



# КриптоАРМ ГОСТ

Руководство программиста

F

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E T  
E E E  
E T E  
E T

E

E

E

E

E

E

E

E

F

E

E

T

E

T

E

T

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E E

E

F

F

E

E

E

E

E

E

T

E

E E

E

E

E

F

F

E

E

E

E

E

E

T

E

E

E

E

E

E

ET

T

T

T

T

T

TF

E M

TF

TF

E

E

E



ц г ь н з к к к н г

Е

МЕ

Е

п п к к п п к п 92

г г к ч н ч к ч

—

—

—

—

— Е

— Т Т

— Т Т

— Т F TF

—

— Е F

— F

—

— Е

—

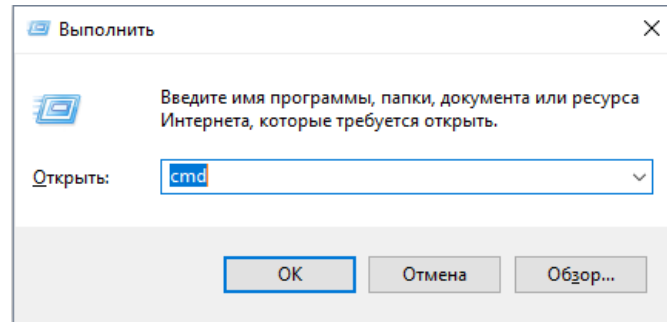




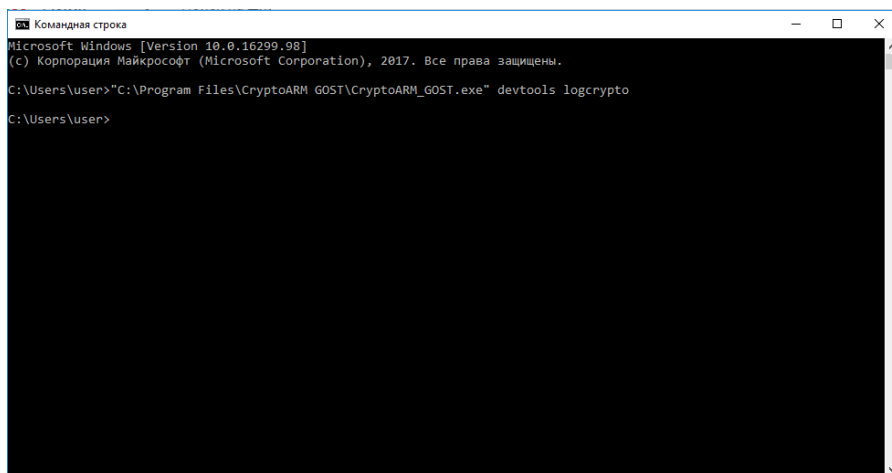
Д н к г \_ н d к d r

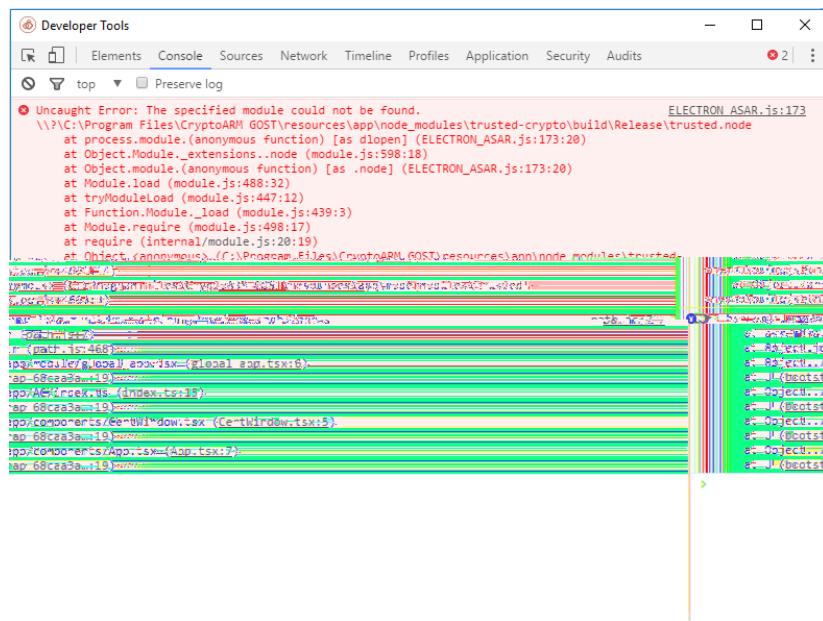
п ф ч гд к п кг R R

T



z dr/ st R / st R d d d st r st



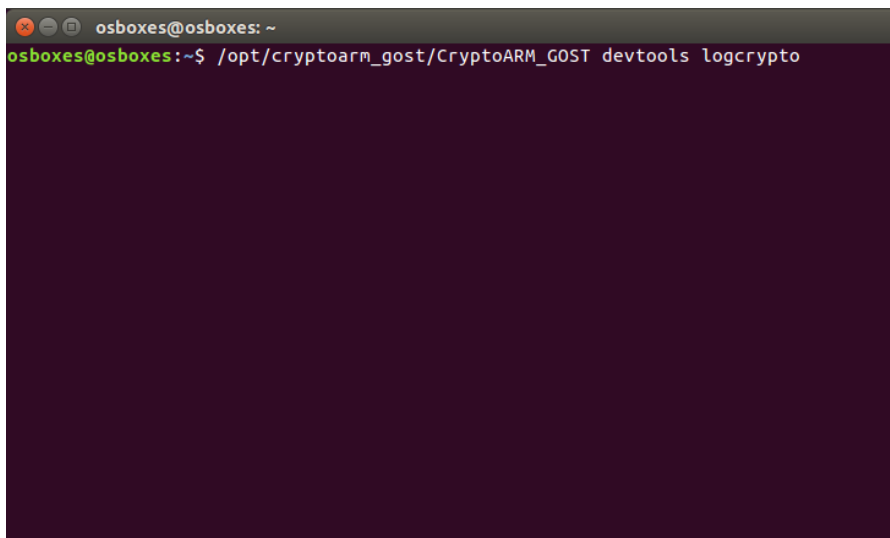


E

п    ϕ ч    г д к    п    к г R

E

st    st z    rst    st ?    R    d st    r    st



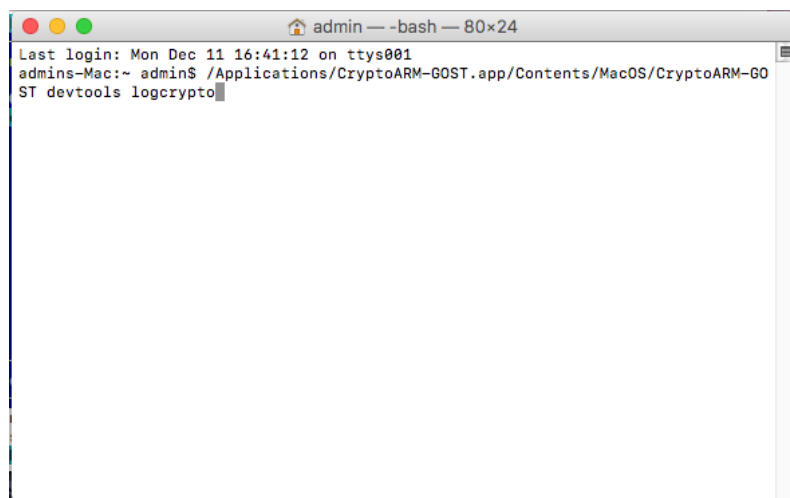
E



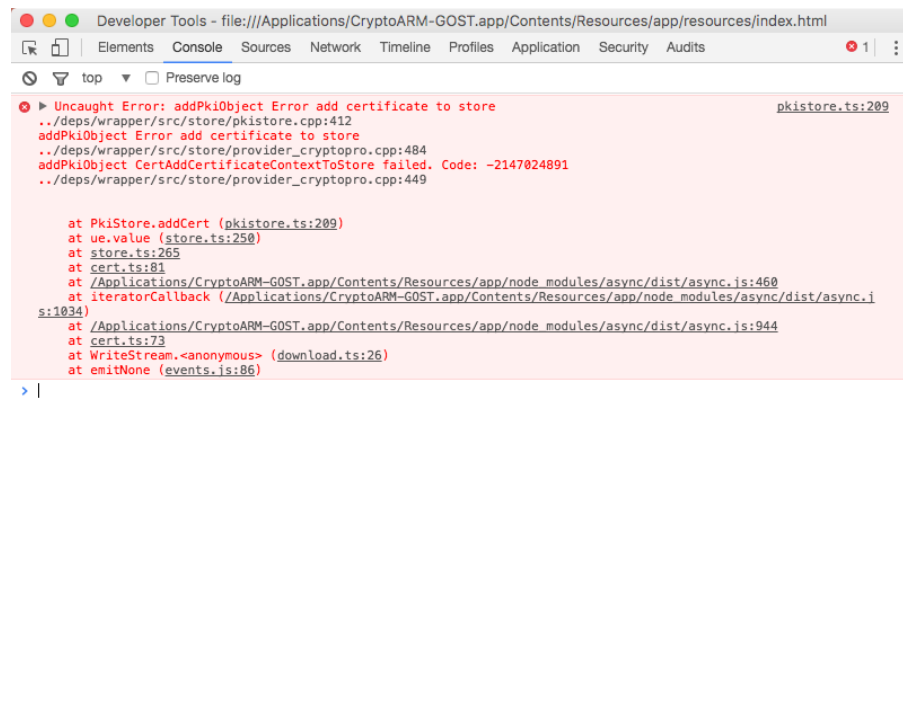
E

9 п ф ч г д к п к г j т

zst r st R z st z R st R d st r st



E



E

Н ? Z Н К П

М

М Т Е

D ч г

Запрос Sign

ЫФ D Ф Г		Г
		Т

ЫФ D Ф Г		Г

<http://www.jsonrpc.org/specification>

--	--	--

ЫФ	Д ФГ	Г

ЫФ	Д ФГ	Г
		Т

ЫФ	Д ФГ	Г

ЫФ	Д ФГ	Г

ЫФ	Д ФГ	Г

ЫФ	Д ФГ	Г

Запрос Verify

ЫФ				D ФГ		Г	

ЫФ	Д ФГ Г	Г

ЫФ	Д ФГ Г	Г

ЫФ	Д ФГ Г	Г

ЫФ	Д ФГ Г	Г

Д ч

Запрос Encrypt



ЫФ	Д Ф Г	Г
		Т

ЫФ	Д Ф Г	Г

ЫФ	Д Ф Г	Г

ЫФ	Д Ф Г	Г
		Т

ЫФ	Д Ф Г	Г

ЫФ	Д Ф Г	Г

ЫФ	Д ФГ Г	Г

ЫФ	Д ФГ Г	Г

Запрос Decrypt

ЫФ	Д ФГ Г	Г
		Т

ЫФ	Д ФГ Г	Г

\_\_\_\_\_

ЫФ		Д ФГ Г	Г

ЫФ		Д ФГ Г	Г
			Т

ЫФ		Д ФГ Г	Г

ЫФ		Д ФГ Г	Г

ЫФ		Д ФГ Г	Г

ЫФ		Д ФГ Г	Г

9 н ?  
н к п

z к г ь rst st н г

— E

E

—

— TF

— E

—

9 гп R

E

E

E

—

—

E

—

—

—

---

[T E](#)



R D ? ?

D	Г	К П Ч	Г
R D ? ? D		E	

D	Г	П	Г
D J DR		E	

К П	Г
R D ? D _ D ? DR_ R D R_ J ? _R ! ? ? ? _ T D ! ? ? ? R ? DR ! ? ? ? D _ ? ? ? D ? DR D _ D ? D D D _ D ? D JJD R _	

isDetached(): boolean

F

certificates(index: number): Certificate

D	Г	П	Г
D			



*F*

certificates(): CertificateCollection

*E*

*E*

*F*

signers(index: number): Signer

κ ΓΠ		Π	Γ
D			

*F*

signers(): SignerCollection

*F*



load(filename: string, format: DataFormat): void

К ГП	П	Г
JD ? D ?		FT

F

static load(filename: string, format: DataFormat): SignedData

К ГП	П	Г
JD ? D ?		FT

import(buffer: Buffer, format: DataFormat): void

К ГП	П	Г
T D ?		FT

static import(buffer: Buffer, format: DataFormat): SignedData

К ГП	П	Г
T D ?		FT

export(format: DataFormat): Buffer

К ГП	П	Г
?		FT

F



*F*

save(filename: string, format: DataFormat): void

Κ ΓΠ		Π	Γ
JD ? D			FT
	?		

*F*

*F*

createSigner(cert: Certificate, key: Key): Signer

Κ ΓΠ		Π	Γ
D		E	
D			

*F*

*F*

verify(certs: CertificateCollection): boolean

Κ ΓΠ		Π	Γ
D R		E E	

*F*





sign(): void

*F*

*F*

*F*



R D

D Γ		Π	Γ
D ? D DR ? J R D		E	

K ΓΠ			Γ
R D ? T DR_R D ? T D JJD T R D ? T DR_R D ? T D JJD D D _ D _			

signedAttributes(): SignerAttributeCollection

F

signedAttributes(index: number): Attribute

K ΓΠ	Π	Γ
D		

unsignedAttributes(): SignerAttributeCollection

unsignedAttributes(index: number): Attribute

K ΓΠ	Π	Γ
------	---	---



D	
---	--

*F*

verifyContent(v: ISignedDataContent): boolean

κ ΓΠ	Π	Γ

verify(): boolean



R D JJD

D	Г	П	Г
JD			

К	ГП	Г
D R		

items(index: number): Signer

К	ГП	П	Г
D			

*F*



R D ? T D JJD

D	Г	П	Г
JD			

К	ГП	Г
D R		
TR		
D D?		

push(attr: Attribute): void

К	ГП	П	Г
?			

removeAt(index: number): void

К	ГП	П	Г
D			

items(index: number): Attribute

К	ГП	П	Г
D			

F





R D

D Γ		Π	Γ
RRTD	?	D_	
RD	?	J T D _	
D	_		

F



R D D

В классе реализованы следующие свойства и методы:

D	Г	П	Г
RRTD ? D		R	
RD ? J T D		R	

К	ГП	Г
D		E T

ktriCertCmp(cert: pki.Certificate): number

E T

F

F

R D D JJD

В классе реализованы следующие свойства и методы:

D	Г	П	Г
JD		T D	

К	ГП	Г
TR		
D D?		





push(ri: CmsRecipientInfo): void

pop(): void

removeAt(index: number): void

*F*

*var r1 = ris.items(0);*

*ris.push(r1);*

*ris.push(r1);*

*ris.pop();*



# 9 ΓΠ

?

T

Π Π		Γ	
R T	—		
R T	— ? JD? ? D ? J		
R T	— ? D?R		

D Γ	Π Γ
? D	
D	

K Π Γ	Γ
T J ? D	
R DR	

constructor()

constructor(handle: native.PKI.Algorithm)

constructor(name: string)

duplicate(): Algorithm

isDigest(): boolean





2 T D

T

Π Π		Γ	
R T	JD?		
D	T D		

D Γ	Π	Γ
R D		
D		

κ ΠΠ	Γ
T J 2 D	
D	FT
2JTDR	FT

duplicate(): Attribute

export(): Buffer

FT

values(): AttributeValueCollection

FT

values(index: number): Buffer

FT

κ ΠΠ	Π	Γ
D		



Т D JTD JJD

Т

П П		Г
R T	_ J D J D T D JTD JJD	

D Г	П	Г
JD		

К ПП	Г
TR	
D D	
D R	

push(val: Buffer): void

К ПП	П	Г
J		

pop(): void

removeAt(index: number): void

К ПП	П	Г
D		

items(index: number): Buffer

К ПП	П	Г
D		



Д ? Д

П П		Г
Р Т	—	
Р Т	— ? JD? ? Д Д ? Д	

Д Г	П Г
Д Р RD ? J Т Д Д Д TR? Д RRTD Д J ? Д RRTD ? Д RT Д Д J ? Д RT Д ? Д Д Д ? Д Т R ? Т D? J R ? Т Д DR ? ? ? Д R Т JR ? RRTD RT JR R ? RRDJ R Д	E

К ПП	Г
? Д Д Т? JR ? R Т J ? Д J ? Д R? Д J ?	



compare(cert: Certificate): number

κ ΓΠ	Π	Γ
D	E	

F

F

equals(cert: Certificate): boolean

κ ΓΠ	Π	Γ
D	E	

hash(algorithm?: string): String

κ ΓΠ	Π	Γ
J		

F



duplicate(): Certificate

К ГП	П	Г
JD J		

F

load(filename: string, format?: DataFormat): void

К ГП	П	Г
JD J D		FT

F

static load(filename: string, format?: DataFormat): Certificate

К ГП	П	Г
JD J D		FT

import(buffer: Buffer, format?: DataFormat): void

К ГП	П	Г
T D		





static import(buffer: Buffer, format?: DataFormat): Certificate

Κ ΓΠ		Π	Γ
T	D		

export(format?: DataFormat): Buffer

Κ ΓΠ		Π	Γ

save(filename: string, format?: DataFormat): void

Κ ΓΠ		Π	Γ
JD	D		

*F*

*F*

download(urls: string[],pathForSave: string, done: (err: Error, certificate: Certificate)): void

Κ ΓΠ		Π	Γ
T	JR		

*F*

*CAIssuersUrls*





D ? D TDR

E T

E T F F EF T E T T

Π Π Γ									
R T	—								
R T	— ?	JD?	?	D	D	?	D TDR	E	T

D Γ	Π Γ
D R	

Κ ΓΠ	Γ
J ?	
R	
D	

load(filename: string, format?: DataFormat): void



Κ ΓΠ	Π Γ
JD ? D	FT
?	

*F*

static load(filename: string, format?: DataFormat): CertificationRequest

Κ ΓΠ	Π Γ
------	-----



JD  D 	FT
--	----

sign(key: Key): void

K ΓΠ	Π	Γ
D		

verify(): boolean



D    ?    D TDR

E            T

E            T            F    F EF            F FT

ET

Π   Π		Γ	
R   T	—		
R   T	— ? JD? ? D    D    ?    D TDR		
		E	T

D   Γ	Π	Γ	
RT   D		E	E
T   D			
D R			

subject(x509name: string)

K ΓΠ	Π	Γ
125 ? D		

pubkey(pubkey: Key)

K ΓΠ	Π	Γ
T   D		

version(version: number)

K ΓΠ	Π	Γ
D R		



D    ?    JJD

Π   Π					Γ	
R	T	-				
R	T	-	?	JJD?		
?	D	D	?	D	JJD	E   E

D	Γ	Π	Γ
JJD			

Κ ΓΠ	Γ
D R	
TR	
D   D?	

items(index: number): Certificate

Κ ΓΠ	Π	Γ
D		

*F*

push(cert: Certificate): void

Κ ΓΠ	Π	Γ
D	E	



pop(): void

*F*

removeAt(index: number): void

κ ΓΠ		Π	Γ
D			

*F*

*F*



?

Π Π		Γ
R T	–	

Κ Π		Γ
T J	?	
D	?	

buildChain(cert: Certificate, certs: CertificateCollection): CertificateCollection

E E

Κ Π
-----





JR	E E
----	-----

*F*

*F*

*F*

```
var crls = new trusted.pki.CrlCollection();  
crls.push(trusted.pki.Crl.load("test.crl"));  
chain.verifyChain(outChain, crls);
```



D

Π		Π		Γ	
R	T	–	D	?	D?R

D		Г		Π		Г	
D				E			
D	D	R	D	R	E	E	
D							
D	D	D			E		
?RR							
DR							
D							
D							
R?J							
R?J							
?J							
D							
R							

K ΓΠ		Γ	
D			
D			
D	D	D	R

encrypt(filenameSource: string, filenameEnc: string, format?: DataFormat): void

K ΓΠ		Π		Γ	
JD	?	DR	T	D	
JD	?	DD			
		?			



```
var trusted = require("trusted-crypto");
var cipher = new trusted.pki.Cipher("aes256");
var cert = new trusted.pki.CertificateCollection();
cert.push(trusted.pki.Certificate.load("cert.crt", trusted.DataFormat.PEM));
cipher.recipientsCerts = cert;
cipher.encrypt("test.txt", "encAssym.enc", trusted.DataFormat.PEM);
```

**decrypt(filenameEnc: string, filenameDec: string, format?: DataFormat): void**

К П	П	Г
JD ? DD		
JD ? D D		
?		
		FT



J

ET

Π Π		Γ	
R T	—		
R T	— ? JD? ? D J		ET

D Γ	Π	Γ
D D		ET FT
R ? T D		
D R		
RRTD ? D		
RRTD D J ? D		
J?R T ? D		
D T ? D		
T		
R ?J ? D		ET
R ?J R ? D		ET
R ?J		ET

Κ ΓΠ	Γ
J ?	
D	
R? D	
? D	
D T?JR	
?R	
T J ? D	
D D	

load(filename: string, format: DataFormat): void

Κ ΓΠ	Π	Γ
JD ? D		
?		





*F*

save(filename: string, dataFormat: DataFormat): void

Κ ΓΠ		Π	Γ
JD ? D ? ? ?			

*F*

compare(crl: Crl): number

Κ ΓΠ	Π	Γ
J	E	E

*F*

*F*

equals(crl: Crl): boolean

Κ ΓΠ	Π	Γ
J	E	E

*F*



*F*

hash(algorithm?: string): String

Κ ΓΠ		Π	Γ
Ⓜ J			

*F*

duplicate(): Crl

*F*

revoked(): RevokedCollection

*F*



J JJD

Π Π		Γ
R T	–	
R T	– ? JJD? ? D J JJD	E E

D Γ	Π Γ
JD	

Κ ΓΠ	Γ
D R	
TR	
D D?	

items(index: number): Crl

Κ ΓΠ	Π Γ
D	

F

push(crl: Crl): void

Κ ΓΠ	Π Γ
J	E





*F*

pop(): void

removeAt(index: number): void

κ ΓΠ		Π	Γ
D			

*F*



R

Π Π				Γ			
R	T	⊖	D?R	!	D ? D !	DR	?R

D Γ		Π		Γ	
D	D				

κ ΓΠ		Γ	
R?	D		

save(filename: string, dataFormat: DataFormat): void

κ ΓΠ		Π		Γ	
JD	?	D			
?	?		?		

F

F



D

F    T

F    F

Π   Π		Γ			
R	T	—			
R	T	— ?	JD?	? D	D

K ΠΠ	Γ			
D D ? D				
D?       ? D D				
D    ? D D				
D?   T J   D				
D T J   D				
? D				

generate(format: DataFormat, pubExp: PublicExponent, keySize: number, password: string): Key

K ΓΠ	Π	Γ	
?		FT	
T D	F		
D R D			
?RR			

F

F

readPrivateKey(filename: string, format: DataFormat, password: string): Key



κ ΓΠ	Π	Γ
JD ? D ? ?RR		FT

*F*

writePrivateKey(filename: string, format: DataFormat, password: string): any

κ ΓΠ	Π	Γ
JD ? D ? ?RR		FT

*F*

*F*

*F*

readPublicKey(filename: string, format: DataFormat): Key

κ ΓΠ	Π	Γ
JD ? D ?		FT

*F*

writePublicKey(filename: string, format: DataFormat): any

κ ΓΠ	Π	Γ
JD ? D		



?	FT
---	----

*F*

*F*

*F*

compare(key: Key): number

κ γπ	π	γ
D		

*F*

*F*



Π Π				Γ			
R	T	–	?R				
R	T	–	?	JD?	?	D	

D		Г		П		Г	
JTD							
J		J	D				
R		J	D				



R

E

Π Π		Γ	
R T	—		
R T	— z z ? ? D R		

Κ ΓΠ		Γ	
D ? D			
D			
?			
J ?			
R? D			
D? D			

certificate(password: string): Certificate

Κ ΓΠ		Π Γ	
?	RR		

key(password: string): Key

Κ ΓΠ		Π Γ	
?	RR		



ca(password: string): CertificateCollection

Κ ΓΠ	Π	Γ
JD ⓘ D		

load(filename: string): void

Κ ΓΠ	Π	Γ
JD ⓘ D		

static load(filename: string): Pkcs12

Κ ΓΠ	Π	Γ
JD ⓘ D		

save(filename: string): void

Κ ΓΠ	Π	Γ
JD ⓘ D		





*F*

*F*

create(cert: Certificate, key: Key, ca: CertificateCollection, password: string, name: string): Pkcs12

E

κ ΓΠ	Π	Γ
D	E	
D		
?	E	E
RR		
D		

*F*

*F*



D 2

ET

T

Π Π			Γ
R	T	–	

Κ ΓΠ			Γ
D	JJ	2J	
D	J	R	R
D	J	D	
J	2	J	

getCrLocal(cert: Certificate, store: PkiStore): any

Κ ΓΠ		Π	Γ
D		E	
R	D		

F

F

getCrDistPoints(cert: Certificate): Array<string>



κ ΓΠ	Π	Γ
D	E	

*F*

checkCrITime(crl: Crl): boolean

κ ΓΠ	Π	Γ
J	E	

*F*

downloadCRL(distPoints: Array<string>, pathForSave: string, done: Function): void

κ ΓΠ	Π	Γ
R R R? D D		

*F*

*F*



D D

Π Π		Γ	
R T	–		
R T	– z z ? zst d d d	T	

D Γ		Π Γ	
RD J T	D		
D J	J D		
DJR			

K ΓΠ		Γ	
T J J D			

duplicate(): Revoked

F



D D R

Π Π Γ									
R	T	_							
R	T	_	z	z	?	zst	d	d	d st T

D Γ Π Γ
JD

K ΓΠ Γ	
D R	
TR	
D D?	

items(index: number): Revoked

Κ ΓΠ	Π	Γ
D		

F

push(rv: Revoked): void

Κ ΓΠ	Π	Γ
	Τ	



*F*

pop(): void

*F*

removeAt(index: number): void

κ ΓΠ		Π	Γ
D			

*F*



99 ΓΠ R D

ⓂR R

M

Π Π		Γ	
R T	_ JD Ⓜ D?R	M	

K ΓΠ		Γ	
D			

export(): native.PKISTORE.IPkiltem[]

import(items: native.PKISTORE.IPkiltem[]): void

K ΓΠ		Π		Γ	
D R		TF			



D

E

Π Π			Γ
R	T	–	

Κ ΓΠ			Γ
D	D		
Ⓡ	R	Ⓢ D D	

getKey(cert: pki.Certificate)

Κ ΓΠ		Π	Γ
D		E	

F

hasPrivateKey(cert: pki.Certificate)

Κ ΓΠ		Π	Γ
D		E	

F





J D

П		П	Г	
R	T	—		

D	Г	П	Г
DR			
D R			
Ⓜ D R			FT T
ⓂR			
RT D Ⓜ D			
RT D D J Ⓜ D			
RRTD Ⓜ D			
RRTD D J Ⓜ D			
RD ⓂJ			



D

П П			Г
Р	Т	—	

Д Г	П Г
<p>?</p> <p>D</p> <p>D</p> <p>? D</p> <p>T</p> <p>?R</p> <p>RT D ? D</p> <p>RT D D J ? D</p> <p>RRTD ? D</p> <p>RRTD D J ? D</p> <p>RD ?J</p> <p>D D</p> <p>? D</p> <p>J?R T ? D</p> <p>D T ? D</p> <p>D</p> <p>D D</p> <p>? ? ? D</p> <p>R ? T D?J</p>	<p>FT T</p> <p>T</p>



R D

M

Π Π Γ									
R T	-	J D ?R							
R T	-	?	?	?	?	D R	D R	D	

D Γ		Π Γ	
?	R	E M	M

Κ ΓΠ		Γ	
?	D		
?	D		
?	J		
?	D		
?	R		
	D		
D	D		
D	R		

addProvider(provider: native.PKISTORE.Provider): void

Κ ΓΠ		Π Γ	
D		TF	

addCert(provider: native.PKISTORE.Provider, category: string, cert: Certificate): string

Κ ΓΠ		Π Γ	
D		TF	
?	D		FT T
D		E	



*F*

addCrl(provider: native.PKISTORE.Provider, category: string, crl: Crl): string

К ГП		П	Г
D	TF	FT	T
D	J	E	

addKey(provider: native.PKISTORE.Provider, key: Key, password: string): string

К ГП	П	Г
D	TF	
D	RR	

*F*

addCsr(provider: native.PKISTORE.Provider, category: string, csr: CertificationRequest): string

К ГП	П	Г
------	---	---



D		TF		FT		T
 D	R	E	T			

*F*

find(ifilter?: native.PKISTORE.IFilter): native.PKISTORE.IPkiltem[]

κ ΓΠ		Π		Γ	
J D		TF			

*F*

findKey(ifilter: native.PKISTORE.IFilter): native.PKISTORE.IPkiltem

κ ΓΠ		Π		Γ	
J D		TF			

*F*



getItem(item: native.PKISTORE.IPkItem): any

κ ΓΠ		Π	Γ
D		TF	

*F*

certs():pki.CertificateCollection

*F*



D R

Π Π			Γ
R	T	—	

Κ ΓΠ			Γ
D	D		
⊗R	⊗	D D	

getKey(cert: pki.Certificate)

Κ ΓΠ		Π	Γ
D		E	

F

hasPrivateKey(cert: pki.Certificate): boolean

Κ ΓΠ		Π	Γ
D		E	

F



D R R D

Π Π		Γ	
R T	– J D ?R		

κ ΓΠ		Γ	
D	D		

objectToPkiltem(path: string): native.PKISTORE.IPkiltem

κ ΓΠ	Π	Γ
?	T	





90      ΓΠ

D RRJ

Π   Π			Γ
R	T	–	

Κ ΓΠ		Γ
T		
R		
D	R	

run(): void

*trusted.common.OpenSSL.run();*

stop(): void

stop(): void



91 ГПТ JR

Функция download

К ГП	П	Г
Т J	Т	
?		



J D

Π Π		Γ
R T	–	

Κ ΓΠ		Γ
R ?		
R		
JD?		

```
static start(filename: string, level: LogLevel = DEFAULT_LOGGER_LEVEL): Logger
```

Κ ΓΠ		Π	Γ
JD ? D			
J D JD DJ		F	FT F F

```
logger = trusted.utils.Logger.start("logger.txt", trusted.LogLevel.ALL);
```

```
start(filename: string, level: LogLevel = DEFAULT_LOGGER_LEVEL): void
```

Κ ΓΠ		Π	Γ
JD ? D			
J D JD DJ		F	FT F F

```
var logger = new trusted.utils.Logger();
```

```
logger.start("logger.txt", trusted.LogLevel.ALL);
```

```
stop(): void
```



*logger.stop();*

clear(): viod



92

г г г

*F*

*F*

*F*

*F*

к г d

н г ь н г



г г Г z ф

\_\_\_\_\_



ф z Г z ф

\_\_\_\_\_

г г к ф

\_\_\_\_\_

z ь г к

\_\_\_\_\_



Ь К Ъ

—  
—

Е