

Classes

[global](#)[ASN1HEX](#)[Base64x](#)[KEYUTIL](#)[KJUR](#)[KJUR.asn1](#)[KJUR.asn1.ASN1Object](#)[KJUR.asn1.ASN1Util](#)[KJUR.asn1.cades](#)[KJUR.asn1.cades.CADESUtil](#)[KJUR.asn1.cades.CompleteCert](#)[KJUR.asn1.cades.OtherCertT](#)[KJUR.asn1.cades.OtherHash](#)[KJUR.asn1.cades.OtherHashA](#)[KJUR.asn1.cades.SignaturePol](#)[KJUR.asn1.cades.SignatureTiv](#)[KJUR.asn1.cms](#)[KJUR.asn1.cms.Attribute](#)[KJUR.asn1.cms.AttributeList](#)[KJUR.asn1.cms.CMSUtil](#)[KJUR.asn1.cms.ContentInfo](#)[KJUR.asn1.cms.ContentType](#)[KJUR.asn1.cms.EncapsulatedC](#)[KJUR.asn1.cms.IssuerAndSeri](#)[KJUR.asn1.cms.MessageDiges](#)[KJUR.asn1.cms.SignedData](#)[KJUR.asn1.cms.SignerInfo](#)[KJUR.asn1.cms.SigningCertific](#)[KJUR.asn1.cms.SigningCertific](#)[KJUR.asn1.cms.SigningTime](#)[KJUR.asn1.csr](#)[KJUR.asn1.csr.CertificationRe](#)[KJUR.asn1.csr.CertificationRe](#)[KJUR.asn1.csr.CSRUtil](#)[KJUR.asn1.DERAbstractStrin](#)[KJUR.asn1.DERAbstractStruc](#)[KJUR.asn1.DERAbstractTime](#)[KJUR.asn1.DERBitString](#)[KJUR.asn1.DERBoolean](#)[KJUR.asn1.DEREnumerated](#)[KJUR.asn1.DERGeneralizedTiv](#)[KJUR.asn1.DERTIA5String](#)[KJUR.asn1.DERTInteger](#)[KJUR.asn1.DERNull](#)

Class KJUR.crypto.Mac

Mac class which is very similar to java.security.Mac class

Defined in: [crypto-1.1.js](#).

Class Summary

[KJUR.crypto.Mac\(params\)](#)

Mac(Message Authentication Code) class which is very similar to java.security.Mac class
Currently this supports following algorithm and providers combination:

- hmacmd5 - cryptojs
- hmacsha1 - cryptojs
- hmacsha224 - cryptojs
- hmacsha256 - cryptojs
- hmacsha384 - cryptojs
- hmacsha512 - cryptojs

NOTE: HmacSHA224 and HmacSHA384 issue was fixed since jsrsasign 4.1.4.

Method Summary

<static>

[KJUR.crypto.Mac.doFinal\(\)](#)

completes hash calculation and returns hash result

<static>

[KJUR.crypto.Mac.doFinalHex\(hex\)](#)

performs final update on the digest using hexadecimal string, then completes the digest computation

<static>

[KJUR.crypto.Mac.doFinalString\(str\)](#)

performs final update on the digest using string, then completes the digest computation

<static>

[KJUR.crypto.Mac.setPassword\(pass\)](#)

set password for Mac This method will set password for (H)Mac internally.

<static>

[KJUR.crypto.Mac.updateHex\(hex\)](#)

update digest by specified hexadecimal string

<static>

[KJUR.crypto.Mac.updateString\(str\)](#)

update digest by specified string

Class Detail

[KJUR.crypto.Mac\(params\)](#)

Mac(Message Authentication Code) class which is very similar to java.security.Mac class

Currently this supports following algorithm and providers combination:

- hmacmd5 - cryptojs
- hmacsha1 - cryptojs
- hmacsha224 - cryptojs
- hmacsha256 - cryptojs
- hmacsha384 - cryptojs
- hmacsha512 - cryptojs

NOTE: HmacSHA224 and HmacSHA384 issue was fixed since jsrsasign 4.1.4. Please use 'ext/cryptojs-312-core-fix.js' instead of 'core.js' of original CryptoJS to avoid those issue.

NOTE2: Hmac signature bug was fixed in jsrsasign 4.9.0 by providing CryptoJS bug workaround.
Please see [KJUR.crypto.Mac.setPassword](#), how to provide password in various ways in detail.

```
var mac = new KJUR.crypto.Mac({alg: "HmacSHA1", "pass": "pass"});
mac.updateString('aaa')
var macHex = md.doFinal()
```

```
// other password representation
var mac = new KJUR.crypto.Mac({alg: "HmacSHA256", "pass": {"hex": "6161"}});
var mac = new KJUR.crypto.Mac({alg: "HmacSHA256", "pass": {"utf8": "aa"}});
var mac = new KJUR.crypto.Mac({alg: "HmacSHA256", "pass": {"rstr": "\x61\x61"}});
```

[KJUR.asn1.DERNumericString](#)
[KJUR.asn1.DERObjectIdentifier](#)
[KJUR.asn1.DEROctetString](#)
[KJUR.asn1.DERPrintableString](#)
[KJUR.asn1.DERSequence](#)
[KJUR.asn1.DERSet](#)
[KJUR.asn1.DERTaggedObject](#)
[KJUR.asn1.DERTeletexString](#)
[KJUR.asn1.DERUTCTime](#)
[KJUR.asn1.DERUTF8String](#)
[KJUR.asn1.tsp](#)
[KJUR.asn1.tsp.AbstractTSAA](#)
[KJUR.asn1.tsp.Accuracy](#)
[KJUR.asn1.tsp.FixedTSAAData](#)
[KJUR.asn1.tsp.MessageImpri](#)
[KJUR.asn1.tsp.PKIFailureInf](#)
[KJUR.asn1.tsp.PKIFreeText](#)
[KJUR.asn1.tsp.PKIStatus](#)
[KJUR.asn1.tsp.PKIStatusInf](#)
[KJUR.asn1.tsp.SimpleTSAAData](#)
[KJUR.asn1.tsp.TimeStampReq](#)
[KJUR.asn1.tsp.TimeStampRes](#)
[KJUR.asn1.tsp.TSPUtil](#)
[KJUR.asn1.tsp.TSTInfo](#)
[KJUR.asn1.x509](#)
[KJUR.asn1.x509.AlgorithmId](#)
[KJUR.asn1.x509.AttributeType](#)
[KJUR.asn1.x509.AuthorityKey](#)
[KJUR.asn1.x509.BasicConstr](#)
[KJUR.asn1.x509.Certificate](#)
[KJUR.asn1.x509.CRL](#)
[KJUR.asn1.x509.CRLDistrib](#)
[KJUR.asn1.x509.CRLEntry](#)
[KJUR.asn1.x509.DistributionP](#)
[KJUR.asn1.x509.DistributionP](#)
[KJUR.asn1.x509.Extension](#)
[KJUR.asn1.x509.ExtKeyUsage](#)
[KJUR.asn1.x509.GeneralName](#)
[KJUR.asn1.x509.GeneralName](#)
[KJUR.asn1.x509.KeyUsage](#)
[KJUR.asn1.x509.OID](#)
[KJUR.asn1.x509.RDN](#)
[KJUR.asn1.x509.SubjectPublic](#)
[KJUR.asn1.x509.TBSCertifica](#)
[KJUR.asn1.x509.TBSCertList](#)
[KJUR.asn1.x509.Time](#)
[KJUR.asn1.x509.X500Name](#)
[KJUR.asn1.x509.X509Util](#)

```
var mac = new KJUR.crypto.Mac({alg: "HmacSHA256", "pass": {"b64": "MiD2/+...a=="}});  
var mac = new KJUR.crypto.Mac({alg: "HmacSHA256", "pass": {"b64u": "MiD2_-...a"}});
```

Parameters:
{Array} params
parameters for constructor

Method Detail

<static> KJUR.crypto.Mac.doFinal()
completes hash calculation and returns hash result

```
md.digest()
```

<static> KJUR.crypto.Mac.doFinalHex(hex)
performs final update on the digest using hexadecimal string, then completes the digest computation

```
md.digestHex('0f2abd')
```

Parameters:
{String} hex
hexadecimal string to final update

<static> KJUR.crypto.Mac.doFinalString(str)
performs final update on the digest using string, then completes the digest computation

```
md.digestString('aaa')
```

Parameters:
{String} str
string to final update

<static> KJUR.crypto.Mac.setPassword(pass)
set password for Mac This method will set password for (H)Mac internally. Argument 'pass' can be specified as following:

- even length string of 0..9, a..f or A..F: implicitly specified as hexadecimal string
- not above string: implicitly specified as raw string
- {rstr: "\x05\x70"}: explicitly specified as raw string
- {hex: "6570"}: explicitly specified as hexadecimal string
- {utf8: "秘密"}: explicitly specified as UTF8 string
- {b64: "Mi7B..=="}: explicitly specified as Base64 string
- {b64u: "Mi7_-"}: explicitly specified as Base64URL string

It is ***STRONGLY RECOMMENDED*** that explicit representation of password argument to avoid ambiguity. For example string "6161" can mean a string "6161" or a hexadecimal string of "aa" (i.e. \x61\x61).

```
mac = KJUR.crypto.Mac({alg: 'hmacsha256'});  
// set password by implicit raw string  
mac.setPassword("\x05\x70\xba9\x0b");  
mac.setPassword("password");  
// set password by implicit hexadecimal string  
mac.setPassword("6570b90b");  
mac.setPassword("6570B90B");  
// set password by explicit raw string  
mac.setPassword({rstr: "\x05\x70\xba9\x0b"});  
// set password by explicit hexadecimal string  
mac.setPassword({hex: "6570b90b"});  
// set password by explicit utf8 string  
mac.setPassword({utf8: "password/パスワード"});  
// set password by explicit Base64 string  
mac.setPassword({b64: "Mb+c3f/=="});  
// set password by explicit Base64URL string  
mac.setPassword({b64u: "Mb-c3f_"});
```

[KJUR.crypto](#)[KJUR.crypto.DSA](#)[KJUR.crypto.ECDSA](#)[KJUR.crypto.ECParameterDB](#)[KJUR.crypto.Mac](#)[KJUR.crypto.MessageDigest](#)[KJUR.crypto.OID](#)[KJUR.crypto.Signature](#)[KJUR.crypto.Util](#)[KJUR.jws](#)[KJUR.jws.IntDate](#)[KJUR.jws.JWS](#)[KJUR.jws.JWSJS](#)[KJUR.lang.String](#)[PKCS5PKEY](#)[RSAKey](#)[X509](#)**Parameters:**

{Object} pass
password for Mac

Since:

crypto 1.1.7 jsrsasign 4.9.0

<static> KJUR.crypto.Mac.updateHex(hex)
update digest by specified hexadecimal string

```
md.updateHex('Dafe36');
```

Parameters:

{String} hex
hexadecimal string to update

<static> KJUR.crypto.Mac.updateString(str)
update digest by specified string

```
md.updateString('New York');
```

Parameters:

{String} str
string to update
