Workgroup: Network Working Group
Internet-Draft: draft-pkcs5-gost-03
Published: 21 March 2022
Intended Status: Informational
Expires: 22 September 2022
Authors: E.K. Karelina, Ed.
InfoTeCS

Generating Password-based Keys Using the GOST Algorithms

Abstract

This document specifies how to use the Password-Based Cryptography Specification version 2.1 (PKCS #5) defined in [RFC8018] to generate password- based keys in conjunction with the Russian national standard GOST algorithms.

PKCS #5 applies a pseudorandom function (a cryptographic hash, cipher, or HMAC) to the input password along with a salt value and repeats the process many times to produce a derived key.

This specification is developed outside the IETF and is published to facilitate interoperable implementations that wish to support the GOST algorithms. This document does not imply IETF endorsement of the cryptographic algorithms used in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document supplements [RFC8018]. It provides a specification of usage of GOST R 34.12-2015 encryption algorithms and the GOST R 34.11-2012 hashing functions in the information systems [GostPkcs5]. The methods described in this document are designed to generate key information using the user's password and protect information using the generated keys.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in

BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Basic Terms and Definitions

Throughout this document, the following notations are used:

P	a password encoded as a Unicode UTF-8 string
S	a random initializing value
c	a number of iterations of algorithm, a positive integer
dkLen	a length in octets of derived key, a positive integer
DK	a derived key of length dkLen
B_n	a set of all octet strings of length n, n >= 0; if n = 0, then the set B_n consists of an empty string of length 0
A||C	a concatenation of two octet strings A, C, i.e., a vector from B_(|A|+|C|), where the left subvector from B_(|A|) is equal to the vector A and the right subvector from B_(|C|) is equal to the vector C: A = (a_(n_1),...,a_1) in B_(n_1) and C = (c_(n_2),..., c_1) in B_(n_2), res = (a_(n_1),...,a_1,c_(n_2),..., c_1) in B_(n_1 + n_2);
\xor	a bit-wise exclusive-or of two octet strings of the same length
MSB^n_r: B_n -> B_r	a truncating of an octet string to size r by removing the least significant n-r octets: MSB^n_r(a_n,...,a_(n-r+1),a_(n-r),...,a_1) =(a_n,...,a_(n-r+1));
LSB^n_r: B_n -> B_r	a truncating of a octet string to size r by removing the most significant n-r octets: LSB^n_r(a_n,...,a_(n-r+1),a_(n-r),...,a_1) =(a_r,...,a_1)
Int(i)	a four-octet encoding of the integer i =< 2^32: (i_1, i_2, i_3, i_4) in B_4, i = i_1 + 2^8 * i_2 + 2^16 * i_3 + 2^24 * i_4
b[i, j]	a substring extraction operator: extracts octets i through j, 0 =< i =< j.
CEIL(x)	the smallest integer greater than, or equal to, x

Table 1

This document uses the following abbreviations and symbols:

HMAC_GOSTR3411	Hashed-based Message Authentication Code. A function for calculating a message authentication code, based on the GOST R 34.11-2012 hash function ([RFC6986]) with 512-bit output in accordance with [RFC2104].

Table 2

4. Algorithm For Generating a Key From a Password

The DK key is calculated by means of a key derivation function PBKDF2(P, S, c, dkLen) [RFC8018], section 5.2 using the HMAC_GOSTR3411 function as the PRF pseudo-random function:

```
DK = PBKDF2(P,S,c,dkLen).
```

The PBKDF2 function is defined as the following algorithm:

1. If dkLen > (2^32 - 1) * 64, output "derived key too long" and stop.

2. Calculate n = CEIL(dkLen / 64).

3. Calculate a set of values for each i from 1 to n:

```
U_1(i) = HMAC_GOSTR3411 (P, S || INT (i))

U_2(i) = HMAC_GOSTR3411 (P, U_1(i))

...

U_c(i) = HMAC_GOSTR3411 (P, U_{c-1}(i))

T(i) = U_1(i) \xor U_2(i) \xor ... \xor U_c(i)
```

4. Concatenate the octet strings T(i) and extract the first dkLen octets to produce a derived key DK:

```
DK = MSB^{n * 64}_dkLen(T(1)||T(2)||...||T(n))
```

5. Data Encryption

5.1. GOST R 34.12-2015 Data Encryption

Data encryption using the DK key is carried out in accordance with the PBES2 scheme (see [RFC8018], section 6.2) using GOST R 34.12-2015 in CTR_ACPKM mode (see [RFC8645]).

5.1.1. Encryption

The encryption process for PBES2 consists of the following steps:

1. Select the random value S of length from 8 to 32 octets.

2. Select the iteration count c depending on the conditions of use. The minimum allowable value for the parameter is 1000.

3. Set the value dkLen = 32.

4. Apply the key derivation function to the password P, the random value S and the iteration count c to produce a derived key DK of length dkLen octets in accordance with the algorithm from Section 4. Generate the sequence T(1) and truncate it to 32 octets, i.e.,

   DK = PBKDF2(P,S,c,32) = MSB^64_32(T(1)).

5. Generate the random value ukm of size n, where n takes a value of 12 or 16 octets, depending on the selected encryption algorithm:

   GOST R 34.12-2015 "Kuznyechik" n = 16 (see [RFC7801])

   GOST R 34.12-2015 "Magma" n = 12 (see [RFC8891])

6. Set the value S' = ukm[1..n-8]

7. For id-gostr3412-2015-magma-ctracpkm and id-gostr3412-2015-kuznyechik-ctracpkm algorithms (see Appendix A.3) encrypt the message M with GOST R 34.12-2015 algorithm with the derived key DK and the random value S' to produce a ciphertext C.

8. For id-gostr3412-2015-magma-ctracpkm-omac and id-gostr3412-2015-kuznyechik-ctracpkm-omac algorithms (see Appendix A.3) encrypt the message M with GOST R 34.12-2015 algorithm with the derived key DK and the ukm in accordance with the following steps:

   - Generate two keys from the derived key DK using the KDF_TREE_GOSTR3411_2012_256 algorithm (see [RFC7836]):

     encryption key K(1)

     MAC key K(2).

   Input parameters for the KDF_TREE_GOSTR3411_2012_256 algorithm take the folowing values:

     K_in = DK

     label = "kdf tree" (8 octets)

     seed = ukm[n-7..n]

     R = 1

   The input string label above is encoded using ASCII.

- Compute MAC for the message M using the K(2) key. Append the computed MAC value to the message M: M||MAC.

- Encrypt the resulting octet string with MAC with GOST R 34.12-2015 algorithm with the derived key K(1) and the random value S' to produce a ciphertext C.

9. Serialize the parameters S, c, ukm as algorithm parameters in accordance with Appendix A.

5.1.2. Decryption

The decryption process for PBES2 consists of the following steps:

1. Set the value dkLen = 32.

2. Apply the key derivation function PBKDF2 to the password P, the random value S and the iteration count c to produce a derived key DK of length dkLen octets in accordance with the algorithm from Section 4. Generate the sequence T(1) and truncate it to 32 octets, i.e., DK = PBKFD2(P,S,c,32) = MSB^64_32(T(1)).

3. Set the value S' = ukm[1..n-8], where n is the size of ukm in octets.

4. For id-gostr3412-2015-magma-ctracpkm and id-gostr3412-2015-kuznyechik-ctracpkm algorithms (see Appendix A.3) decrypt the ciphertext C with GOST R 34.12-2015 algorithm with the derived key DK and the random value S' to produce the message M.

5. For id-gostr3412-2015-magma-ctracpkm-omac and id-gostr3412-2015-kuznyechik-ctracpkm-omac algorithms (see Appendix A.3) decrypt the ciphertext C with GOST R 34.12-2015 algorithm with the derived key DK and the ukm in accordance with the following steps:

   - Generate two keys from the derived key DK using the KDF_TREE_GOSTR3411_2012_256 algorithm:

     encryption key K(1)

     MAC key K(2).

   Input parameters for the KDF_TREE_GOSTR3411_2012_256 algorithm take the folowing values:

     K_in = DK

     label = "kdf tree" (8 octets)

seed = ukm[n-7..n]

R = 1

The input string label above is encoded using ASCII.

- Decrypt the ciphertext C with GOST R 34.12-2015 algorithm with the derived key K(1) and the random value S' to produce the plaintext. The last k octets of the text are the message authentication code MAC', where k depends on the selected encryption algorithm.

- Compute MAC for the text[1..m - k] using the K(2) key, where m is the size of text.

- Compare the original message authentication code MAC and the receiving message authentication code MAC'. If the sizes or values do not match, the message is distorted.

6. Message Authentication

PBMAC1 scheme is used for message authentication (see [RFC8018], section 7.1). This scheme bases on the HMAC_GOSTR3411 function.

6.1. MAC Generation

The MAC generation operation for PBMAC1 consists of the following steps:

1. Select the random value S of length from 8 to 32 octets.

2. Select the iteration count c depending on the conditions of use. The minimum allowable value for the parameter is 1000.

3. Set the dkLen to at least 32 octets. It depends on previous parameter values.

4. Apply the key derivation function to the password P, the random value S and the iteration count c to generate a sequence K of length dkLen octets in accordance with the algorithm from Section 4.

5. Truncate the sequence K to 32 octets to get the derived key DK, i.e., DK = LSB^dkLen_32(K).

6. Process the message M with the underlying message authentication scheme with the derived key DK to generate a message authentication code T.

7. Save the parameters S, c, ukm as algorithm parameters in accordance with Appendix A.

6.2. MAC Verification

The MAC verification operation for PBMAC1 consists of the following steps:

1. Set the dkLen to at least 32 octets. It depends on previous parameter values.

2. Apply the key derivation function to the password P, the random value S and the iteration count c to generate a sequence K of length dkLen octets in accordance with the algorithm from Section 4.

3. Truncate the sequence K to 32 octets to get the derived key DK, i.e., DK = LSB^dkLen_32(K).

4. Process the message M with the underlying message authentication scheme with the derived key DK to generate a message authentication code MAC'.

5. Compare the original message authentication code MAC and the receiving message authentication code MAC'. If the sizes or values do not match, the message is distorted.

7. Security Considerations

This entire document is about security.

For information on security considerations for password-based cryptography see [RFC8018].

Conforming applications MUST use unique values for ukm and S.

It is RECOMMENDED to use the value of parameter c equal to 2000 for generating the derived key in PBKDF2 algorithm.

It is RECOMMENDED to use the value of parameter S equal to 32 octets for generating the derived key in PBKDF2 algorithm.

It is RECOMMENDED to use the exact algorithm parameters in symmetric algorithms "Magma" and "Kuznyechik". They are defined in Appendix A.3.

8. IANA Considerations

This document makes no requests for IANA action.

9. References

9.1. Normative References

[GostPkcs5] Karelina, E., Pianov, S., and A. Davletshina, "Information technology. Cryptographic Data Security. Password-based key security.", R 1323565.1.xxx-2022 (work in progress). Federal Agency on Technical Regulating and Metrology (In Russian).

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6986] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013, <https://www.rfc-editor.org/info/rfc6986>.

[RFC7801] Dolmatov, V., Ed., "GOST R 34.12-2015: Block Cipher "Kuznyechik"", RFC 7801, DOI 10.17487/RFC7801, March 2016, <https://www.rfc-editor.org/info/rfc7801>.

[RFC7836] Smyshlyaev, S., Ed., Alekseev, E., Oshkin, I., Popov, V., Leontiev, S., Podobaev, V., and D. Belyavsky, "Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", RFC 7836, DOI 10.17487/RFC7836, March 2016, <https://www.rfc-editor.org/info/rfc7836>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8645] Smyshlyaev, S., Ed., "Re-keying Mechanisms for Symmetric Keys", RFC 8645, DOI 10.17487/RFC8645, August 2019, <https://www.rfc-editor.org/info/rfc8645>.

[RFC8891] Dolmatov, V., Ed. and D. Baryshkov, "GOST R 34.12-2015: Block Cipher "Magma"", RFC 8891, DOI 10.17487/RFC8891, September 2020, <https://www.rfc-editor.org/info/rfc8891>.

9.2. Informative References

[RFC6070] Josefsson, S., "PKCS #5: Password-Based Key Derivation Function 2 (PBKDF2) Test Vectors", RFC 6070, DOI 10.17487/RFC6070, January 2011, <https://www.rfc-editor.org/info/rfc6070>.

Appendix A. Identifiers and Parameters

This section defines ASN.1 syntax for the key derivation functions, the encryption schemes, the message authentication scheme, and supporting techniques ([RFC8018]).

```
rsadsi OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) 113549 }
pkcs OBJECT IDENTIFIER ::= { rsadsi 1 }
pkcs-5 OBJECT IDENTIFIER ::= { pkcs 5 }
```

A.1. PBKDF2

The object identifier id-PBKDF2 identifies the PBKDF2 key derivation function:

```
id-PBKDF2 OBJECT IDENTIFIER ::= { pkcs-5 12 }
```

The parameters field associated with this OID in an AlgorithmIdentifier SHALL have type PBKDF2-params:

```
PBKDF2-params ::= SEQUENCE
{
    salt            CHOICE
    {
        specified       OCTET STRING,
        otherSource     AlgorithmIdentifier {{PBKDF2-SaltSources}}
    },
    iterationCount  INTEGER (1000..MAX),
    keyLength       INTEGER (32..MAX) OPTIONAL,
    prf             AlgorithmIdentifier {{PBKDF2-PRFs}}
}
```

The fields of type PBKDF2-params have the following meanings:

- salt contains the random value S in OCTET STRING.

- iterationCount specifies the iteration count c.

- keyLength is the length of the derived key in octets. It is optional field for PBES2 sheme since it is always 32 octets. It

MUST be present for PBMAC1 sheme and MUST be at least 32 octets since the HMAC_GOSTR3411 function has a variable key size.

- prf identifies the pseudorandom function. The identifier value MUST be id-tc26-hmac-gost-3411-12-512, the parameters value must be NULL:

```
id-tc26-hmac-gost-3411-12-512 OBJECT IDENTIFIER ::=
{
    iso(1) member-body(2) ru(643) reg7(7)
    tk26(1) algorithms(1) hmac(4) 512(2)
}
```

A.2. PBES2

The object identifier id-PBES2 identifies the PBES2 encryption scheme:

```
id-PBES2 OBJECT IDENTIFIER ::= { pkcs-5 13 }
```

The parameters field associated with this OID in an AlgorithmIdentifier SHALL have type PBES2-params:

```
PBES2-params ::= SEQUENCE
{
    keyDerivationFunc   AlgorithmIdentifier { { PBES2-KDFs } },
    encryptionScheme    AlgorithmIdentifier { { PBES2-Encs } }
}
```

The fields of type PBES2-params have the following meanings:

- keyDerivationFunc identifies the key derivation function in accordance with Appendix A.1.

- encryptionScheme identifies the encryption scheme in with Appendix A.3.

A.3. Identifier and Parameters of Gost34.12-2015 Encryption Scheme

The Gost34.12-2015 encryption algorithm identifier SHALL take one of the following values:

```
id-gostr3412-2015-magma-ctracpkm OBJECT IDENTIFIER ::=
{
    iso(1) member-body(2) ru(643) rosstandart(7)
    tc26(1) algorithms(1) cipher(5)
    gostr3412-2015-magma(1) mode-ctracpkm(1)
}
```

In case of use id-gostr3412-2015-magma-ctracpkm identifier the data is encrypted by the GOST R 34.12-2015 Magma cipher in CTR_ACPKM mode in accordance with [RFC8645]. The block size is 64 bits, the section size is fixed within a specific protocol based on the requirements of the system capacity and the key lifetime.

```
id-gostr3412-2015-magma-ctracpkm-omac OBJECT IDENTIFIER ::=
{
    iso(1) member-body(2) ru(643) rosstandart(7)
    tc26(1) algorithms(1) cipher(5)
    gostr3412-2015-magma(1) mode-ctracpkm-omac(2)
}
```

In case of use id-gostr3412-2015-magma-ctracpkm-omac identifier the data is encrypted by the GOST R 34.12-2015 Magma cipher in CTR_ACPKM mode in accordance with [RFC8645], and MAC is computed by the GOST R 34.12-2015 Magma cipher in MAC mode (MAC size is 64 bits). The block size is 64 bits, the section size is fixed within a specific protocol based on the requirements of the system capacity and the key lifetime.

```
id-gostr3412-2015-kuznyechik-ctracpkm OBJECT IDENTIFIER ::=
{
    iso(1) member-body(2) ru(643) rosstandart(7)
    tc26(1) algorithms(1) cipher(5)
    gostr3412-2015-kuznyechik(2) mode-ctracpkm(1)
}
```

In case of use id-gostr3412-2015-kuznyechik-ctracpkm identifier the data is encrypted by the GOST R 34.12-2015 Kuznyechik cipher in CTR_ACPKM mode in accordance with [RFC8645]. The block size is 128 bits, the section size is fixed within a specific protocol based on the requirements of the system capacity and the key lifetime.

```
id-gostr3412-2015-kuznyechik-ctracpkm-omac OBJECT IDENTIFIER ::=
{
    iso(1) member-body(2) ru(643) rosstandart(7)
    tc26(1) algorithms(1) cipher(5)
    gostr3412-2015-kuznyechik(2) mode-ctracpkm-omac(2)
}
```

In case of use id-gostr3412-2015-kuznyechik-ctracpkm-omac identifier the data is encrypted by the GOST R 34.12-2015 Kuznyechik cipher in CTR_ACPKM mode in accordance with [RFC8645], and MAC is computed by the GOST R 34.12-2015 Kuznyechik cipher in MAC mode (MAC size is 128 bits). The block size is 128 bits, the section size is fixed within a specific protocol based on the requirements of the system capacity and the key lifetime.

The parameters field in an AlgorithmIdentifier SHALL have type Gost3412-15-Encryption-Parameters:

```
Gost3412-15-Encryption-Parameters ::= SEQUENCE
{
    ukm OCTET STRING
}
```

The field of type Gost3412-15-Encryption-Parameters have the following meanings:

- ukm MUST be present and MUST contain n octets. Its value depends on the selected encryption algorithm:

  GOST R 34.12-2015 "Kuznyechik" n = 16 (see [RFC7801])

  GOST R 34.12-2015 "Magma" n = 12 (see [RFC8891])

A.4. PBMAC1

The object identifier id-PBMAC1 identifies the PBMAC1 message authentication scheme:

```
id-PBMAC1 OBJECT IDENTIFIER ::= { pkcs-5 14 }
```

The parameters field associated with this OID in an AlgorithmIdentifier SHALL have type PBMAC1-params:

```
PBMAC1-params ::=  SEQUENCE
{
    keyDerivationFunc AlgorithmIdentifier { { PBMAC1-KDFs } },
    messageAuthScheme AlgorithmIdentifier { { PBMAC1-MACs } }
}
```

The fields of type PBMAC1-params have the following meanings:

- keyDerivationFunc is identifier and parameters of key derivation function in accordance with Appendix A.1

- messageAuthScheme is identifier and parameters of HMAC_GOSTR3411 algorithm.

Appendix B. PBKDF2 HMAC_GOSTR3411 Test Vectors

These test vectors are formed by analogy with test vectors from [RFC6070]. The input strings below are encoded using ASCII. The sequence "\0" (without quotation marks) means a literal ASCII NULL value (1 octet). "DK" refers to the Derived Key.

```
Input:
    P = "password" (8 octets)
    S = "salt" (4 octets)
    c = 1
    dkLen = 64

Output:
    DK = 64 77 0a f7 f7 48 c3 b1 c9 ac 83 1d bc fd 85 c2
         61 11 b3 0a 8a 65 7d dc 30 56 b8 0c a7 3e 04 0d
         28 54 fd 36 81 1f 6d 82 5c c4 ab 66 ec 0a 68 a4
         90 a9 e5 cf 51 56 b3 a2 b7 ee cd db f9 a1 6b 47

Input:
    P = "password" (8 octets)
    S = "salt" (4 octets)
    c = 2
    dkLen = 64

Output:
    DK = 5a 58 5b af df bb 6e 88 30 d6 d6 8a a3 b4 3a c0
         0d 2e 4a eb ce 01 c9 b3 1c 2c ae d5 6f 02 36 d4
         d3 4b 2b 8f bd 2c 4e 89 d5 4d 46 f5 0e 47 d4 5b
         ba c3 01 57 17 43 11 9e 8d 3c 42 ba 66 d3 48 de

Input:
    P = "password" (8 octets)
    S = "salt" (4 octets)
    c = 4096
    dkLen = 64

Output:
    DK = e5 2d eb 9a 2d 2a af f4 e2 ac 9d 47 a4 1f 34 c2
         03 76 59 1c 67 80 7f 04 77 e3 25 49 dc 34 1b c7
         86 7c 09 84 1b 6d 58 e2 9d 03 47 c9 96 30 1d 55
         df 0d 34 e4 7c f6 8f 4e 3c 2c da f1 d9 ab 86 c3

Input:
    P = "password" (8 octets)
    S = "salt" (4 octets)
    c = 16777216
    dkLen = 64

Output:
    DK = 49 e4 84 3b ba 76 e3 00 af e2 4c 4d 23 dc 73 92
         de f1 2f 2c 0e 24 41 72 36 7c d7 0a 89 82 ac 36
         1a db 60 1c 7e 2a 31 4e 8c b7 b1 e9 df 84 0e 36
         ab 56 15 be 5d 74 2b 6c f2 03 fb 55 fd c4 80 71

Input:
    P = "passwordPASSWORDpassword" (24 octets)
```

```
    S = "saltSALTsaltSALTsaltSALTsaltSALTsalt" (36 octets)
    c = 4096
    dkLen = 100

Output:
    DK = b2 d8 f1 24 5f c4 d2 92 74 80 20 57 e4 b5 4e 0a
         07 53 aa 22 fc 53 76 0b 30 1c f0 08 67 9e 58 fe
         4b ee 9a dd ca e9 9b a2 b0 b2 0f 43 1a 9c 5e 50
         f3 95 c8 93 87 d0 94 5a ed ec a6 eb 40 15 df c2
         bd 24 21 ee 9b b7 11 83 ba 88 2c ee bf ef 25 9f
         33 f9 e2 7d c6 17 8c b8 9d c3 74 28 cf 9c c5 2a
         2b aa 2d 3a

Input:
    P = "pass\0word" (9 octets)
    S = "sa\0lt" (5 octets)
    c = 4096
    dkLen = 64

Output:
    DK = 50 df 06 28 85 b6 98 01 a3 c1 02 48 eb 0a 27 ab
         6e 52 2f fe b2 0c 99 1c 66 0f 00 14 75 d7 3a 4e
         16 7f 78 2c 18 e9 7e 92 97 6d 9c 1d 97 08 31 ea
         78 cc b8 79 f6 70 68 cd ac 19 10 74 08 44 e8 30
```

Author's Address

Karelina Ekaterina (editor)
InfoTeCS
2B stroenie 1, ul. Otradnaya
Moscow
127273
Russian Federation

Phone: +7 (495) 737-61-92
Email: Ekaterina.Karelina@infotecs.ru