情報通信システム論b

RPC **リモートコンピュータ上の機能を呼び出す
データ構造の例**

**XML-RPC**

<?xml version="1.0" encoding="utf-8"?>

<methodCall>

　　<methodName>groups.removeAgentFromGroupRole</methodName>

　　<params><param><value><struct>

 <member><name>ReadKey</name><value><string>JOG6809</string></value></member>

 <member><name>WriteKey</name><value><string>JOG8086</string></value></member>

　　　……………………….

　　</struct></value></param></params>
</methodCall>

 **JSON-RPC**

{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}

ASN.1
**ASN.1** (Abstract Syntax Notation One，抽象構文記法１)

通信データのデータ構造（プロトコル）の記述の使用される．

例）

Point ::= SEQUENCE {

 　x INTEGER,

 　y INTEGER

}

AlgorithmIdentifer ::= SEQUENCE {

 Algorithm OBJECT IDENTIFIER,

 DomainParameters

}

DomainParameters ::= SEQUENCE {

 p INTEGER, -- odd prime

 g INTEGER, -- generator

 q INTEGER, -- factor of p-1

}

DER

**ASN.1での具体的な符号化（バイナリ化）方法**

* BER (Basic Encoding Rules) 曖昧性がある．
* **DER** (Distinguished Encoding Rules，BERのサブセット版)

**DER** の構造

 識別子（タグ） ＋ コンテンツ長 ＋ コンテンツ　（型＋長さ＋値）　の繰り返し
 　コンテンツの中に DER を入れ子できる．

　　<https://letsencrypt.org/ja/docs/a-warm-welcome-to-asn1-and-der/>

識別子（1Byte: 8bit）

**全体 8bit (2 + 1 + 5 Bit)**

* **2bit: Class**
	+ 00 (0x00): UNIVERSAL（汎用）
	+ 01 (0x40): APPLICATION（応用）
	+ 10 (0x80): Context-Specific（コンテキスト特定）
	+ 11 (0xc0): PRIVATE（私用）
* **1bit**
	+ 0 (0x00): Primitive（基本型）
	+ 1 (0x20): Constructed（構造型．入れ子構造を取る）
* **5bit** タグ（後に続くデータの型）
	+ 0 + 0 0010 (0x02): 整数（INTEGER）
	+ 0 + 0 0011 (0x03): bit列（BIT STRING）
	+ 0 + 0 0100 (0x04): 8bit文字列（OCTET STRING）
	+ 0 + 0 0110 (0x06): オブジェクト（OBJECT IDENTIFER）
	+ 0 + 0 1100 (0x0c): UTF8文字列（UTF8String）
	+ 0 + 1 0000 (0x10): シーケンス（SEQUENCE）
	+ 0 + 1 0001 (0x11): セット（SET）
	+ **1 + 1 0000 (0x30): 構造型シーケンス（SEQUENCE） (**Constructed**)**
	+ 1 + 1 0001 (0x31): 構造型セット（SET） **(**Constructed**)**

ASN.1とDERの例(1)

**Point ::= SEQUENCE {**

 **x INTEGER,**

 **y INTEGER**

**}**

**30 06 02 01 05 02 01 67**

**6Byteの構造型（入れ子）
　　内部のデータは 02 01 05 と 02 01 67, 02 は整数データ**

ⅹ　0x05 = 5

y 0x67 = 103

ASN.1とDERの例(2)

**Message ::= SEQUENCE {**

 **no INTEGER,**

 **msg OCTET STRING**

**}**

**30 10 02 01 10 04 0b 48 65 6c 6c 6f 20 57 6f 72 6c 64**

**16Byteの構造型（入れ子）
　　内部のデータは 02 01 10 と 04 0b 48 … 04 は8bit文字列**

no　　　0x10 = 16

msg 　“Hello World

ASN.1とDERの例(3)

**LDAP**

**30 5b 02 01 02 60 37 02 01 03 04 28 63 6e 3d 6c 0 [ . . . ` 7 . . . . ( c n = l**

**64 61 70 61 75 74 68 2c 63 6e 3d 75 73 65 72 73 d a p a u t h , c n = u s e r s**

**2c 64 63 3d 65 64 75 74 75 69 73 2c 64 63 3d 6c , d c = e d u t u i s , d c = l**

**6f 63 61 6c 80 08 \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* a0 1d o c a l . . . . . . . . . . . .**

**30 1b 04 19 31 2e 33 2e 36 2e 31 2e 34 2e 31 2e 0 . . . 1 . 3 . 6 . 1 . 4 . 1 .**

**34 32 2e 32 2e 32 37 2e 38 2e 35 2e 31 4 2 . 2 . 2 7 . 8 . 5 . 1**

 SEQUENCE (91Byte)

 　INTEGER (1Byte) 2 // MessageID

 APPLICATION Constructed 0x60[0x40+0x20] (55)

 INTEGER (1Byte) 3 // version

 OCTET STRING (40Byte) "cn=ldapauth,cn=users,dc=edutuis,dc=local"

 Context-Specific Primitive 0x80[0x80+0x00] (8Byte) パスワード \*\*\*\*\*\*…

 Context-Specific Constructed 0xa0[0x80+0x20] (29Byte)

 SEQUENCE (27Byte)

 OCTET STRING (25Byte) "1.3.6.1.4.1.42.2.27.8.5.1"

ASN.1とDERの例(4)

Diffie-Hellman鍵交換

SubjectPublicKeyInfo ::= SEQUENCE {

 algorithm AlgorithmIdentifier,

 subjectPublicKey BIT STRING

}

AlgorithmIdentifer ::= SEQUENCE {

 Algorithm OBJECT IDENTIFIER,

 DomainParameters

}

DomainParameters ::= SEQUENCE {

 p INTEGER, -- odd prime, p = q + 1

 g INTEGER, -- generator, g

 q INTEGER, -- factor of p - 1

}

ASN.1とDERの例(4)

**30 81 9e 30 57 06 09 2a 86 48 86 f7 0d 01 03 01**

**30 4a 02 41 00 be 53 20 60 b6 9f 17 ae f2 47 32**

**87 25 0d 24 42 d3 95 40 48 82 09 d8 f7 af 07 bf**

**fc 6f b8 eb 5f 9a 8c e2 eb 65 97 30 99 38 73 6b**

**0c 28 c1 4a c7 4d 86 45 11 3b c5 2a cc a2 cd 3d**

**f0 bb 06 41 d3 02 01 02 02 02 01 ff 03 43 00 02**

**40 4b 51 d6 00 27 fd e1 72 5e 51 10 36 6f 27 21**

**0c a2 cc a7 c6 3c 76 62 6e 95 7b 1b a0 ed d8 16**

**16 0b 2c f6 41 87 2c 76 91 84 35 13 9a 7d 22 8a**

**38 84 17 09 7c 7c a1 40 27 8a 4f d8 dd c3 1e 0f**

**05**

**アルゴリズム:**  **2a 86 48 86 f7 0d 01 03 01**

**Pキー:**

 **00 be 53 20 60 b6 9f 17 ae f2 47 32**

**87 25 0d 24 42 d3 95 40 48 82 09 d8 f7 af 07 bf**

**fc 6f b8 eb 5f 9a 8c e2 eb 65 97 30 99 38 73 6b**

**0c 28 c1 4a c7 4d 86 45 11 3b c5 2a cc a2 cd 3d**

**f0 bb 06 41 d3**

**Gキー: 02**

**Seed(長さ0): 00**

**Yキー:**

 **4b 51 d6 00 27 fd e1 72 5e 51 10 36 6f 27 21**

**0c a2 cc a7 c6 3c 76 62 6e 95 7b 1b a0 ed d8 16**

**16 0b 2c f6 41 87 2c 76 91 84 35 13 9a 7d 22 8a**

**38 84 17 09 7c 7c a1 40 27 8a 4f d8 dd c3 1e 0f**

**05**

SEQUENCE

　　SEQUENCE

　　　　OBJECT　 AlgorithmIdentifer

　　　　SEQUENCE

　　　　　　INTEGER　P Key

　　　　　　INTEGER　G Key

　　　　　　INTEGER　PKEY長 – 1

　　BIT STRING　subjectPublicKey　（seed + Y Key)

アプリケーション独自

例） Second Life/ OpenSimulator

<https://polaris.star-dust.jp/pukiwiki/?Second+Life/Protocol>

例） TLS

<https://polaris.star-dust.jp/pukiwiki/?TLS>