なプログラミング言語のライブラリの対応技術暗号と入力・出力のデータフォーマットをラ

イブラリごとのドキュメントで調査することで、ソフトウェアライブラリ間の暗号技術相互運用性を目指した。

代表的なプログラミング言語のライブラリの多くは、サードパーティーによって開発・提供されている。サードパーティー側は、利用可能な機能や方法等をドキュメントに記載し、利用者側が使いやすい工夫がなされているが、

本研究では、暗号技術ライブラリの利用におけるデータフォーマット共通化といった相互運用性の確保に向けて、最初に各言語での代表的な暗号技術ライブラリの洗い出しを行い、次にそれぞれの暗号技術ライブラリで利用可能な暗号技術を仕様面から調査し、その整理を行った。対象となる言語を10個選び、それぞれの言語における暗号技術ライブラリを調査し、リストアップした。またそれぞれの暗号技術ライブラリの詳細を仕様を基に調査し、利用可能な方式などを整理して、表として示した。その結果、いくつかの新たな視点が得られた。まず、各暗号技術ライブラリでの暗号方式の表記法の不統一であった。これにより送受信側で同一と思い込みつつ違う内容の暗号文が交わされる可能性を指摘した。同様に、パディングについても設定ミスが発生しうる可能性を指摘した。さらに、それらの内容を基にデータフォーマットで利用されているフォーマットの種類について考察をし、それらに起きうる問題点を指摘した。

今後は、これらの問題点の現実的なリスクを実際のテスト実装を基に調査をし明確化することが求められるだろう。また、今後の暗号技術ライブラリのために、相互運用テストが可能なテストデータ群を揃えてテストスイートを構築することもこういった相互運用性向上の一助となろう。

参考文献

- [1] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek and S. Fahl, "Developers Need Support, Too: A Survey of Security Advice for Software Developers," 2017 IEEE Cybersecurity Development (SecDev), Cambridge, MA, 2017, pp. 22-26.
- [2] Y. Acar et al., "Comparing the Usability of Cryptographic APIs," 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2017, pp. 154-171.
- [3] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek and C. Stransky, "How Internet Resources Might Be Helping You Develop Faster but Less Securely," in IEEE Security & Privacy, vol. 15, no. 2, pp. 50-60, March-April 2017.
- [4] Y. Acar, S. Fahl and M. L. Mazurek, "You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users," 2016 IEEE Cybersecurity Development (SecDev), Boston, MA, 2016, pp. 3-8.
- [5] 金岡 晃, "サービスへの高機能暗号の適用を容易にするフレームワークと暗号プロトコル記述言語", 2014 年度
- [6] 前島 涼太、金岡 晃, "暗号プロトコル辞典：暗号技術のユーザビリティ向上に向けた暗号プロトコルデータベースの検討", マルチメディア、分散、協調とモバイル (DICOMO2017) シンポジウム、2017
- [7] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. 2017. A Stitch in Time: Supporting Android Developers in WritingSecure Code. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)
- [8] Siqi Ma, David Lo, Teng Li, and Robert H. Deng. 2016. CDRep: Automatic Repair of Cryptographic Misuses in Android Applications. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
SunPKCS11 プロバイダ			
SUN プロバイダ			
SunRsaSign プロバイダ			
SunJSSE プロバイダ			
SunJCE プロバイダ	AES,AESWrp, Blowfish, DES, DESede,RC2, DESedeWrap, PBEWithMD5AndDES, PBEWithMD5AndTripleDES, PBEWithSHA1AndDESede, PBEWithSHA1AndRC2_40, PBEWithSHA1AndRC2_128, PBEWithSHA1AndRC4_40, PBEWithSHA1AndRC4_128, PBEWithHmacSHA1AndAES_128, PBEWithHmacSHA1AndAES_128, PBEWithHmacSHA224AndAES_128, PBEWithHmacSHA256AndAES_128, PBEWithHmacSHA384AndAES_128, PBEWithHmacSHA512AndAES_128, PBEWithHmacSHA1AndAES_256, PBEWithHmacSHA224AndAES_256, PBEWithHmacSHA256AndAES_256, PBEWithHmacSHA384AndAES_256, PBEWithHmacSHA512AndAES_256	RC4	ECB, CBC, PCBC, CTR, CTS, CFB, OFBB
SunMSCAPI プロバイダ			

表 2 Java 言語ライブラリにおける各暗号方式の対応状況<共通鍵暗号> (1)

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
SunEC プロバイダ			
OracleUcrypto プロバイダ	AES		ECB, CBC, CTR, GCM, CFB
BouncyCastle プロバイダ	AES, Blowfish, Camellia, CAST5, CAST6, DES, TripleDES, DSTU7624, GOST, IDEA, Noekeon, RC5, RC6, Rijndael, SEED, Shacal2, Serpent, Skipjack, SM4, TEA, Threefish, Twofish, XTEA, HC128, ChaCha, Salsa20, ISAAC, VMPC, Grain	RC4	CBC, CFB, GCF, EAX, OCB, OFB, SIC, KCTR, GOFB, CCM, GCM, KCCM
FlexiProvider CoreProvider	TripleDES, IDEA, RC2, RC5, RC6, MARS, Serpent, Twofish, Rijndael, SAFER		ECB, CBC, CFB, OFB, CTR
FlexiProvider ECProvider			
FlexiProvider PQCPProvider			
FlexiProvider NFPProvider			
GNU Crypto	Anubis, Blowfish, CAST5, DES, Khazad, Rijnael, Serpent, Square, TripleDES, Twofish		CBC, CFB, CTR, ECB, ICM, OFB
Cryptix	AES, Blowfish, CAST5, DES, IDEA, MARS, RC2, RC6, Rijndael, SKIPJACK, Serpent, Square, TripleDES, Twofish	RC4,	
CroccryptLib			
JPBC			

表 3 Java 言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞（2）

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
SunPKCS11 プロバイダ			
SUN プロバイダ			DSA
SunRsaSign プロバイダ			RSA
SunJSSE プロバイダ			RSA
SunJCE プロバイダ	RSA		
SunMSCAPI プロバイダ	RSA		RSA
SumEC プロバイダ		ECDH	ECDSA
OracleUcrypto プロバイダ	RSA		RSA
BouncyCastle プロバイダ	RSA, ElGamal, NTRU		
FlexiProvider CoreProvider	RSA, RSA-OAEP, ElGamal		DSA, RSA
FlexiProvider ECPProvider	ECIES	ECDH	ECDSA, ECNR
FlexiProvider PQCPProvider			MerkleOTS, CoronadoOTS, NiederreiterCFS, NTRUSign
FlexiProvider NFPProvider			IQRDSA, IQDSA, IQGQ
GNU Crypto		DH, SRP-6	DSA, RSA
Cryptix			
CrococryptLib	RSASSA-OAEP, RSASSA-PKCS1		
JPBC			

表 4 Java 言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
SunPKCS11 プロバイダ			
SUN プロバイダ	MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		
SunRsaSign プロバイダ			
SunJSSE プロバイダ			
SunJCE プロバイダ		HMAC	
SunMSCAPI プロバイダ			
SunEC プロバイダ			
OracleUcrypto プロバイダ	MD5, SHA-1, SHA-256, SHA-384, SHA-512		
BouncyCastle プロバイダ	Blake2 DSTU7564, Keccak, MD2, MD4, MD5, RipeMD, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3, SHAKE, Skein, SM3, Tiger, GOST3411, Whirlpool,	Poly1305, CMAC, GMAC, HMAC	
FlexiProvider CoreProvider	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD4, MD5, RIPEMD, Tiger, DHA, FORK	CBC, CMAC, HMAC, TT,	
FlexiProvider ECProvider			
FlexiProvider PQCProvider			
FlexiProvider NFPProvider			
GNU Crypto	Haval, MD2, MD4, MD5, RipeMD, SHA-1, SHA-256, SHA-384, SHA-512, Tiger, Whirlpool,	HMAC, TMMH16	
Cryptix			
CrocoencryptLib			ABE, HIBE, InnerProduct, Anonymous HIBE, HVE, SS, IBS, KEM
JPBC			

表 5 Java 言語ライブラリにおける各暗号方式の対応状況<ハッシュ関数、その他>

ライブラリ名称	備考	URL
SunPKCS11 プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
SUN プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
SunRsaSign プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
SunJSSE プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
SunJCE プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
SunMSCAPI プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
SunEC プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
OracleUcrypto プロバイダ		https://docs.oracle.com/javase/1.8/docs/technotes/guides/security/SunProviders.html
BouncyCastle プロバイダ		https://bouncycastle.org/specifications.html
FlexiProvider CoreProvider		https://www.flexiprvider.de/javadoc/flexiprvider/docs/de/flexiprvider/core/FlexiCoreProvider.html
FlexiProvider ECPProvider		https://www.flexiprvider.de/javadoc/flexiprvider/docs/de/flexiprvider/ec/package-summary.html
FlexiProvider PQCPProvider		https://www.flexiprvider.de/javadoc/flexiprvider/docs/de/flexiprvider/pqc/FlexiPQCProvider.html
FlexiProvider NFPProvider		https://www.flexiprvider.de/javadoc/flexiprvider/docs/de/flexiprvider/nf/FlexiNFPProvider.html
GNU Crypto		https://www.gnu.org/software/gnu-crypto/manual/api/index.html
Cryptix		http://www.cryptix.org/
CroccryptLib		http://www.frankhissen.de/croccryptlib-home-en-frank-hissen-it-software.html
JPBC		http://gas.dia.unisa.it/projects/jpb/#.Wm_p4Kh9hF

表 6 Java 言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
Stanford Javascript Crypto Library	AES		CBC, CCM, CTR, GCM, OCB 2.0
js-nacl		NaCl crypto_secretbox (xsalsa20-poly1305), NaCl crypto_stream	
tweetnacl-js		NaCl crypto_secretbox (xsalsa20-poly1305)	
jsbn			
JSEncrypt			
jsencryptolib	AES, DES, 3DES		ECB, CBC, CTR
crypto-js	AES, DES, 3DES	Rabbit, RC4, RC4Drop	CBC, CFB, CTR, OFB, ECB
jsSHA			
Clippers JavaScript Crypto Library	AES		

表 7 JavaScript 言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

6. 議論

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
Stanford Javascript Crypto Library			
js-nacl	NaCl crypto_box (x25519-xsalsa20-poly1305)		NaCl crypto_sign (Ed25519)
tweetnacl-js	NaCl crypto_box (x25519-xsalsa20-poly1305)		NaCl crypto_sign (Ed25519)
jsbn	RSA		
JSEncrypt	RSA		
jsencryptolib	RSA, SA, ECC, CPK, IBE		
crypto-js			
jsSHA			
Clippers JavaScript Crypto Library			

表 8 JavaScript 言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
Stanford Javascript Crypto Library	RIPMD, SHA-1, SHA-256, SHA-512	HMAC	
js-nacl	SHA-512	NaCl crypto_onetimeauth, NaCl crypto_auth	
tweetnacl-js	SHA-512		
jsbn			
JSencrypt			
jsencryptolib	SHA-1, SHA-256	HMAC, CBCMAC, CMAC	
crypto-js	MD5, SHA-1, SHA-256	HMAC, CBCMAC, CMAC	PKCS7, ISO97971, AnsiX923, ISO10126, ZeroPadding, NoPadding
jsSHA	SHA-1, SHA-224, SHA3-224, SHA-256, SHA3-256, SHA-384, SHA3-384, SHA-512, SHA3-512, SHAKE128, SHAKE256		
Clippers JavaScript Crypto Library	SHA-256		

表 9 JavaScript 言語ライブラリにおける各暗号方式の対応状況＜ハッシュ関数、その他＞

ライブラリ名称	備考	URL
Stanford Javascript Crypto Library		https://crypto.stanford.edu/sjcl/
js-nacl		https://github.com/tonyg/js-nacl
tweetnacl-js		https://github.com/dchest/tweetnacl-js/blob/master/README.md#documentation
jsbn		http://www-cs-students.stanford.edu/~tjw/jsbn/
JSencrypt		http://travistidwell.com/jsencrypt/
jsencryptolib		https://code.google.com/archive/p/jsencryptolib/
crypto-js		https://code.google.com/archive/p/crypto-js/
jsSHA		https://caligatio.github.io/jsSHA/
Clippers JavaScript Crypto Library		https://github.com/clipperez/javascript-crypto-library

表 10 JavaScript 言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
OpenSSL libcrypto	RC2, DES, TripleDES, IDEA, blowfish, CAST5, RC5, AES, ARIA, Camellia, SEED, SM4,	chacha20, chacha20_poly1305	ECB, CBC, CFB, OFB, CTR, GCM, CCM, XTS, WRAP, OCB
NSS (Network Security Service)	RC2, AES, TripleDES, DES	RC4	
cryptlib	AES, Blowfish, CAST-128, DES, Triple DES, IDEA, RC2, RC5, Skipjack	RC4	
Libgcrypt	IDEA, TripleDES, CAST5, Blowfish, AES, Twofish, SERPENT, RC2, Camellia, GOST28147-89	RC4, Salsa20, chacha20	ECB, CFB, CBC, OFB, CTR, CCM, GCM, OCB, XTS,
Nettle	AES, RC2, Blowfish, Camellia, CAST128, DES, TripleDES, Serpent, Twofish	RC4, ChaCha, salsa20	CBC, CTR, CFB
wolfCrypt	AES, DES, TripleDES, Camellia	RC4, RABBIT, HC-128, ChaCha	CBC, CTR, GCM, CCM-8, CTR, GCM
BeeCrypt	AES, Blowfish		
NaCl		NaCl crypto_secretbox (xsalsa20-poly1305), NaCl crypto_stream	
Sodium	AES	XSalsa20-poly1305, ChaCha20, XChaCha20, Salsa20, XSalsa20	GCM

表 11 C 言語ライブラリにおける各暗号方式の対応状況<共通鍵暗号>

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
OpenSSL libcrypto	RSA, RSA-PSS, X25519	DH, ECDH	RSA, DSA, ECDSA
NSS (Network Security Service)	RSA	DH, ECDH	RSA, DSA, ECDSA
cryptlib	RSA, Elgamal	DH, ECDH	RSA, DSA, ECDSA
Libgcrypt	RSA, Elgamal		RSA, DSA
Nettle	RSA		RSA, DSA, ECDSA, EdDSA
wolfCrypt	RSA	DH, EDH	DSA
BeeCrypt	DHAES, RSA	DH	DSA, ElGamal, RSA
NaCl	NaCl crypto.box (x25519-xsalsa20-poly1305)		NaCl crypto.sign (Ed25519)
Sodium	x25519-xsalsa20-poly1305	X25519	Ed25519

表 12 C 言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	備考	URL
OpenSSL libcrypto		https://wiki.openssl.org/index.php/Libcrypto_API
NSS (Network Security Service)		https://nss-crypto.org/
cryptlib		http://www.cryptlib.com/security-software/cryptlib-api
Libgcrypt		https://www.gnupg.org/documentation/manuals/gcrypt/
Nettle		https://www.lysator.liu.se/nisse/nettle/
wolfCrypt		https://www.wolfssl.com/products/wolfcrypt/
BeeCrypt		http://beecrypt.sourceforge.net/doxygen/c/index.html
NaCl		http://nacl.cr.yp.to/features.html
Sodium		https://download.libsodium.org/doc/

表 13 C 言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ハッシュ関数	MAC	その他
OpenSSL libcrypto	MD2, MD4, MD5, BLAKE2, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, MDC2, RIPEMD, Whirlpool	HMAC, CMAC	PKCS7, ISO7816-4, ANSI923, ISO10126, ZeroPadding
NSS (Network Security Service)	SHA-1, SHA-256, SHA-384, SHA-512, MD2, MD5	HMAC	
cryptlib	MD2, MD4, MD5, RIPEMD-160, SHA-1, SHA-256	HMAC	
Libgcrypt	SHA-1, RIPEMD, MD5, MD4, MD2, TIGER, HAVAL, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, CRC32, Whirlpool, GOST R 34.11-94, GOST R 34.11-2012, BLAKE2	HMAC, CMAC, GMAC, Poly1305	
Nettle	MD5, MD2, MD4, RIPEMD160, SHA-1, GOSTHASH94, SHA-256, SHA-224, SHA-512, SHA-384, SHA3-224, SHA3-256, SHA3-384, SHA3-512	HMAC, UMAC, Poly1305	
wolfCrypt	MD5, MD4, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, BLAKE2, RIPEMD-160	HMAC, GMAC, Poly1305	
BeeCrypt	MD5, RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	HMAC	
NaCl	SHA-512	NaCl crypto_onetimeauth, NaCl crypto_auth	
Sodium	BLAKE2, SipHash, SHA-256, SHA-512	HMAC	

表 14 C 言語ライブラリにおける各暗号方式の対応状況<ハッシュ関数、その他>

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
OpenSSL libcrypto	RC2, DES, TripleDES, IDEA, blowfish, CAST5, RC5, AES, ARIA, Camellia, SEED, SM4,	chacha20, chacha20_poly1305	ECB, CBC, CFB, OFB, CTR, GCM, CCM, XTS, WRAP, OCB
NSS (Network Security Service)	RC2, AES, TripleDES, DES	RC4	
Crypto++ Library	AES, RC6, MARS, Twofish, Serpent, CAST-256, ARIA, IDEA, Blowfish, Triple-DES, Camellia, SEED, Kalyna, RC5, SIMON, SPECK, SM4, Threefish, Skipjack, SHACAL-2, TEA, XTEA, DES, SEAL, RC2, SAFER, GOST, SHARK	ChaCha, Panama, Sosemanuk, Salsa20, XSalsa20, RC4	ECB, CBC, CBC, CTS, CFB, OFB, CTR, GCM, CCM, EAX
Botan	AES, ARIA, Blowfish, Camellia, CAST-128, CAST-256, DES/3DES, GOST, IDEA, KASUMI, Lion, MISTY1, Noekeon, SEED, Serpent, SHACAL2, SM4, Threefish-512, Twofish, XTEA	chacha20, chacha20_poly1305, Salsa20, XSalsa20, SHAKE-128, RC4	EAX, OCB, GCM, SIV, CCM, CTR, CBC, XTS, CFB, OFB
Nettle	AES, RC2, Blowfish, Camellia, CAST128, DES, TripleDES, Serpent, Twofish	RC4, ChaCha, salsa20	CBC, CTR, CFB
NaCl		NaCl crypto_secretbox (xsalsa20-poly1305), NaCl crypto_stream	

表 15 C++言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
OpenSSL libcrypto	RSA, RSA-PSS, X25519	DH, ECDH	RSA, DSA, ECDSA
NSS (Network Security Service)	RSA	DH, ECDH	RSA, DSA, ECDSA
Crypto++ Library	RSA, ElGamal, Rabin-Williams (RW), LUC, ECIES	DH, MQV, HMQV, FHMQV, LUCDIF, XTR-DH, ECDH, ECMQV	RSA, DSA, ElGamal, Nyberg-Rueppel (NR), ECGDSA, LUCCELG, DLIES, ESIGN, ECDSA, ECNR
Botan	RSA, ElGamal	DH, ECDH, McEliece, NewHope	ECDSA, DSA, Ed25519, ECGDSA, ECKCDSA, SM2, GOST 34.10-2001, XMSS
Nettle	RSA		RSA, DSA, ECDSA, EdDSA
NaCl	NaCl crypto_box (x25519-xsalsa20-poly1305)		NaCl crypto_sign (Ed25519)

表 16 C++言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
OpenSSL libcrypto	MD2, MD4, MD5, BLAKE2, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, MDC2, RIPEMD, Whirlpool	HMAC, CMAC	PKCS7, ISO7816-4, ANSI923, ISO10126, ZeroPadding
NSS (Network Security Service)	SHA-1, SHA-256, SHA-384, SHA-512, MD2, MD5	HMAC	
Crypto++ Library	BLAKE2, Keccak, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, Poly1305, SipHash, Tiger, RIPEMD, SM3, Whirlpool, MD2, MD4, MD5, Panama Hash	BLAKE2, CMAC, CBC, DMAC, GMAC, HMAC, Poly1305, SipHash, VMAC, Two-Track-MAC	
Botan	SHA-1, SHA-2, SHA-3, RIPEMD-160, Skein-512, BLAKE2b, SM3, Tiger, Whirlpool, GOST 34.11, MD5, MD4	HMAC, CMAC, Poly1305, SipHash, GMAC, CBC-MAC, X9.19 DES-MAC	
Nettle	MD5, MD2, MD4, RIPEMD160, SHA-1, GOSTHASH94, SHA-256, SHA-224, SHA-512, SHA-384, SHA3-224, SHA3-256, SHA3-384, SHA3-512	HMAC, UMAC, Poly1305	
NaCl	SHA-512	NaCl crypto_onetimeauth, NaCl crypto_auth	

表 17 C++言語ライブラリにおける各暗号方式の対応状況<ハッシュ関数、その他>

ライブラリ名称	備考	URL
OpenSSL libcrypto		https://wiki.openssl.org/index.php/Libcrypto_API
NSS (Network Security Service)		https://nss-crypto.org/
Crypto++ Library		https://www.cryptopp.com/
Botan		https://botan.randombit.net/
Nettle		https://www.lysator.liu.se/nisse/nettle/
NaCl		http://nacl.cr.yp.to/features.html

表 18 C++言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
CryptoSys PKI	TriplesDES, AES		
CryptoSYS API	AES, DES, TriplesDES, Blowfish	chacha20_poly1305, RC4, Salsa20, ChaCha20	GCM
elixis	AES, TriplesDES		
Microsoft CNG	AES, DES, RC2, TriplesDES		

表 19 C#言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
CryptoSys PKI	RSA, RSA-OAEP		RSA, ECDSA
CryptoSYS API			
elixis			
Microsoft CNG		ECDH	DSA, RSA, ECDSA

表 20 C#言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
CryptoSys PKI	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD5, RIPEMD	HMAC	
CryptoSYS API	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD5, RIPEMD	HMAC, CMAC, Poly1305	
elixis	MD5		
Microsoft CNG	MD5, RIPEMD160, SHA-1, SHA-256, SHA-384, SHA-512	HMAC	

表 21 C#言語ライブラリにおける各暗号方式の対応状況＜ハッシュ関数、その他＞

ライブラリ名称	備考	URL
CryptoSys PKI		https://www.cryptosys.net/
CryptoSYS API		https://www.cryptosys.net/
elixis		https://code.google.com/archive/p/elixis/
Microsoft CNG		https://docs.microsoft.com/en-us/dotnet/standard/security/cryptographic-services

表 22 C#言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

言語	ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
Ruby	OpenSSL libcrypto	RC2, DES, TripleDES, IDEA, blowfish, CAST5, RC5, AES, ARIA, Camellia, SEED, SM4,	chacha20, chacha20_poly1305	ECB, CBC, CFB, OFB, CTR, GCM, CCM, XTS, WRAP, OCB
Ruby	NaCl		NaCl crypto_secretbox (xsalsa20-poly1305), NaCl crypto_stream	
Ruby	EzCrypto	AES		
Perl	OpenSSL libcrypto	RC2, DES, TripleDES, IDEA, blowfish, CAST5, RC5, AES, ARIA, Camellia, SEED, SM4,	chacha20, chacha20_poly1305	ECB, CBC, CFB, OFB, CTR, GCM, CCM, XTS, WRAP, OCB
Perl	OpenPGP	IDEA, DES3, Blowfish, Rijndael, Twofish, CAST5		

表 23 Ruby 言語と Perl 言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

言語	ライブラリ名称	公開鍵暗号	鍵交換	電子署名
Ruby	OpenSSL libcrypto	RSA, RSA-PSS, X25519	DH, ECDH	RSA, DSA, ECDSA
Ruby	NaCl	NaCl crypto_box (x25519-xsalsa20-poly1305)		NaCl crypto_sign (Ed25519)
Ruby	EzCrypto			
Perl	OpenSSL libcrypto	RSA, RSA-PSS, X25519	DH, ECDH	RSA, DSA, ECDSA
Perl	OpenPGP			RSA, DSA, ElGamal

表 24 Ruby 言語と Perl 言語ライブラリにおける各暗号方式の対応状況<公開鍵暗号>

言語	ライブラリ名称	ハッシュ関数	MAC	その他
Ruby	OpenSSL libcrypto		HMAC, CMAC	PKCS7, ISO7816_4, ANSI923, ISO10126, ZeroPadding
Ruby	NaCl		NaCl crypto_onetimeauth, crypto_auth	
Ruby	EzCrypto			
Perl	OpenSSL libcrypto	MD2, MD4, MD5, BLAKE2, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, MDC2, RIPEMD, Whirlpool	HMAC, CMAC	PKCS7, ISO7816_4, ANSI923, ISO10126, ZeroPadding
Perl	OpenPGP	MD5, SHA-1, RIPEMD160		

表 25 Ruby 言語と Perl 言語ライブラリにおける各暗号方式の対応状況<ハッシュ関数、その他>

言語	ライブラリ名称	備考	URL
Ruby	OpenSSL libcrypto		https://ruby-doc.org/stdlib-2.4.0/libdoc/openssl/rdoc/OpenSSL/Cipher.html
Ruby	NaCl		http://nacl.cr.yp.to/features.html
Ruby	EzCrypto		https://github.com/pelle/ezcrypto
Perl	OpenSSL libcrypto		
Perl	OpenPGP		

表 26 Ruby 言語と Perl 言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
pycrypto	AES, DES, RC2, Blowfish, CAST, DES, TripleDES,	RC4	
m2crypto Botan	AES, ARIA, Blowfish, Camellia, CAST-128, CAST-256, DES/3DES, GOST, IDEA, KASUMI, Lion, MISTY1, Noekeon, SEED, Serpent, SHACAL2, SM4, Threefish-512, Twofish, XTEA	RC4 chacha20, chacha20_poly1305, Salsa20, XSalsa20, SHAKE-128, RC4	EAX, OCB, GCM, SIV, CCM, CTR, CBC, XTS, CFB, OFB

表 27 Python 言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
pycrypto	RSA-OAEP, RSA		RSA
m2crypto	RSA	DH, ECDH	RSA, DSA, ECDSA
Botan	RSA, ElGamal	DH, ECDH, McEliece, NewHope	ECDSA, DSA, Ed25519, ECGDSA, ECKCDSA, SM2, GOST 34.10-2001, XMSS

表 28 Python 言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
pycrypto	MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD4, RIPEMD	HMAC	
m2crypto		HMAC	
Botan	SHA-1, SHA-2, SHA-3, RIPEMD-160, Skein-512, BLAKE2b, SM3, Tiger, Whirlpool, GOST 34.11, MD5, MD4	HMAC, CMAC, Poly1305, SipHash, GMAC, CBC-MAC, X9.19 DES-MAC	

表 29 Python 言語ライブラリにおける各暗号方式の対応状況＜ハッシュ関数、その他＞

ライブラリ名称	備考	URL
pycrypto		https://pypi.python.org/pypi/pycrypto
m2crypto		https://gitlab.com/m2crypto/m2crypto
Botan		https://botan.randombit.net/

表 30 Python 言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
OpenSSL libcrypto	RC2, DES, TripleDES, IDEA, blowfish, CAST5, RC5, AES, ARIA, Camellia, SEED, SM4,	chacha20, chacha20-poly1305	ECB, CBC, CFB, OFB, CTR, GCM, CCM, XTS, WRAP, OCB
Sodium	AES	XSalsa20-poly1305, ChaCha20, XChaCha20, Salsa20, XSalsa20	GCM
phplib	AES, Blowfish, DES, RC2, Rijndael, TripleDES, Twofish	RC4	
phpCrypt	AES, Blowfish, CAST, DES, TripleDES, RC2, Rijndael, Skipjack	RC4, Enigma, Vigenere	CBC, CFB, CTR, ECB, NCFB, NOFB, OFB, PCBC
Mcrypt	TripleDES, Blowfish, CAST, CRYPT, DES, GOST, IDEA, LOKI97, MARS, PANAMA, Rijndael, RC2, RC6, SAFER, Serpent, Skipjack, Tean, Threeway, Twofish, Wake, XTEA	Enigma, RC4	CBC, CFB, ECB, OFB

表 31 PHP 言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
OpenSSL libcrypto	RSA, RSA-PSS, X25519	DH, ECDH	RSA, DSA, ECDSA
Sodium	x25519-xsalsa20-poly1305	X25519	Ed25519
phpseclib	RSA		RSA
phpCrypt			
Mcrypt			

表 32 PHP 言語ライブラリにおける各暗号方式の対応状況＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
OpenSSL libcrypto	MD2, MD4, MD5, BLAKE2, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, MDC2, RIPEMD, Whirlpool	HMAC, CMAC	PKCS7, ISO7816-4, ANSI923, ISO10126, ZeroPadding
Sodium	BLAKE2, SipHash, SHA-256, SHA-512	HMAC	
phpseclib	MD5, MD2, SHA-1, SHA-256, SHA-512	HMAC	
phpCrypt			ANSI X.923, ZeroPadding, ISO10126, PKCS7, ISO7816-4
Mcrypt			

表 33 PHP 言語ライブラリにおける各暗号方式の対応状況＜ハッシュ関数、その他＞

ライブラリ名称	備考	URL
OpenSSL libcrypto		https://wiki.openssl.org/index.php/Libcrypto_API
Sodium		https://download.libsodium.org/doc/
phpseclib		https://github.com/phpseclib/phpseclib
phpCrypt		http://www.gilfether.com/phpcrypt/
Mcrypt		http://php.net/manual/ja/book.mcrypt.php

表 34 PHP 言語ライブラリにおける各暗号方式の対応状況＜備考、URL＞

ライブラリ名称	ブロック暗号	ストリーム暗号	暗号利用モード
Microsoft CNG	AES, DES, RC2, TripleDES		
Devv.Core.Crypto	RC2, DES, TripleDES, AES		

表 35 Visual Basic 言語ライブラリにおける各暗号方式の対応状況＜共通鍵暗号＞

ライブラリ名称	公開鍵暗号	鍵交換	電子署名
Microsoft CNG		ECDH	DSA, RSA, ECDSA
Devv.Core.Crypto			

表 36 Visual Basic 言語ライブラリにおける各暗号方式の対応状況
＜公開鍵暗号＞

ライブラリ名称	ハッシュ関数	MAC	その他
Microsoft CNG	MD5, RIPEMD160, SHA-1, SHA-256, SHA-384, SHA-512	HMAC	
Devv.Core.Crypto	MD5,		

表 37 Visual Basic 言語ライブラリにおける各暗号方式の対応状況
＜ハッシュ関数、その他＞

ライブラリ名称	備考	URL
Microsoft CNG		https://docs.microsoft.com/en-us/dotnet/standard/security/cryptographic-services
Devv.Core.Crypto		https://archive.codeplex.com/?p=crypto

表 38 Visual Basic 言語ライブラリにおける各暗号方式の対応状況
 ＜備考、URL＞