Encrypting Message

Generated by Doxygen 1.9.2

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 EncryptedMessaging Namespace Reference

Classes

- class **Bytes**

  This class is used for combining and converting Byte array based on the input.
- class Contact

  This class contains all the functionalities related to contact features.
- class ContactConverter

  This class is used for converting contacts to public keys.
- class ContactMessage
- class Contacts

  This class contains all functions used for multiple contacts/group functionalities associated with them.
- class Context

  Our mission is to exacerbate the concept of security in messaging and create something conceptually new and innovative from a technical point of view. Top-level encrypted communication (there is no backend , there is no server-side contact list, there is no server but a simple router, the theory is that if the server does not exist then the server cannot be hacked, the communication is anonymous, the IDs are derived from a hash of the public keys, therefore in no case it is possible to trace who originates the messages, the encryption key is changed for each single message, and a system of digital signatures guarantees the origin of the messages and prevents attacks "men in de middle"). We use different concepts introduced with Bitcoin technology and the library itself: there are no accounts, the account is simply a pair of public and private keys, groups are also supported, the group ID is derived from a hash computed through the public keys of the members, since the hash process is irreversible, the level of anonymity is maximum). The publication of the source wants to demonstrate the genuineness of the concepts we have adopted! Thanks for your attention!
- class CryptoServiceProvider

  This class handles the encryption and decryption of the passphrases and validations.
- class **Functions**
- interface ICloudManager
- class Message

  Here are all the functions necessary to process messages in binary format, create messages and read them.
- class MessageFormat

  Message settings set for the recpients.
- class Messaging

  This class contains all the functions related to the messaging functionality for the users.
- class My

This class allows you to access all the information about the user who is using the application: the contact, if his cryptographic keys, the tokens of his device for notifications, the user id, etc.

- class **ProcessResponsesFromCloud**

  Process handlling and generating outpusts based on the conditional functions implemented.

- class Repository

  This library provides functions to retrieve messages on the server and store them locally.

- class Setting

  Configuration functions of setting message saving and deleting based on user input.

- class **Time**

  This class deals with time configuration related to the timestamps and sending and delivery of messages.

4.2 EncryptedMessaging.Cloud Namespace Reference

Classes

- class **Counter**
- class **ReceiveCloudCommands**
- class **SendCloudCommands**

  This class contains commands that can be sent to the cloud server

Enumerations

- enum **Keys** : byte {
  CommandType , **Subject** , **Id** , **Data** ,
  Name , **PubKey** , **Token** , **IsVideo** ,
  Device , **All** , **Hash** }
- enum **CommandType** : byte { **Post** , **Get** }
- enum **Subject** : byte {
  BackupContact , **PushNotification** , **DataStorage** , **Avatar** ,
  ClientRequest }

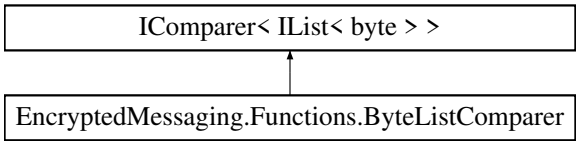
Chapter 5

Class Documentation

5.1 EncryptedMessaging.Functions.ByteListComparer Class Reference

Compare the two byte list and return the result.

Inheritance diagram for EncryptedMessaging.Functions.ByteListComparer:



Public Member Functions

- int Compare (IList< byte > x, IList< byte > y)

5.1.1 Detailed Description

Compare the two byte list and return the result.

5.1.2 Member Function Documentation

5.1.2.1 Compare()

```
int EncryptedMessaging.Functions.ByteListComparer.Compare (
            IList< byte > x,
            IList< byte > y )
```

Parameters

x	
y	

Returns

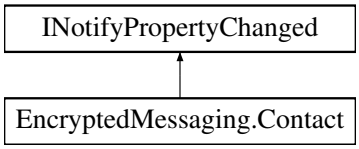
The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Functions.cs

5.2 EncryptedMessaging.Contact Class Reference

This class contains all the functionalities related to contact features.

Inheritance diagram for EncryptedMessaging.Contact:



Classes

- class MessageInfo

  Set message information on creation and sent.
- class RemoteReaded

  Add a timestamp to the message.

Public Types

- enum RuntimePlatform {
  Undefined , Android , iOS , Windows ,
  UWP , Unix , Mac }

  Base class for the runtime platform.

Public Member Functions

- Contact (Context context, List< byte[]> participants, string name=null, string language="en", RuntimePlatform os=RuntimePlatform.Undefined, string firebaseToken=null, string deviceToken=null, bool isServer=false)

  Checks if the chat initiated is between a single user or multiple partipants/group.
- string GetRealName ()

  Get the contact's real name (if set), otherwise return null
- string Pseudonym ()

  Set assumed Pseudonyms for the participants in the group.
- void ReadPosts (bool resetPaginate=false)

  Read the messages archived for this contact and view them in the chat view. The use of command in sequence will carry out the pagination (load the messages in blocks), to reset the pagination use the appropriate parameter.
- delegate Action PostBackup (ulong chatId, byte[] post, DateTime receptionDate)

  Delegated function that is passed by the client application to back up a post
- DateTime GetPosts (Action< byte[], DateTime > action, List< DateTime > exclude=null, DateTime receprionAntecedent=default, int? take=null)

  Read chat posts (encrypted in row format) for this contact or group (you can use this function to backup chat data)
- void SetPost (byte[] post, DateTime receptionDate)

  Manually set the post (it is recommended to use this function only for restoring data saved with the help of GetPosts)
- DateTime GetMessages (Action< Message, bool > actionToExecuteForEachMessage, List< DateTime > exclude=null, DateTime receprionAntecedent=default, int? take=null)

  Returns to a specific action, the messages visible in the chat. Useful function for exporting messages.
- void Save (bool cloudBackup=false)

  Save the data on cloud storage.
- object Clone ()

  Create a shallow copy of the current object
- string GetQrCode ()

  Get thr QR code string for the context.

Static Public Member Functions

- static void ExportPosts (Context context, PostBackup exportAction, List< DateTime > exclude, DateTime receprionAntecedent=default, int? take=null)

  Export all posts (encrypted in row format), useful for example for making a backup

Public Attributes

- Dictionary< string, object > **Session** = new Dictionary<string, object>()

  It can be used by server applications to store data relating to this contact during a session. When the session expires, the Contact object is deleted and the session data will be deleted (the concept is similar to web application session data)
- System.Drawing.Color **LightColor**

  Represents an ARGB (alpha, red, green, blue) color.
- System.Drawing.Color **DarkColor**

  Represents an ARGB (alpha, red, green, blue) color.
- string **Language**

  Language used for this contact.
- RuntimePlatform **Os**

  Operating system in use for runtime application.
- DateTime **LastNotMyMessageTime**

Date of the last message time of the user.

- DateTime **LastNotMyMessageTimeAtSendLastReading**
- bool SendConfirmationOfReading = true
- bool **TranslationOfMessages**

  Indicates whether messages are required to be translated. If you change this parameter, you need to save the contact
- bool IsServer = false
- MessageInfo **LastMessageDelivered**

  Set message information on last message delivery.
- MessageInfo **LastMessageSent**

  Set message information on last message sent.

Properties

- string? **Name** `[getset]`

  Get the contact's name (if set), otherwise return null.
- string?? **MyRemoteName** `[getset]`

  The name of my contact registered on the recipient's device
- string **LightColorAsHex** `[get]`

  Convert color to hex.
- string **DarkColorAsHex** `[get]`

  Convert color to hex.
- string **PublicKeys** `[getset]`

  Check if public key exists
- DateTime **LastMessageTime** `[getset]`

  Date of the most recent message in the chat. Be careful, this value is UTC
- string LastMessageTimeDistance `[getset]`
- string **LastMessageTimeText** `[get]`

  Date of the most recent message in the chat in string format. This value is Local Time
- List< byte[ ]> **Participants** `[get]`

  The public keys of all members of the group
- bool? IsBlocked `[getset]`
- bool ImBlocked `[getset]`
- bool IsMuted `[getset]`
- bool IsVisible `[get]`
- RemoteReaded[ ] **RemoteReadedList** `[getset]`

  Add new timestamp to to the list.
- bool **IsGroup** `[get]`

  Set as group if more than two participants.
- string **LastMessagePreview** `[getset]`

  Get the preview of last message.
- bool **LastMessageIsMy** `[getset]`

  Boolean check for last message.
- uint **RemoteUnreaded** `[getset]`

  This is an empirical value, please do not use it unless necessary. Unfortunately in iOS it is not possible to locally update the red dot with the number of unread messages, this value must be sent with the notification.
- int **LastMessageFontAttributes** `[getset]`

  Integer value for last message font attributes.
- ulong **ChatId** `[get]`

  Integer value of the Chat Id.
- ulong? **UserId** `[get]`

  Is the user id of your single contact, if is a group this value is null

- string **FirebaseToken** [getset]

  Get the fire base token value and save it.
- string **DeviceToken** [getset]

  Return and save the device token.
- byte[] **Avatar** [getset]

  Set avator for the input byte array.
- DateTime **LastSeen** [getset]

  The last time the user watched this chat. All messages after this date are to be considered as unseen.
- int **UnreadMessages** [getset]

  Integer value get for unread messages.
- object **MessageContainerUI** [getset]

  It can be used by the client program to store the user interface of this chat
- object **NameFirstLetter** [get]

  Get the First letter and set it to uppercase.

Events

- PropertyChangedEventHandler **PropertyChanged**

  Represents the method that will handle the proporty changed event raised when a property when a property is changed on a component.
- Contacts.LastMessagChanged **LastMessageChanged**

  Check if the last message was changed.

5.2.1 Detailed Description

This class contains all the functionalities related to contact features.

5.2.2 Member Enumeration Documentation

5.2.2.1 RuntimePlatform

`enum EncryptedMessaging.Contact.RuntimePlatform`

Base class for the runtime platform.

Enumerator

Undefined	Undefined
Android	Andriod
iOS	IOS
Windows	Windows
UWP	Universal windows platform
Unix	Unix
Mac	Macintosh

5.2.3 Constructor & Destructor Documentation

5.2.3.1 Contact()

```
EncryptedMessaging.Contact.Contact (
            Context context,
            List< byte[]> participants,
            string name = null,
            string language = "en",
            RuntimePlatform os = RuntimePlatform.Undefined,
            string firebaseToken = null,
            string deviceToken = null,
            bool isServer = false )
```

Checks if the chat initiated is between a single user or multiple partipants/group.

Parameters

context	Context
participants	Users in the contacts
name	Name of the user
language	Application language
os	Operating system
firebaseToken	Token Firebase
deviceToken	Device Token of iOS
isServer	Is a server

5.2.4 Member Function Documentation

5.2.4.1 Clone()

```
object EncryptedMessaging.Contact.Clone ( )
```

Create a shallow copy of the current object

Returns

5.2.4.2 ExportPosts()

```
static void EncryptedMessaging.Contact.ExportPosts (
            Context context,
            PostBackup exportAction,
            List< DateTime > exclude,
            DateTime receprionAntecedent = default,
            int?  take = null )  [static]
```

Export all posts (encrypted in row format), useful for example for making a backup

Parameters

context	Context
exportAction	Action that will be performed for each post (If you want to back up put the backup execution code here)
exclude	List of posts to exclude using the received date as a filter. It is recommended to use this parameter to avoid exporting posts that have already been exported in the past
receprionAntecedent	If set, consider only posts that are dated before the value indicated. It is useful for paginating messages in the chat view, or for telling the loading of messages in blocks. How to use this parameter: You need to store the date of the oldest message that is displayed in the chat, when you want to load a second block of messages you have to pass this date in order to get the next block
take	Limit the number of messages to take (set this value to paginate messages in chunks), in case you don't want the whole message list. Pass null to process all posts, without any paging!

5.2.4.3 GetMessages()

```
DateTime EncryptedMessaging.Contact.GetMessages (
            Action< Message, bool > actionToExecuteForEachMessage,
            List< DateTime > exclude = null,
            DateTime receprionAntecedent = default,
            int?  take = null )
```

Returns to a specific action, the messages visible in the chat. Useful function for exporting messages.

Parameters

actionToExecuteForEachMessage	Action that is performed for each message (use this action to export or process messages).
exclude	List of id messages to exclude (The reception time is used as an identifier)
receprionAntecedent	Filter messages considering only those prior to a certain date (useful for paging messages in blocks)
take	Limit the number of messages to take. If not set, the value set in the Context.Setting.MessagePagination settings will be used. Pass the Context.Setting.KeepPost value to process all messages!

Returns

Returns the date of arrival of the oldest message processed by the function. Use this value to page further requests by passing the "receprionAntecedent" parameter.

5.2.4.4 GetPosts()

```
DateTime EncryptedMessaging.Contact.GetPosts (
            Action< byte[], DateTime > action,
            List< DateTime > exclude = null,
            DateTime receprionAntecedent = default,
            int?  take = null )
```

Read chat posts (encrypted in row format) for this contact or group (you can use this function to backup chat data)

Parameters

action	Action to be performed for each post, the byte[] is binary data of the encrypted post that is read from the repository, DateTime is the time the post was received which you can use as a unique ID (you can use the ticks property of DateTime as a unique id)
exclude	List of posts to exclude using the received date as a filter. It is recommended to use this parameter to avoid exporting posts that have already been exported in the past
receprionAntecedent	If set, consider only posts that are dated before the value indicated. It is useful for paginating messages in the chat view, or for telling the loading of messages in blocks. How to use this parameter: You need to store the date of the oldest message that is displayed in the chat, when you want to load a second block of messages you have to pass this date in order to get the next block
take	Limit the number of messages to take (set this value to paginate messages in chunks), in case you don't want the whole message list. Pass null to process all posts, without any paging!

Returns

Returns the date of arrival of the oldest message processed by the function. Use this value to page further requests by passing the "receprionAntecedent" parameter

5.2.4.5 GetQrCode()

```
string EncryptedMessaging.Contact.GetQrCode ( )
```

Get thr QR code string for the context.

Returns

5.2.4.6 GetRealName()

```
string EncryptedMessaging.Contact.GetRealName ( )
```

Get the contact's real name (if set), otherwise return null

Returns

Contact's real name if set

5.2.4.7 PostBackup()

```
delegate Action EncryptedMessaging.Contact.PostBackup (
            ulong chatId,
            byte[] post,
            DateTime receptionDate )
```

Delegated function that is passed by the client application to back up a post

Parameters

chatId	Chat id
post	Encrypted post
receptionDate	Date of receipt of the post

Returns

5.2.4.8 Pseudonym()

```
string EncryptedMessaging.Contact.Pseudonym ( )
```

Set assumed Pseudonyms for the participants in the group.

Returns

5.2.4.9 ReadPosts()

```
void EncryptedMessaging.Contact.ReadPosts (
            bool resetPaginate = false )
```

Read the messages archived for this contact and view them in the chat view. The use of command in sequence will carry out the pagination (load the messages in blocks), to reset the pagination use the appropriate parameter.

Parameters

resetPaginate	Reset pagination, i.e. messages will be loaded from starting with the most recent message block (the first page of messages).

5.2.4.10 Save()

```
void EncryptedMessaging.Contact.Save (
            bool cloudBackup = false )
```

Save the data on cloud storage.

Parameters

cloudBackup	Boolean

5.2.4.11 SetPost()

```
void EncryptedMessaging.Contact.SetPost (
            byte[] post,
            DateTime receptionDate )
```

Manually set the post (it is recommended to use this function only for restoring data saved with the help of GetPosts)

Parameters

post	The encrypted post binary data to be saved in the repository
receptionDate	DateTime is the time the post was received which you can use as a unique ID (you can use the ticks property of DateTime as a unique id)

5.2.5 Member Data Documentation

5.2.5.1 IsServer

```
bool EncryptedMessaging.Contact.IsServer = false
```

5.2.5.2 SendConfirmationOfReading

```
bool EncryptedMessaging.Contact.SendConfirmationOfReading = true
```

5.2.6 Property Documentation

5.2.6.1 ImBlocked

```
bool EncryptedMessaging.Contact.ImBlocked  [get], [set]
```

5.2.6.2 IsBlocked

```
bool?  EncryptedMessaging.Contact.IsBlocked  [get], [set]
```

5.2.6.3 IsMuted

```
bool EncryptedMessaging.Contact.IsMuted  [get], [set]
```

5.2.6.4 IsVisible

```
bool EncryptedMessaging.Contact.IsVisible  [get]
```

5.2.6.5 LastMessageTimeDistance

```
string EncryptedMessaging.Contact.LastMessageTimeDistance  [get], [set]
```

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Contact.cs

5.3 EncryptedMessaging.ContactConverter Class Reference

This class is used for converting contacts to public keys.

Public Member Functions

- ContactConverter (Context context)

  Set the context to readonly.
- bool PublicKeysToParticipants (string publicKeys, out List< byte[ ]> participants)

  This function obtains the list of participants from a string that represents everyone's public key
- bool ParticipantsToPublicKeys (List< byte[ ]> participants, out string publicKeys, bool removeMyKey=false)

  Boolean check for validating key.
- bool ValidateKeys (string keys)

  Boolean check for partipants public keys.
- void NormalizeParticipants (ref List< byte[ ]> participants, bool removeMyKey=false)

  Remove they key if it is not null and assign a new Public key Binary for empty values.

Static Public Member Functions

- static ulong GetUserId (byte[ ] publicKey)

  From the public key he obtains the user ID, a unique number represented by 8 bytes (ulong) For privacy reasons this algorithm is not reversible: From the public key we can obtain the user ID but it is not possible to trace the public key by having the user ID
- static ulong ParticipantsToChatId (List< byte[ ]> participants, string name)

  Calculate the hash id of the contact. For groups, the name also comes into play in the computation because there can be groups with the same participants but different names
- static bool ValidateKeys (List< byte[ ]> participants)
- static bool ValidateKey (string base64Key)

  This will check if the basekey is valid , if not key is converted from base 64 key.
- static bool ValidateKey (byte[ ] key)

  Validates the key provided.
- static List< ulong > ParticipantsToUserIds (List< byte[ ]> participants, Context context)

  Change the partipants to their specific User Ids.

5.3.1 Detailed Description

This class is used for converting contacts to public keys.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ContactConverter()

```
EncryptedMessaging.ContactConverter.ContactConverter (
            Context context )
```

Set the context to readonly.

Parameters

context	Context

5.3.3 Member Function Documentation

5.3.3.1 GetUserId()

```
static ulong EncryptedMessaging.ContactConverter.GetUserId (
            byte[] publicKey )  [static]
```

From the public key he obtains the user ID, a unique number represented by 8 bytes (ulong) For privacy reasons this algorithm is not reversible: From the public key we can obtain the user ID but it is not possible to trace the public key by having the user ID

Parameters

publicKey	

Returns

5.3.3.2 NormalizeParticipants()

```
void EncryptedMessaging.ContactConverter.NormalizeParticipants (
            ref List< byte[]> participants,
            bool removeMyKey = false )
```

Remove they key if it is not null and assign a new Public key Binary for empty values.

Parameters

participants	Participants
removeMyKey	Byte array

5.3.3.3 ParticipantsToChatId()

```
static ulong EncryptedMessaging.ContactConverter.ParticipantsToChatId (
            List< byte[]> participants,
            string name )  [static]
```

Calculate the hash id of the contact. For groups, the name also comes into play in the computation because there can be groups with the same participants but different names

Parameters

participants	Partipants
name	The name parameter must only be passed for groups, because there are groups with the same members but different names

Returns

Unisgned Integer

5.3.3.4 ParticipantsToPublicKeys()

```
bool EncryptedMessaging.ContactConverter.ParticipantsToPublicKeys (
            List< byte[]> participants,
            out string publicKeys,
            bool removeMyKey = false )
```

Boolean check for validating key.

Parameters

participants	Partipants
publicKeys	Public Key
removeMyKey	Remove Key

Returns

5.3.3.5 ParticipantsToUserIds()

```
static List< ulong > EncryptedMessaging.ContactConverter.ParticipantsToUserIds (
            List< byte[]> participants,
            Context context )  [static]
```

Change the partipants to their specific User Ids.

Parameters

participants	Partipants
context	Context

Returns

User Id

5.3.3.6 PublicKeysToParticipants()

```
bool EncryptedMessaging.ContactConverter.PublicKeysToParticipants (
            string publicKeys,
            out List< byte[]> participants )
```

This function obtains the list of participants from a string that represents everyone's public key

Parameters

publicKeys	string that represents everyone's public key
participants	participants in the group

Returns

Boolean

5.3.3.7 ValidateKey() [1/2]

```
static bool EncryptedMessaging.ContactConverter.ValidateKey (
            byte[] key )  [static]
```

Validates the key provided.

Parameters

key	

Returns

5.3.3.8 ValidateKey() [2/2]

```
static bool EncryptedMessaging.ContactConverter.ValidateKey (
            string base64Key )  [static]
```

This will check if the basekey is valid , if not key is converted from base 64 key.

Parameters

base64Key	

Returns

key

5.3.3.9 ValidateKeys() [1/2]

```
static bool EncryptedMessaging.ContactConverter.ValidateKeys (
            List< byte[]> participants )  [static]
```

Parameters

participants	

Returns

5.3.3.10 ValidateKeys() [2/2]

```
bool EncryptedMessaging.ContactConverter.ValidateKeys (
            string keys )
```

Boolean check for partipants public keys.

Parameters

keys	Key

Returns

Boolean

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/ContactConverter.cs

5.4 EncryptedMessaging.ContactMessage Class Reference

Classes

- class Properties

  Strings used in this class.

Public Member Functions

- ContactMessage (byte[ ] data)
- ContactMessage (string qrCode)

  Convert the QR code to base 64 string.
- void GetProperties (Context context, out string language, out Contact.RuntimePlatform os, out string firebaseToken, out string deviceToken)

  Return the OS and language for a non-group contact
- List< byte[ ]> GetParticipantsKeys (Context context)

  Get the keys of the particpant by converting the byte array input.

Static Public Member Functions

- static ContactMessage GetContactMessage (byte[ ] data)

  Returns the contact represented by the input. If the input is invalid it will return null
- static ContactMessage GetContactMessage (string qrCode)

  Returns the contact represented by the input. If the input is invalid it will return null
- static byte[ ] GetDataMessageContact (Contact contact, Context context, bool addFirebaseTokken=true, bool addDeviceTokken=true, bool fullDataParticipant=true, bool purposeIsUpdateOnly=false)

  Assign public keys to the partipants and set the firebase and device token for the users.
- static string GetQrCode (Contact contact, Context context)

  Get the contact info of the other user from the base 64 string.
- static string GetMyQrCode (Context context)

  Get the contact onfo of the person using the app from base 64 string.
- static void AddContact (string qrCode, Context context, Contacts.SendMyContact sendMyContact=Contacts.SendMyContact.Se

  Create new contact by using the QR code

Public Attributes

- bool **IsUpdate** = false

  Boolean for update, default is set false.
- List< Properties > **Participants** = new List<Properties>()

  List for participants.

Properties

- string **Name** `[getset]`

  Update the name of the group.

5.4.1 Detailed Description

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ContactMessage() [1/2]

```
EncryptedMessaging.ContactMessage.ContactMessage (
            byte[] data )
```

Parameters

data	Byte array

5.4.2.2 ContactMessage() [2/2]

```
EncryptedMessaging.ContactMessage.ContactMessage (
            string qrCode )
```

Convert the QR code to base 64 string.

Parameters

qrCode	QR code

5.4.3 Member Function Documentation

5.4.3.1 AddContact()

```
static void EncryptedMessaging.ContactMessage.AddContact (
            string qrCode,
            Context context,
            Contacts.SendMyContact sendMyContact = Contacts.SendMyContact.Send )  [static]
```

Create new contact by using the QR code

Parameters

qrCode	QR code
context	Context
sendMyContact	Option to send my contact to the contact I add

5.4.3.2 GetContactMessage() [1/2]

```
static ContactMessage EncryptedMessaging.ContactMessage.GetContactMessage (
            byte[] data )  [static]
```

Returns the contact represented by the input. If the input is invalid it will return null

Parameters

data	input byte array

Returns

5.4.3.3 GetContactMessage() [2/2]

```
static ContactMessage EncryptedMessaging.ContactMessage.GetContactMessage (
            string qrCode )  [static]
```

Returns the contact represented by the input. If the input is invalid it will return null

Parameters

qrCode	input base 64

Returns

5.4.3.4 GetDataMessageContact()

```
static byte[] EncryptedMessaging.ContactMessage.GetDataMessageContact (
            Contact contact,
            Context context,
            bool addFirebaseTokken = true,
            bool addDeviceTokken = true,
            bool fullDataParticipant = true,
            bool purposeIsUpdateOnly = false )  [static]
```

Assign public keys to the partipants and set the firebase and device token for the users.

Parameters

contact	User contact
context	Context
addFirebaseTokken	Token Firebase
addDeviceTokken	Device Token of iOS
fullDataParticipant	
purposeIsUpdateOnly	Update contact

Returns

5.4.3.5 GetMyQrCode()

```
static string EncryptedMessaging.ContactMessage.GetMyQrCode (
            Context context ) [static]
```

Get the contact onfo of the person using the app from base 64 string.

Parameters

context	context

Returns

5.4.3.6 GetParticipantsKeys()

```
List< byte[]> EncryptedMessaging.ContactMessage.GetParticipantsKeys (
            Context context )
```

Get the keys of the particpant by converting the byte array input.

Parameters

context	Context

Returns

Keys

5.4.3.7 GetProperties()

```
void EncryptedMessaging.ContactMessage.GetProperties (
            Context context,
            out string language,
            out Contact.RuntimePlatform os,
            out string firebaseToken,
            out string deviceToken )
```

Return the OS and language for a non-group contact

Parameters

context	Context
language	Language
os	Os
firebaseToken	Token Firebase
deviceToken	Device token for IOS

5.4.3.8 GetQrCode()

```
static string EncryptedMessaging.ContactMessage.GetQrCode (
            Contact contact,
            Context context )  [static]
```

Get the contact info of the other user from the base 64 string.

Parameters

contact	User contact
context	context

Returns

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/ContactMessage.cs

5.5 EncryptedMessaging.Contacts Class Reference

This class contains all functions used for multiple contacts/group functionalities associated with them.

Classes

- class Observable

  Add new participants to the group.

Public Types

- enum SendMyContact { None , Send , SendNamelessForUpdate }

  Enumerator that specifies whether when a contact is added to him I should send mine, and for what purpose

Public Member Functions

- Contacts (Context context)

  Set the time stamp when a message is sent to the participants.
- delegate void LastMessagChanged (Message message)

  Server side rule for last message changed.
- void **RestoreContactFromCloud** ()

  This Fucktion is used to restore a contact from the cloud storage.
- Observable< Contact > GetContacts ()

  Get a observable contact list.
- void ForEachContact (Action< Contact > action)

  For each visible contact it performs an action (Server-type contacts are excluded from performing the action, because they are generally not visible in the Address Book).
- delegate void OnContactAddedHandler (Contact contact)

  Delegate for the event that is triggered when a contact is added.
- Contact AddContact (ContactMessage contactMessage, SendMyContact sendMyContact=SendMyContact.Send)

  Turn a message into a contact. The ContactMessage is a type of data that is used to send contacts over the network
- Contact AddContact (string qrCode, SendMyContact sendMyContact=SendMyContact.Send)

  Add contact from the imput strng QR code.
- Contact AddContact (List< Contact > contacts, string name=null, bool isServer=false, SendMyContact sendMyContact=SendMyContact.Send)

  Add a contact to the address book, if there is already a contact with the same key then the existing contact will be renamed
- Contact AddContact (string publicKeys, string name=null, bool isServer=false, SendMyContact sendMyContact=SendMyContact.Send)

  Add a contact to the address book, if there is already a contact with the same key then the existing contact will be renamed
- Contact AddContactByKeys (string publicKeys, string name=null, bool isServer=false, SendMyContact sendMyContact=SendMyContact.Send)

  Add a contact to the address book, if there is already a contact with the same key then the existing contact will be renamed
- Contact AddContact (List< byte[ ]> participants, string name=null, bool isServer=false, string language=null, SendMyContact sendMyContact=SendMyContact.Send)

  Add contact to the address book, if the contact already exist, return duplicate.
- void RemoveContact (Contact contact)

  Remove the contact from the address book.
- void RemoveContact (string key)

  Remove the contact from the address book.
- void ClearContact (string key)

  Clear the chat history with the contact.
- List< byte[ ]> GetParticipants (ulong chatId)

  Get the List of contacts in the group.
- Contact GetContact (ulong chatId)

  Get the single contact stored in the list.
- Contact GetContactByUserID (ulong userId)

  Get contacts by user Id, if not present, returns a null value.
- Contact GetParticipant (byte[ ] key)

Look for a contact based on his public key, if not present in the book, a null value will be returned; Each contact is a group of at least 2 people: The user and the recipients; Contacts with more than one recipient are groups and will not be taken into consideration for research.

- string GetParticipantName (byte[ ] key)

  If the participant is present in the address book, he returns his name, otherwise he invents a name
- List< Contact > GetGroupParicipantContacts (Contact contact)

  Look for contacts in the group list and add new partipants if not present already.
- Contact ContactAlreadyExists (List< byte[ ]> participants, string name)

  Check if a contact already exists. The check is done using public keys
- Contact ContactAlreadyExists (string publicKeys, string name)

  Check if a contact already exists. The check is done using public keys
- string Pseudonym (string publicKeys)

  Convert public key to participants.
- string Pseudonym (List< byte[ ]> participants)

  Assign a Pseudonym for a participant in the list.
- string Pseudonym (ulong userId)

  Assign a Pseudonym for a participant for the unisgned long integer input in the list.
- void Colors (ulong userId, out System.Drawing.Color light, out System.Drawing.Color dark)

  Color functionality for the event on user Id.
- void Colors (List< byte[ ]> participants, out System.Drawing.Color light, out System.Drawing.Color dark)

  Color functionality for the event on partipants.
- Contact GetMyContact (bool nameless=false)

  Create Contact user data in the contact is null.
- Contact GetCloudContact ()

  If the contact is null on cloud, then it is stored on the cloud server.

Static Public Attributes

- static string **CloudPubKey** = @"ApkrRQUe7qbaKY05Lbs5z+o001UNzXlfHgm+9KEN41vE"

  Public key for cloud.
- static ulong **CloudUserId**

  set unsigned integer value for cloud user id.

Events

- Action< List< Contact > > **ContactsListChanged**

  Event action for contact list change.
- OnContactAddedHandler **OnContactAdded**

  Event that is triggered when new contacts are added

5.5.1 Detailed Description

This class contains all functions used for multiple contacts/group functionalities associated with them.

5.5.2 Member Enumeration Documentation

5.5.2.1 SendMyContact

`enum EncryptedMessaging.Contacts.SendMyContact`

Enumerator that specifies whether when a contact is added to him I should send mine, and for what purpose

Enumerator

None	Do not send my contact
Send	Send my contact to another contact, the contact contains my real name information
SendNamelessForUpdate	It is used to update the firebase tokens and the device id without changing the contact name. This is required when a user reinstalls the application.

5.5.3 Constructor & Destructor Documentation

5.5.3.1 Contacts()

```
EncryptedMessaging.Contacts.Contacts (
            Context context )
```

Set the time stamp when a message is sent to the participants.

Parameters

context	

5.5.4 Member Function Documentation

5.5.4.1 AddContact() [1/5]

```
Contact EncryptedMessaging.Contacts.AddContact (
            ContactMessage contactMessage,
            SendMyContact sendMyContact = SendMyContact.Send )
```

Turn a message into a contact. The ContactMessage is a type of data that is used to send contacts over the network

Parameters

contactMessage	A contact that has been received by the network
sendMyContact	Define whether the contact added here should receive our contact via the network, whether with the name or without. If we send our contact without a name, the recipient will only update the Firebase tokens and the device id, if instead the contact has the name, the recipient will add our contact including our real name.

Returns

5.5.4.2 AddContact() [2/5]

```
Contact EncryptedMessaging.Contacts.AddContact (
            List< byte[]> participants,
            string name = null,
            bool isServer = false,
            string language = null,
            SendMyContact sendMyContact = SendMyContact.Send )
```

Add contact to the address book, if the contact already exist, return duplicate.

Parameters

participants	Users in the list.
name	The name of the group/contact
isServer	Is a server (It will not be visible in the contacts directory)
language	Language
sendMyContact	Option to send my contact to the contact I add

Returns

5.5.4.3 AddContact() [3/5]

```
Contact EncryptedMessaging.Contacts.AddContact (
            List< Contact > contacts,
            string name = null,
            bool isServer = false,
            SendMyContact sendMyContact = SendMyContact.Send )
```

Add a contact to the address book, if there is already a contact with the same key then the existing contact will be renamed

Parameters

contacts	The list of source contacts to create a group
name	The name of the group/contact
isServer	Is a server (It will not be visible in the contacts directory)
sendMyContact	Option to send my contact to the contact I add

Returns

5.5.4.4 AddContact() [4/5]

```
Contact EncryptedMessaging.Contacts.AddContact (
            string publicKeys,
            string name = null,
            bool isServer = false,
            SendMyContact sendMyContact = SendMyContact.Send )
```

Add a contact to the address book, if there is already a contact with the same key then the existing contact will be renamed

Parameters

publicKeys	The group keys or the contact key
name	The name of the group/contact
isServer	Is a server (It will not be visible in the contacts directory)
sendMyContact	Option to send my contact to the contact I add

Returns

5.5.4.5 AddContact() [5/5]

```
Contact EncryptedMessaging.Contacts.AddContact (
            string qrCode,
            SendMyContact sendMyContact = SendMyContact.Send )
```

Add contact from the imput strng QR code.

Parameters

qrCode	QR code
sendMyContact	Option to send my contact to the contact I add

Returns

5.5.4.6 AddContactByKeys()

```
Contact EncryptedMessaging.Contacts.AddContactByKeys (
            string publicKeys,
            string name = null,
```

```
            bool isServer = false,
            SendMyContact sendMyContact = SendMyContact.Send )
```

Add a contact to the address book, if there is already a contact with the same key then the existing contact will be renamed

Parameters

publicKeys	The group keys or the contact key
name	The name of the group/contact
isServer	Is a server (It will not be visible in the contacts directory)
sendMyContact	Option to send my contact to the contact I add

Returns

5.5.4.7 ClearContact()

```
void EncryptedMessaging.Contacts.ClearContact (
            string key )
```

Clear the chat history with the contact.

Parameters

key	Public key

5.5.4.8 Colors() [1/2]

```
void EncryptedMessaging.Contacts.Colors (
            List< byte[]> participants,
            out System.Drawing.Color light,
            out System.Drawing.Color dark )
```

Color functionality for the event on partipants.

Parameters

participants	
light	
dark	

5.5.4.9 Colors() [2/2]

```
void EncryptedMessaging.Contacts.Colors (
            ulong userId,
```

```
            out System.Drawing.Color light,
            out System.Drawing.Color dark )
```

Color functionality for the event on user Id.

Parameters

user↩ Id	
light	
dark	

5.5.4.10 ContactAlreadyExists() [1/2]

```
Contact EncryptedMessaging.Contacts.ContactAlreadyExists (
            List< byte[]> participants,
            string name )
```

Check if a contact already exists. The check is done using public keys

Parameters

participants	Chat participants
name	For groups you must also pass the name because it is allowed to have groups with the same members but different names

Returns

Returns the duplicate contact if it already exists, otherwise null

5.5.4.11 ContactAlreadyExists() [2/2]

```
Contact EncryptedMessaging.Contacts.ContactAlreadyExists (
            string publicKeys,
            string name )
```

Check if a contact already exists. The check is done using public keys

Parameters

publicKeys	PublicKeys of contact
name	Name of the partipants

Returns

Returns the duplicate contact if it already exists, otherwise null

5.5.4.12 ForEachContact()

```
void EncryptedMessaging.Contacts.ForEachContact (
            Action< Contact > action )
```

For each visible contact it performs an action (Server-type contacts are excluded from performing the action, because they are generally not visible in the Address Book).

Parameters

action	Action to execute

5.5.4.13 GetCloudContact()

```
Contact EncryptedMessaging.Contacts.GetCloudContact ( )
```

If the contact is null on cloud, then it is stored on the cloud server.

Returns

5.5.4.14 GetContact()

```
Contact EncryptedMessaging.Contacts.GetContact (
            ulong chatId )
```

Get the single contact stored in the list.

Parameters

chatId	64 bit unsigned integer

Returns

5.5.4.15 GetContactByUserID()

```
Contact EncryptedMessaging.Contacts.GetContactByUserID (
            ulong userId )
```

Get contacts by user Id, if not present, returns a null value.

Parameters

user↩ Id	

Returns

5.5.4.16 GetContacts()

```
Observable< Contact > EncryptedMessaging.Contacts.GetContacts ( )
```

Get a observable contact list.

Returns

5.5.4.17 GetGroupParicipantContacts()

```
List< Contact > EncryptedMessaging.Contacts.GetGroupParicipantContacts (
            Contact contact )
```

Look for contacts in the group list and add new partipants if not present already.

Parameters

contact	>The list of source contact to create a group

Returns

5.5.4.18 GetMyContact()

```
Contact EncryptedMessaging.Contacts.GetMyContact (
            bool nameless = false )
```

Create Contact user data in the contact is null.

Parameters

nameless	boolean

Returns

5.5.4.19 GetParticipant()

```
Contact EncryptedMessaging.Contacts.GetParticipant (
            byte[] key )
```

Look for a contact based on his public key, if not present in the book, a null value will be returned; Each contact is a group of at least 2 people: The user and the recipients; Contacts with more than one recipient are groups and will not be taken into consideration for research.

Parameters

key	Public key

Returns

5.5.4.20 GetParticipantName()

```
string EncryptedMessaging.Contacts.GetParticipantName (
            byte[] key )
```

If the participant is present in the address book, he returns his name, otherwise he invents a name

Parameters

key	Public key

Returns

5.5.4.21 GetParticipants()

```
List< byte[]> EncryptedMessaging.Contacts.GetParticipants (
            ulong chatId )
```

Get the List of contacts in the group.

Parameters

chatId	64 bit unsigned integer

Returns

5.5.4.22 LastMessagChanged()

```
delegate void EncryptedMessaging.Contacts.LastMessagChanged (
            Message message )
```

Server side rule for last message changed.

Parameters

message	string

5.5.4.23 OnContactAddedHandler()

```
delegate void EncryptedMessaging.Contacts.OnContactAddedHandler (
            Contact contact )
```

Delegate for the event that is triggered when a contact is added.

Parameters

contact	The new contact added

5.5.4.24 Pseudonym() [1/3]

```
string EncryptedMessaging.Contacts.Pseudonym (
            List< byte[]> participants )
```

Assign a Pseudonym for a participant in the list.

Parameters

participants	

Returns

5.5.4.25 Pseudonym() [2/3]

```
string EncryptedMessaging.Contacts.Pseudonym (
            string publicKeys )
```

Convert public key to participants.

Parameters

publicKeys	

Returns

5.5.4.26 Pseudonym() [3/3]

```
string EncryptedMessaging.Contacts.Pseudonym (
            ulong userId )
```

Assign a Pseudonym for a participant for the unisgned long integer input in the list.

Parameters

user↩ Id	Unsigned long integer

Returns

5.5.4.27 RemoveContact() [1/2]

```
void EncryptedMessaging.Contacts.RemoveContact (
            Contact contact )
```

Remove the contact from the address book.

Parameters

contact	

5.5.4.28 RemoveContact() [2/2]

```
void EncryptedMessaging.Contacts.RemoveContact (
            string key )
```

Remove the contact from the address book.

Parameters

key	Public key

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Contacts.cs

5.6 EncryptedMessaging.Context Class Reference

Our mission is to exacerbate the concept of security in messaging and create something conceptually new and innovative from a technical point of view. Top-level encrypted communication (there is no backend , there is no server-side contact list, there is no server but a simple router, the theory is that if the server does not exist then the server cannot be hacked, the communication is anonymous, the IDs are derived from a hash of the public keys, therefore in no case it is possible to trace who originates the messages, the encryption key is changed for each single message, and a system of digital signatures guarantees the origin of the messages and prevents attacks "men in de middle"). We use different concepts introduced with Bitcoin technology and the library itself: there are no accounts, the account is simply a pair of public and private keys, groups are also supported, the group ID is derived from a hash computed through the public keys of the members, since the hash process is irreversible, the level of anonymity is maximum). The publication of the source wants to demonstrate the genuineness of the concepts we have adopted! Thanks for your attention!

Public Member Functions

- Context (string entryPoint, string networkName="testnet", bool multipleChatModes=false, string privateKeyOrPassphrase=null, bool isServer=false, bool? internetAccess=null, Action< Action > invokeOnMainThread=null, Func< string, string > getSecureKeyValue=null, SecureStorage.Initializer.SetKeyKalueSecure setSecureKeyValue=null, Func< string > getFirebaseToken=null, Func< string > getAppleDeviceToken=null, string cloudPath=null)

  This method initializes the network. You can join the network as a node, and contribute to decentralization, or hook yourself to the network as an external user. To create a node, set the MyAddress parameter with your web address.If MyAddress is not set then you are an external user.
- delegate void OnMessageArrived (Message message)

  Delegate for the action to be taken when messages arrive
- delegate void ViewMessageUi (Message message, bool isMyMessage)

  Function delegated with the event that creates the message visible in the user interface. This function will then be called whenever a message needs to be drawn in the chat. Server-type host systems that don't have messages to render in chat probably don't need to set this action
- delegate void OnContactEventDelegate (Message message)

  This delegate allows you to set up a event that will be called whenever a message arrives from a contact.
- delegate void LastReadedTimeChangeEvent (Contact contact, ulong participantId, DateTime lastRadTime)

  Event that is raised to inform when someone has read a sent message
- delegate void MessageDeliveredEvent (Contact contact, DateTime deliveredTime, bool isMy)

  Delegate for the event that notifies when messages are sent
- delegate void AlertMessage (string text)

  Delegate action for an alert message.
- delegate bool ShareTextMessage (string text)

  Delegate action for a Sharing text message.

Static Public Member Functions

- static void OnConnectivityChange (bool Connectivity)

  Function that must be called whenever the host system has a change of state on the connection. This parameter must be set when starting the application. If it is not set, the libraries do not know if there are changes in the state of the internet connection, and the messages could remain in the queue without being sent.
- static void ReEstablishConnection (bool iMSureThereIsConnection=false)

  Function that reactivates the connection when it is lost. Its use is designed for all those situations in which the connection could be interrupted, for example mobile applications can interrupt the connection when they are placed in the background. When the application returns to the foreground it is advisable to call this comondo to reactivate the connection.
  If this method is not called, the mobile application returns to the foreground, it could stop working and stop receiving messages, while notifications could arrive anyway if routed with Firebase or other external services.

Public Attributes

- Setting **Setting**

  Configuration functions of the message settings.
- SecureStorage.Initializer **SecureStorage**

  Initialize securestorage class.
- Messaging **Messaging**

  Functions related to the messaging functionality of the users.
- TimeSpan **SessionTimeout**

  Timespan for session expiry.
- My **My**

Access all information of the user using the application.

- Contacts **Contacts**

  Contacts functionality associated with the groups.

- readonly Action< Action > **InvokeOnMainThread**

  Use this property to call the main thread when needed: The main thread must be used whenever the user interface needs to be updated, for example, any operation on an ObservableCollection that changes elements must be done by the main thread, otherwise rendering on the graphical interface will generate an error.

- ICloudManager **CloudManager**

  Set up a cloud during context initialization if you want to use cloud features to save avatarms, contacts and other data

Static Public Attributes

- static Uri **EntryPoint**

  The entry point server, to access the network

Properties

- bool **IsServer** `[get]`

  Boolean for server.

- bool **IsConnected** `[get]`

  Returns the current status of the connection with the router/server

Events

- OnMessageArrived **OnNotification**

  Delegate that runs automatically when messages are received. On systems that have a stable connection (server or desktop), this event can be used to generate notifications. Note: Only messages that have viewable content in the chat trigger this event

- ViewMessageUi **ViewMessage**

  It is the function delegate who writes a message in the chat. This function must be set when the App() class is initialized in the common project.

- Action< Message > **OnContactEvent**

  This delegate allows you to set up a event that will be called whenever a system message arrives. Messages that have a graphical display in the chat do not trigger this event. Use OnMessageArrived to intercept incoming messages that have a content display in the chat

- LastReadedTimeChangeEvent OnLastReadedTimeChange

  Event that is performed when a contact reads a message that has been sent

- MessageDeliveredEvent **OnMessageDelivered**

  Event that occurs when a message has been sent. Use this event to notify the host application when a notification needs to be sent to the recipient.

- static Action< Context > **OnContextIsInitialized**

  Function that is called when the context has been fully initialized. If you want to automate something after context initialization, you can do so by assigning an action to this value!

5.6.1 Detailed Description

Our mission is to exacerbate the concept of security in messaging and create something conceptually new and innovative from a technical point of view. Top-level encrypted communication (there is no backend , there is no server-side contact list, there is no server but a simple router, the theory is that if the server does not exist then the server cannot be hacked, the communication is anonymous, the IDs are derived from a hash of the public keys, therefore in no case it is possible to trace who originates the messages, the encryption key is changed for each single message, and a system of digital signatures guarantees the origin of the messages and prevents attacks "men in de middle"). We use different concepts introduced with Bitcoin technology and the library itself: there are no accounts, the account is simply a pair of public and private keys, groups are also supported, the group ID is derived from a hash computed through the public keys of the members, since the hash process is irreversible, the level of anonymity is maximum). The publication of the source wants to demonstrate the genuineness of the concepts we have adopted! Thanks for your attention!

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Context()

```
EncryptedMessaging.Context.Context (
            string entryPoint,
            string networkName = "testnet",
            bool multipleChatModes = false,
            string privateKeyOrPassphrase = null,
            bool isServer = false,
            bool?  internetAccess = null,
            Action< Action > invokeOnMainThread = null,
            Func< string, string > getSecureKeyValue = null,
            SecureStorage.Initializer.SetKeyKalueSecure setSecureKeyValue = null,
            Func< string > getFirebaseToken = null,
            Func< string > getAppleDeviceToken = null,
            string cloudPath = null )
```

This method initializes the network. You can join the network as a node, and contribute to decentralization, or hook yourself to the network as an external user. To create a node, set the MyAddress parameter with your web address.If MyAddress is not set then you are an external user.

Parameters

entryPoint	The entry point server, to access the network
networkName	The name of the infrastructure. For tests we recommend using "testnet"
multipleChatModes	If this mode is enabled there will be multiple chat rooms simultaneously, all archived messages will be preloaded with the initialization of this library, this involves a large use of memory but a better user experience. Otherwise, only one char room will be managed at a time, archived messages will be loaded only when you enter the chat, this mode consumes less memory.
privateKeyOrPassphrase	
isServer	Indicates if a server is initialized: The server has some differences compared to the device which are: It does not store contacts (contacts are acquired during the session and when it expires they are deleted, so the contacts are not even synchronized on the cloud, there is no is the backup and restore that instead occurs for applications on devices).

Parameters

internetAccess	True if network is available
invokeOnMainThread	Method that starts the main thread: Actions that have consequences with updating the user interface must run on the main thread otherwise they cause a crash
getSecureKeyValue	System secure function to read passwords and keys saved with the corresponding set function
setSecureKeyValue	System secure function for saving passwords and keys
getFirebaseToken	Function to get FirebaseToken (the function is passed and not the value, so as not to block the main thread as this sometimes takes a long time). FirebaseToken is used by firebase, to send notifications to a specific device. The sender needs this information to make the notification appear to the recipient.
getAppleDeviceToken	Function to get AppleDeviceToken (the function is passed and not the value, so as not to block the main thread as this sometimes takes a long time). In ios AppleDeviceToken is used to generate notifications for the device. Whoever sends the encrypted message needs this data to generate a notification on the device of who will receive the message.
cloudPath	Specify the location of the cloud directory (where it saves and reads files), if you don't want to use the system one. The cloud is used only in server mode

5.6.3 Member Function Documentation

5.6.3.1 AlertMessage()

```
delegate void EncryptedMessaging.Context.AlertMessage (
            string text )
```

Delegate action for an alert message.

Parameters

text	string

5.6.3.2 LastReadedTimeChangeEvent()

```
delegate void EncryptedMessaging.Context.LastReadedTimeChangeEvent (
            Contact contact,
            ulong participantId,
            DateTime lastRadTime )
```

Event that is raised to inform when someone has read a sent message

Parameters

contact	Contact (group or single user))

Parameters

participant↩ Id	ID of participant who has read
lastRadTime	When the last reading took place

5.6.3.3 MessageDeliveredEvent()

```
delegate void EncryptedMessaging.Context.MessageDeliveredEvent (
            Contact contact,
            DateTime deliveredTime,
            bool isMy )
```

Delegate for the event that notifies when messages are sent

Parameters

contact	Contact (group or single user)
deliveredTime	When message was delivered.
isMy	Boolean

5.6.3.4 OnConnectivityChange()

```
static void EncryptedMessaging.Context.OnConnectivityChange (
            bool Connectivity )  [static]
```

Function that must be called whenever the host system has a change of state on the connection. This parameter must be set when starting the application. If it is not set, the libraries do not know if there are changes in the state of the internet connection, and the messages could remain in the queue without being sent.

Parameters

Connectivity	Network connecttion status true or false

5.6.3.5 OnContactEventDelegate()

```
delegate void EncryptedMessaging.Context.OnContactEventDelegate (
            Message message )
```

This delegate allows you to set up a event that will be called whenever a message arrives from a contact.

Parameters

message	

5.6.3.6 OnMessageArrived()

```
delegate void EncryptedMessaging.Context.OnMessageArrived (
            Message message )
```

Delegate for the action to be taken when messages arrive

Parameters

message	Message

5.6.3.7 ReEstablishConnection()

```
static void EncryptedMessaging.Context.ReEstablishConnection (
            bool iMSureThereIsConnection = false )  [static]
```

Function that reactivates the connection when it is lost. Its use is designed for all those situations in which the connection could be interrupted, for example mobile applications can interrupt the connection when they are placed in the background. When the application returns to the foreground it is advisable to call this comondo to reactivate the connection.
If this method is not called, the mobile application returns to the foreground, it could stop working and stop receiving messages, while notifications could arrive anyway if routed with Firebase or other external services.

Parameters

iMSureThereIsConnection	

5.6.3.8 ShareTextMessage()

```
delegate bool EncryptedMessaging.Context.ShareTextMessage (
            string text )
```

Delegate action for a Sharing text message.

Parameters

text	string

Returns

5.6.3.9 ViewMessageUi()

```
delegate void EncryptedMessaging.Context.ViewMessageUi (
            Message message,
            bool isMyMessage )
```

Function delegated with the event that creates the message visible in the user interface. This function will then be called whenever a message needs to be drawn in the chat. Server-type host systems that don't have messages to render in chat probably don't need to set this action

Parameters

message	The message to render in the chat view
isMyMessage	True if you call it to render my message

5.6.4 Event Documentation

5.6.4.1 OnLastReadedTimeChange

```
LastReadedTimeChangeEvent EncryptedMessaging.Context.OnLastReadedTimeChange
```

Event that is performed when a contact reads a message that has been sent

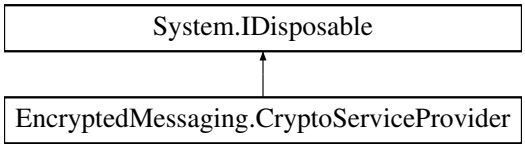
The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Context.cs

5.7 EncryptedMessaging.CryptoServiceProvider Class Reference

This class handles the encryption and decryption of the passphrases and validations.

Inheritance diagram for EncryptedMessaging.CryptoServiceProvider:



Public Member Functions

- CryptoServiceProvider (byte[ ] key=null)

  The ImportCspBlob method initializes the key data of an AsymmetricAlgorithm object using a blob that is compatible with the unmanaged Microsoft Cryptographic API (CAPI).
- CryptoServiceProvider (string passphrase)

  Instance the object
- string GetPrivateKeyBase58 ()

  Gets the private key through wallet import format.
- bool VerifyHash (byte[ ] hash256, byte[ ] signature)

  Verifies that a digital signature is valid by determining the hash value in the signature using the specified hash algorithm and padding, and comparing it to the provided hash value.
- byte[ ] SignHash (byte[ ] hash256)

  Generates a digital signature for the specified hash value.
- byte[ ] ExportCspBlob (bool includePrivateKey)

  Exports a blob that contains the key information associated with an AsymmetricAlgorithm privatekey.
- string GetPassphrase ()

  Get the passPhrase property, If the string is empty return null.
- void ImportCspBlob (byte[ ] key)

  The ImportCspBlob method initializes the key data of an AsymmetricAlgorithm object using a blob that is compatible with the unmanaged Microsoft Cryptographic API (CAPI).
- byte[ ] Encrypt (byte[ ] data)

  Encrypts the private key.
- byte[ ] Decrypt (byte[ ] encryptedData)

  Decrypts the private key.
- bool IsValid ()

  Checks validity and returns boolean.
- void **Dispose** ()

  Public implementation of Dispose pattern callable by consumers.

Static Public Member Functions

- static byte[ ] ComputeHash (byte[ ] data)

  Computes the hash value for the specified byte array.

Protected Member Functions

- virtual void Dispose (bool disposing)

  Protected implementation of Dispose pattern.

5.7.1 Detailed Description

This class handles the encryption and decryption of the passphrases and validations.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 CryptoServiceProvider() [1/2]

```
EncryptedMessaging.CryptoServiceProvider.CryptoServiceProvider (
            byte[] key = null )
```

The ImportCspBlob method initializes the key data of an AsymmetricAlgorithm object using a blob that is compatible with the unmanaged Microsoft Cryptographic API (CAPI).

Parameters

key	

5.7.2.2 CryptoServiceProvider() [2/2]

```
EncryptedMessaging.CryptoServiceProvider.CryptoServiceProvider (
            string passphrase )
```

Instance the object

Parameters

passphrase	Passphrase or private key base 64

5.7.3 Member Function Documentation

5.7.3.1 ComputeHash()

```
static byte[] EncryptedMessaging.CryptoServiceProvider.ComputeHash (
            byte[] data )  [static]
```

Computes the hash value for the specified byte array.

Parameters

data	Combined packages

Returns

Byte array

5.7.3.2 Decrypt()

```
byte[] EncryptedMessaging.CryptoServiceProvider.Decrypt (
            byte[] encryptedData )
```

Decrypts the private key.

Parameters

encryptedData	

Returns

5.7.3.3 Dispose()

```
virtual void EncryptedMessaging.CryptoServiceProvider.Dispose (
            bool disposing )  [protected], [virtual]
```

Protected implementation of Dispose pattern.

Parameters

disposing	

5.7.3.4 Encrypt()

```
byte[] EncryptedMessaging.CryptoServiceProvider.Encrypt (
            byte[] data )
```

Encrypts the private key.

Parameters

data	

Returns

5.7.3.5 ExportCspBlob()

```
byte[] EncryptedMessaging.CryptoServiceProvider.ExportCspBlob (
            bool includePrivateKey )
```

Exports a blob that contains the key information associated with an AsymmetricAlgorithm privatekey.

Parameters

includePrivateKey	

Returns

5.7.3.6 GetPassphrase()

```
string EncryptedMessaging.CryptoServiceProvider.GetPassphrase ( )
```

Get the passPhrase property, If the string is empty return null.

Returns

5.7.3.7 GetPrivateKeyBase58()

```
string EncryptedMessaging.CryptoServiceProvider.GetPrivateKeyBase58 ( )
```

Gets the private key through wallet import format.

Returns

5.7.3.8 ImportCspBlob()

```
void EncryptedMessaging.CryptoServiceProvider.ImportCspBlob (
            byte[] key )
```

The ImportCspBlob method initializes the key data of an AsymmetricAlgorithm object using a blob that is compatible with the unmanaged Microsoft Cryptographic API (CAPI).

Parameters

key	A byte array that represents an asymmetric key blob.

5.7.3.9 IsValid()

```
bool EncryptedMessaging.CryptoServiceProvider.IsValid ( )
```

Checks validity and returns boolean.

Returns

5.7.3.10 SignHash()

```
byte[] EncryptedMessaging.CryptoServiceProvider.SignHash (
            byte[] hash256 )
```

Generates a digital signature for the specified hash value.

Parameters

hash256	The hash value of the data that is being signed.

Returns

5.7.3.11 VerifyHash()

```
bool EncryptedMessaging.CryptoServiceProvider.VerifyHash (
            byte[] hash256,
            byte[] signature )
```

Verifies that a digital signature is valid by determining the hash value in the signature using the specified hash algorithm and padding, and comparing it to the provided hash value.

Parameters

hash256	The hash value of the signed data.
signature	The signature data to be verified.

Returns

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/CryptoServiceProvider.cs

5.8 EncryptedMessaging.ICloudManager Interface Reference

Public Member Functions

- void SaveDataOnCloud (string type, string name, byte[ ] data, bool shared=false)

  Save a data to the cloud in a set type, with a name key
- void LoadDataFromCloud (string type, string name, int? ifSizeIsDifferent=null, bool shared=false)

  Sends a previously saved data request command, and if it exists an event will be generated OnDataLoad
- void LoadAllDataFromCloud (string type, bool shared=false)

  Upload from the cloud all the data saved in a specific type group. An event OnDataLoad will be generated for each data
- void DeleteDataOnCloud (string type, string name, bool shared=false)

  Delete a data that has been saved on the cloud
- void OnLoadData (string type, string name, bool shared, byte[ ] data)

  Load the data from the saved storage.

Events

- EventHandler **OnDataLoad**

  Represents the onDataLoad method that will be handled when the event provides the data.

5.8.1 Detailed Description

```
</summary>
```

5.8.2 Member Function Documentation

5.8.2.1 DeleteDataOnCloud()

```
void EncryptedMessaging.ICloudManager.DeleteDataOnCloud (
            string type,
            string name,
            bool shared = false )
```

Delete a data that has been saved on the cloud

Parameters

type	The group to which the data belongs
name	The unique key assigned to the object
shared	If true, an object in the common area will be deleted, otherwise an object will be deleted from the private area accessible only to the current user

5.8.2.2 LoadAllDataFromCloud()

```
void EncryptedMessaging.ICloudManager.LoadAllDataFromCloud (
            string type,
            bool shared = false )
```

Upload from the cloud all the data saved in a specific type group. An event OnDataLoad will be generated for each data

Parameters

type	The group to which the data belongs
shared	If true, load the data from a common area among all contacts, otherwise they will be load from a private area accessible only to the current user

5.8.2.3 LoadDataFromCloud()

```
void EncryptedMessaging.ICloudManager.LoadDataFromCloud (
            string type,
            string name,
            int?  ifSizeIsDifferent = null,
            bool shared = false )
```

Sends a previously saved data request command, and if it exists an event will be generated OnDataLoad

Parameters

type	The group to which the data belongs
name	The unique key assigned to the object
ifSizeIsDifferent	Upload the data only if the size has changed (It is an empirical method to avoid creating communication traffic for data we already have, it would be more correct to use a hash, but this creates a computational load on the cloud)
shared	If true, load the data from a common area among all contacts, otherwise they will be load from a private area accessible only to the current user

5.8.2.4 OnLoadData()

```
void EncryptedMessaging.ICloudManager.OnLoadData (
            string type,
            string name,
            bool shared,
            byte[] data )
```

Load the data from the saved storage.

Parameters

type	The group to which the data belongs
name	The unique key assigned to the object
shared	If true,load the data from a common area among all contacts, otherwise they will be load from a private area accessible only to the current user
data	An array of data to save

5.8.2.5 SaveDataOnCloud()

```
void EncryptedMessaging.ICloudManager.SaveDataOnCloud (
            string type,
            string name,
            byte[] data,
            bool shared = false )
```

Save a data to the cloud in a set type, with a name key

Parameters

type	The group to which the data belongs
name	The unique key assigned to the object
data	An array of data to save
shared	If true, save the data in a common area among all contacts, otherwise they will be saved in a private area accessible only to the current user

The documentation for this interface was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/CloudManager.cs

5.9 EncryptedMessaging.Message Class Reference

Here are all the functions necessary to process messages in binary format, create messages and read them.

Public Member Functions

- Message (Context context, Contact contact, MessageFormat.MessageType type, byte[ ] author, DateTime creation, byte[ ] data, DateTime receptionTime, ulong postId, bool encrypted, ulong chatId, ulong authorId)

  Message properties.
- byte[ ] **GetAuthor** ()

  This array is the identifier of the chat participant. If you want to get the corresponding contact, you have to use the function Contacts.GetParticipant(Author), If the contact is not in the address book you receive a null value
- string AuthorName ()

  If the sender is present in the address book, it returns its name, otherwise it invents a name. If the message is encrypted this value is null, then use AuthorId to know the author.

- byte[ ] GetData ()

  Returns the message data, however it may return null if the private key is changed or if I contact is deleted If you don't need messages when rendering the UI, but you think you need them in the future, don't use this function but use GetDataFunction to get the data when you need it (so you will save the ram memory)
- bool GetSubApplicationCommandWithParameters (out ushort appId, out ushort command, out List< byte[ ]> parameters)

  Read the commands and parameters sent by a sub application via Messaging.SendCommandToSubApplication(↩ Contact, ushort, ushort, bool, bool, byte[ ][ ])
- bool GetSubApplicationCommandWithData (out ushort appId, out ushort command, out byte[ ] data)

  Read the commands and data sent by a sub application via Messaging.SendCommandToSubApplication(Contact, ushort, ushort, bool, bo
- bool GetSubApplicationCommand (out ushort appId, out ushort command, out byte[ ] data, out List< byte[ ]> parameters)

  Read the commands sent by a sub application via Messaging.SendCommandToSubApplication(Contact, ushort, ushort, bool, bool, byte[ ] or Messaging.SendCommandToSubApplication(Contact, ushort, ushort, bool, bool, byte[ ][ ]) It is an alternative method to the methods GetSubApplicationCommandWithData and GetSubApplicationCommandWithParameters
- void GetShareEncryptedContentData (out string contentType, out byte[ ] privateKey, out string description, out string serverUrl)

  Read messages generated by Messaging.ShareEncryptedContent(Contact, string, byte[ ], string, string) , containing shared and encrypted material on the server. It is used to share Videos or other huge files which cannot be sent with the messaging protocol
- Func< byte[ ]> GetDataFunction ()

  If you need to access the data after the UI has been drawn, you can get this function and memorize it in your event For example, if you have to show a video, or if you have to play audio when the user clicks on the message in the chat
- void Delete (bool alsoDeleteRemote=true)

  Delete the post.

Public Attributes

- Context **Context**

  Context implementation.
- readonly ulong **AuthorId**

  The author Id NOTE: This value is setting only in not encrypted message!
- readonly ulong **ChatId**

  Unique identification number of the chat.
- bool **Encrypted**

  Indicates whether this message was forwarded with encryption

Properties

- Contact **Contact** `[getset]`

  The chat group.
- MessageFormat.MessageType **Type** `[getset]`

  The public key of the author.
- DateTime **Creation** `[getset]`

  Creation date and time UTC format
- DateTime **ReceptionTime** `[getset]`

  Reception date and time UTC format
- ulong? **ReplyToPostId** `[getset]`

  If different from null it indicates that this is a reply post previously sent in the group that the contact represents
- ulong **PostId** `[get]`

  Unique identifier that indicates the message in a chat. Use this property in all contexts where you need to identify a message, for example, when replying to a specific message, you are referring to the message with the ID of the message you want to reply to.
- string? **Translation** `[getset]`

  Translation set for the chat with the recipent.

5.9.1 Detailed Description

Here are all the functions necessary to process messages in binary format, create messages and read them.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Message()

```
EncryptedMessaging.Message.Message (
            Context context,
            Contact contact,
            MessageFormat.MessageType type,
            byte[] author,
            DateTime creation,
            byte[] data,
            DateTime receptionTime,
            ulong postId,
            bool encrypted,
            ulong chatId,
            ulong authorId )
```

Message properties.

Parameters

context	Context
contact	Recipent
type	Type
author	Sender
creation	Date/time of sending
data	Byte array
receptionTime	Date/time of delivery
postId	Unsigned long integer
encrypted	Boolean
chatId	Unsigned long integer
authorId	Unsigned long integer

5.9.3 Member Function Documentation

5.9.3.1 AuthorName()

```
string EncryptedMessaging.Message.AuthorName ( )
```

If the sender is present in the address book, it returns its name, otherwise it invents a name. If the message is encrypted this value is null, then use AuthorId to know the author.

Returns

5.9.3.2 Delete()

```
void EncryptedMessaging.Message.Delete (
            bool alsoDeleteRemote = true )
```

Delete the post.

Parameters

alsoDeleteRemote	Boolean

5.9.3.3 GetData()

```
byte[] EncryptedMessaging.Message.GetData ( )
```

Returns the message data, however it may return null if the private key is changed or if I contact is deleted If you don't need messages when rendering the UI, but you think you need them in the future, don't use this function but use GetDataFunction to get the data when you need it (so you will save the ram memory)

Returns

5.9.3.4 GetDataFunction()

```
Func< byte[]> EncryptedMessaging.Message.GetDataFunction ( )
```

If you need to access the data after the UI has been drawn, you can get this function and memorize it in your event For example, if you have to show a video, or if you have to play audio when the user clicks on the message in the chat

Returns

5.9.3.5 GetShareEncryptedContentData()

```
void EncryptedMessaging.Message.GetShareEncryptedContentData (
            out string contentType,
            out byte[] privateKey,
            out string description,
            out string serverUrl )
```

Read messages generated by Messaging.ShareEncryptedContent(Contact, string, byte[], string, string) , containing shared and encrypted material on the server. It is used to share Videos or other huge files which cannot be sent with the messaging protocol

Parameters

contentType	Three characters describing the type of content being shared. Use the three characters of the file expansion, for example: MP4, DOC, PDF, ISO, etc ...
privateKey	The private key to decrypt the content.
description	Literal description of content
serverUrl	The URL (max 256 char) of the server where the shared content resides. The name of the file is not necessary because it is obtained from the private key. If this value is allowed, then the server will be the default one

5.9.3.6 GetSubApplicationCommand()

```
bool EncryptedMessaging.Message.GetSubApplicationCommand (
            out ushort appId,
            out ushort command,
            out byte[] data,
            out List< byte[]> parameters )
```

Read the commands sent by a sub application via Messaging.SendCommandToSubApplication(Contact, ushort, ushort, bool, bool, by or Messaging.SendCommandToSubApplication(Contact, ushort, ushort, bool, bool, byte[][]) It is an alternative method to the methods GetSubApplicationCommandWithData and GetSubApplicationCommandWithParameters

Parameters

appId	Sub application Id (plugin Id)
command	Id of the command used in the protocol of the sub application
data	The data that was sent with the command
parameters	The parameters that were sent with the command

Returns

False if the received message is not of the SubApplicationCommandWithData type

5.9.3.7 GetSubApplicationCommandWithData()

```
bool EncryptedMessaging.Message.GetSubApplicationCommandWithData (
            out ushort appId,
            out ushort command,
            out byte[] data )
```

Read the commands and data sent by a sub application via Messaging.SendCommandToSubApplication(Contact, ushort, ushort, bool

Parameters

appId	Sub application Id (plugin Id)
command	Id of the command used in the protocol of the sub application
data	The data that was sent with the command

Returns

False if the received message is not of the SubApplicationCommandWithData type

5.9.3.8 GetSubApplicationCommandWithParameters()

```
bool EncryptedMessaging.Message.GetSubApplicationCommandWithParameters (
            out ushort appId,
            out ushort command,
            out List< byte[]> parameters )
```

Read the commands and parameters sent by a sub application via Messaging.SendCommandToSubApplication(↩ Contact, ushort, ushort, bool, bool, byte[ ][ ])

Parameters

appId	Sub application Id (plugin Id)
command	Id of the command used in the protocol of the sub application
parameters	The parameters that were sent with the command

Returns

False if the received message is not of the SubApplicationCommandWithParameters type

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/MessageFormat.cs

5.10 EncryptedMessaging.MessageFormat Class Reference

Message settings set for the recpients.

Public Types

- enum MessageType : byte {
  Text , Image , Audio , Contact ,
  AudioCall , VideoCall , Location , PdfDocument ,
  LastReading , Delete , SmallData , Data ,
  NameChange , ContactStatus , Binary , Inform ,
  PhoneContact , StartAudioGroupCall , StartVideoGroupCall , EndCall ,
  DeclinedCall , SubApplicationCommandWithData , SubApplicationCommandWithParameters , ShareEncryptedContent
  ,
  ReplyToMessage }

  It is a numerator indicating the data type of the message. It is used to indicate what the transmitted or received data refers to. Each message can contain the data in a byte array, and the typo allows you to correctly reconstruct the message by taking the data.
- enum InformType : byte { AvatarHasUpdated }

  Used to notify an event

Public Member Functions

- MessageFormat (Context context)

  Set context for the message format.
- bool ReadDataPost (byte[ ] dataPost, ulong chatId, DateTime receptionTime, out Message message, bool isNewPost=false)

  Reads a post in binary format with the encrypted data and turns it into a clear message
- byte[ ] CreateDataPost (MessageType type, byte[ ] data, List< byte[ ]> participants, out DateTime creation↩ Date, ulong? replyToPostId, bool encrypted=true)

  This feature creates an encrypted post for chat participants. It must therefore be sent to the server in order to be distributed to all.
- byte[ ] CreateDataPostUnencrypted (MessageType type, byte[ ] data, ulong[ ] usersId, out DateTime creationDate, ulong? replyToPostId)

  This feature creates an unencrypted post for chat participants. It must therefore be sent to the server in order to be distributed to all.

5.10.1 Detailed Description

Message settings set for the recpients.

5.10.2 Member Enumeration Documentation

5.10.2.1 InformType

```
enum EncryptedMessaging.MessageFormat.InformType : byte
```

Used to notify an event

Enumerator

AvatarHasUpdated	This info is sent to the contact to inform me that my avatar has been changed

5.10.2.2 MessageType

```
enum EncryptedMessaging.MessageFormat.MessageType : byte
```

It is a numerator indicating the data type of the message. It is used to indicate what the transmitted or received data refers to. Each message can contain the data in a byte array, and the typo allows you to correctly reconstruct the message by taking the data.

Enumerator

Text	Text in unicode format
Image	An image

Enumerator

Audio	An audio message mp3
Contact	A contact to add to the address book
AudioCall	Obsolete
VideoCall	Obsolete
Location	Geographic location
PdfDocument	PDF document
LastReading	Used to notify that messages have been read
Delete	Asks to delete a message
SmallData	Array of bytes[], Each bytes cannot exceed 256 bytes - Use the Functions.SplitIncomingData method to get the array
Data	Array of bytes[] - Use the Functions.SplitIncomingData method to get the array
NameChange	Used to send the name with which the contract is registered in my address book, so I can have notifications with the name used locally in my contacts
ContactStatus	Used to check the status of the contact
Binary	Message type that is used in form
Inform	Used to send simple information (See the InformType enumerator for a list of the information it passes)
PhoneContact	Contact stored on th phone
StartAudioGroupCall	Obsolete
StartVideoGroupCall	Obsolete
EndCall	Obsolete
DeclinedCall	Obsolete
SubApplicationCommandWithData	Used to send commands for sub applications (plugins, modules, additional features): Each sub application has an ID and a command that must be sent with the messaging protocol. See: "<see cref="Messaging.SendCommandToSubApplication(EncryptedMessaging.Contact, ushort, ushort, bool, bool, byte[])"/>"
SubApplicationCommandWithParameters	Used to send commands for sub applications (plugins, modules, additional features): Each sub application has an ID and a command that must be sent with the messaging protocol. See: "<see cref="Messaging.SendCommandToSubApplication(EncryptedMessaging.Contact, ushort, ushort, bool, bool, byte[][])"/>"
ShareEncryptedContent	
ReplyToMessage	Used internally to indicate that the message is a reply to another previous message

5.10.3 Constructor & Destructor Documentation

5.10.3.1 MessageFormat()

```
EncryptedMessaging.MessageFormat.MessageFormat (
            Context context )
```

Set context for the message format.

Parameters

context	

5.10.4 Member Function Documentation

5.10.4.1 CreateDataPost()

```
byte[] EncryptedMessaging.MessageFormat.CreateDataPost (
            MessageType type,
            byte[] data,
            List< byte[]> participants,
            out DateTime creationDate,
            ulong?  replyToPostId,
            bool encrypted = true )
```

This feature creates an encrypted post for chat participants. It must therefore be sent to the server in order to be distributed to all.

Parameters

type	Message type
data	Message data
participants	Public keys of each participant
creationDate	Return current time GTM
replyTo↩ PostId	The post Id property of the message you want to reply to
encrypted	if you need to send a message to someone who does not have the sender's contact in the address book, or the data is already encrypted, it is possible with this parameter to delete the encryption. Users are not allowed to receive unencrypted messages, this function is specific for messages to servers or cloud systems

Returns

Data post

5.10.4.2 CreateDataPostUnencrypted()

```
byte[] EncryptedMessaging.MessageFormat.CreateDataPostUnencrypted (
            MessageType type,
            byte[] data,
```

```
            ulong[] usersId,
            out DateTime creationDate,
            ulong?  replyToPostId )
```

This feature creates an unencrypted post for chat participants. It must therefore be sent to the server in order to be distributed to all.

Parameters

type	Message type
data	Message data
usersId	User ID of each participant
creationDate	Return current time GTM
replyTo↩ PostId	The post Id property of the message you want to reply to

Returns

Data post

5.10.4.3 ReadDataPost()

```
bool EncryptedMessaging.MessageFormat.ReadDataPost (
            byte[] dataPost,
            ulong chatId,
            DateTime receptionTime,
            out Message message,
            bool isNewPost = false )
```

Reads a post in binary format with the encrypted data and turns it into a clear message

Parameters

dataPost	Post in binary format
chatId	Chat Id
receptionTime	Reception Time UTC format
message	
isNewPost	

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/MessageFormat.cs

5.11 EncryptedMessaging.Contact.MessageInfo Class Reference

Set message information on creation and sent.

Public Member Functions

- **MessageInfo** ()

  Message information function for creation and dataId.

Public Attributes

- long **Creation**

  Integer data type for when a message is created
- uint **DataId**

  unsigned integer data id.

5.11.1 Detailed Description

Set message information on creation and sent.

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Contact.cs

5.12 EncryptedMessaging.Messaging Class Reference

This class contains all the functions related to the messaging functionality for the users.

Classes

- class **SendMessageParameters**

Public Member Functions

- Messaging (Context context, bool multipleChatModes)

  Class initializer (prepares for use)
- void SetMultipleChatModes (bool value)

  By default set to false, as only one messaging chat is handled.
- void SendMessage (MessageType type, byte[ ] data, Contact toContact, ulong? replyToPostId=null, ulong? chatId=null, ulong[ ] toIdUsers=null, bool directlyWithoutSpooler=false, bool encrypted=true, bool isLogin=false)

  Send the message to all participants in the current chat room, and save a copy of the local storage
- void SendText (string text, Contact toContact, ulong? replyToPostId=null)

  Send a text
- void SendPicture (byte[ ] png, Contact toContact, ulong? replyToPostId=null)

  For images, the only format allowed is PNG. The longest side of the image must not exceed 800 pixels, for example 800 X 480, or 480 X 800 (in no case any side must exceed 800 pixels in length)
- void SendInfo (InformType inform, Contact toContact, ulong? chatId=null, ulong[ ] toIdUsers=null, bool directlyWithoutSpooler=false, bool encrypted=true)

  Send a info

- void SendBinary (Contact toContact, byte[ ] binaryData, bool directlyWithoutSpooler=false, bool encrypted=true)

  Send a binary data block To read the data on the target client/server: var values = Functions.SplitData(true,data);
- void SendBinaryUnencrypetd (ulong chatId, ulong[ ] toIdUsers, byte[ ] binaryData, bool directlyWithoutSpooler=false)

  Send a binary unencrypted (Clients are only able to receive encrypted messages: do not use this command to send messages to communicate between clients) To read the data on the target client/server: var values = Functions.SplitData(true,data);
- void SendSmallData (Contact toContact, bool directlyWithoutSpooler=false, bool encrypted=true, params byte[ ][ ] values)

  Send array of bytes, not exceeding 255 bytes To read the data on the target client: var values = Functions.SplitData(true,data);
- void SendData (Contact toContact, bool directlyWithoutSpooler=false, bool encrypted=true, params byte[ ][ ] values)

  Send array of bytes To read the data on the target client: var values = Functions.SplitData(false,data);
- void SendSmallDataToCloud (bool directlyWithoutSpooler=false, bool encrypted=true, params byte[ ][ ] values)

  Send array of bytes, not exceeding 255 bytes To read the data on the target client: var values = Functions.SplitData(data);
- void SendKeyValueCollection (Contact toContact, bool directlyWithoutSpooler=false, params Tuple< byte, byte[ ]>[ ] keyValue)

  Sends a sequence of key-values, where the key is a byte and the value is an array of 256 bytes To read the data on the target client: var values = Functions.SplitData(data);
- void SendKeyValueCollection (Contact toContact, bool directlyWithoutSpooler=false, bool valueMustBeLessOf256Bytes=false, params Tuple< byte, byte[ ]>[ ] keyValue)

  Sends a sequence of key-values, where the key is a byte and the value a byte array To read the data on the target client: var values = Functions.SplitData(data);
- void SendKeyValueCollectionToCloud (bool directlyWithoutSpooler=false, params Tuple< byte, byte[ ]>[ ] keyValue)

  Sends a sequence of key-values, where the key is a byte and the value is an array of max 256 bytes To read the data on the target client: var values = Functions.SplitData(data);
- void SendKeyValueCollectionToCloud (bool directlyWithoutSpooler=false, bool valueMustBeLessOf256Bytes=false, params Tuple< byte, byte[ ]>[ ] keyValue)

  Sends a sequence of key-values, where the key is a byte and the value is an array of bytes To read the data on the target client: var values = Functions.SplitData(data);
- void LoginToServer (bool directlyWithoutSpooler, Contact onServer)

  The servers don't have the client's public key because they don't have the contact list. The login consists in sending your contact to the server, so that it can have the public key to communicate in encrypted form. If there are no more communications, the server will automatically remove the contact, so in the future it will be necessary to log in again
- void SendAudio (byte[ ] mp3, Contact toContact, ulong? replyToPostId=null)

  The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.
- void SendContact (Contact contact, Contact toContact, bool directlyWithoutSpooler=false, bool purposeIsUpdateOnly=false, bool isLogin=false)

  The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.
- void NotifyContactNameChange (Contact toContact)

  // I send the name with which the contract is registered in my address book, so I can have notifications with the name used locally in my contacts
- void SendContactStatus (Contact toContact)
- void SendAudioCall (byte[ ] call, Contact toContact)

  The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.
- void SendVideoCall (byte[ ] call, Contact toContact)

  The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.
- void SendStartAudioGroupCall (byte[ ] call, Contact toContact)

  Start a group audio call for the chat room.

- void SendStartVideoGroupCall (byte[ ] call, Contact toContact)

  Start a group video call for the chat room.
- void SendEndCall (byte[ ] call, Contact toContact)

  End a group call for the chat room.
- void SendDeclinedCall (byte[ ] call, Contact toContact)

  Decline a call from the recipent.
- void SendLocation (double latitude, double longitude, Contact toContact, ulong? replyToPostId=null)

  Submit a geographic location
- void SendPdfDocument (byte[ ] document, Contact toContact, ulong? replyToPostId=null)

  Send a document of the format pdf.
- void SendPhoneContact (byte[ ] phoneContact, Contact toContact, ulong? replyToPostId=null)

  Send a contact card to the recipent
- void SendCommandToSubApplication (Contact toContact, ushort appId, ushort command, bool directly↩ WithoutSpooler=false, bool encrypted=true, params byte[ ][ ] values)

  This command allows sub-applications (plugins, modules, extensions) to send commands with parameters. Use the "<see cref="Message.GetSubApplicationCommandWithParameters(out ushort, out ushort, out List{byte[]})"/>" method of the Message class to read this command on the receiving device
- void SendCommandToSubApplication (Contact toContact, ushort appId, ushort command, bool directly↩ WithoutSpooler=false, bool encrypted=true, byte[ ] data=null)

  This command allows sub-applications (plugins, modules, extensions) to send commands with data. Use the "<see cref="Message.GetSubApplicationCommandWithData(out ushort, out ushort, out byte[ ])"/>" method of the Message class to read this command on the receiving device
- void ShareEncryptedContent (Contact toContact, string contentType, byte[ ] privateKey, string description, string serverUrl=null)

  Share encrypted content on the server with other contacts. Use the "<see cref="Message.GetShareEncrypted↩ ContentData(out string, out byte[ ], out string, out string)"/>" method of the Message class to read this command on the receiving device

Properties

- Contact? **CurrentChatRoom** `[getset]`

  Get the current chat group and set the time of last viewed and if message read.

5.12.1 Detailed Description

This class contains all the functions related to the messaging functionality for the users.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Messaging()

```
EncryptedMessaging.Messaging.Messaging (
            Context context,
            bool multipleChatModes )
```

Class initializer (prepares for use)

Parameters

context	Context
multipleChatModes	If false, one messaging chat will be handled at a time

5.12.3 Member Function Documentation

5.12.3.1 LoginToServer()

```
void EncryptedMessaging.Messaging.LoginToServer (
            bool directlyWithoutSpooler,
            Contact onServer )
```

The servers don't have the client's public key because they don't have the contact list. The login consists in sending your contact to the server, so that it can have the public key to communicate in encrypted form. If there are no more communications, the server will automatically remove the contact, so in the future it will be necessary to log in again

Parameters

directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
onServer	Server to login

5.12.3.2 NotifyContactNameChange()

```
void EncryptedMessaging.Messaging.NotifyContactNameChange (
            Contact toContact )
```

// I send the name with which the contract is registered in my address book, so I can have notifications with the name used locally in my contacts

Parameters

toContact	The recipient

5.12.3.3 SendAudio()

```
void EncryptedMessaging.Messaging.SendAudio (
            byte[] mp3,
```

```
            Contact toContact,
            ulong?  replyToPostId = null )
```

The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.

Parameters

mp3	mp3 64 kbps file
toContact	The recipient
replyTo↩ PostId	The post Id property of the message you want to reply to

5.12.3.4 SendAudioCall()

```
void EncryptedMessaging.Messaging.SendAudioCall (
            byte[] call,
            Contact toContact )
```

The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.

Parameters

call	audio call
toContact	The recipient

5.12.3.5 SendBinary()

```
void EncryptedMessaging.Messaging.SendBinary (
            Contact toContact,
            byte[] binaryData,
            bool directlyWithoutSpooler = false,
            bool encrypted = true )
```

Send a binary data block To read the data on the target client/server: var values = Functions.SplitData(true,data);

Parameters

toContact	The recipient
binaryData	Binary data block to send
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it

5.12.3.6 SendBinaryUnencrypetd()

```
void EncryptedMessaging.Messaging.SendBinaryUnencrypetd (
            ulong chatId,
            ulong[] toIdUsers,
            byte[] binaryData,
            bool directlyWithoutSpooler = false )
```

Send a binary unencrypted (Clients are only able to receive encrypted messages: do not use this command to send messages to communicate between clients) To read the data on the target client/server: var values = Functions.↩ SplitData(true,data);

Parameters

toIdUsers	The recipients
binaryData	Binary data block to send
chatId	Id of the chat
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost

5.12.3.7 SendCommandToSubApplication() [1/2]

```
void EncryptedMessaging.Messaging.SendCommandToSubApplication (
            Contact toContact,
            ushort appId,
            ushort command,
            bool directlyWithoutSpooler = false,
            bool encrypted = true,
            byte[] data = null )
```

This command allows sub-applications (plugins, modules, extensions) to send commands with data. Use the "<see cref="Message.GetSubApplicationCommandWithData(out ushort, out ushort, out byte[])"/>" method of the Message class to read this command on the receiving device

Parameters

toContact	Recipient
appId	Sub application Id (plugin Id)
command	Id of the command used in the protocol of the sub application
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it
data	Data relating to the command sent. NOTE: if you intend to send an array of data, use the other overload

5.12.3.8 SendCommandToSubApplication() [2/2]

```
void EncryptedMessaging.Messaging.SendCommandToSubApplication (
            Contact toContact,
            ushort appId,
            ushort command,
            bool directlyWithoutSpooler = false,
            bool encrypted = true,
            params byte values[][] )
```

This command allows sub-applications (plugins, modules, extensions) to send commands with parameters. Use the "<see cref="Message.GetSubApplicationCommandWithParameters(out ushort, out ushort, out List{byte[]})"/>" method of the Message class to read this command on the receiving device

Parameters

toContact	Recipient
appId	Sub application Id (plugin Id)
command	Id of the command used in the protocol of the sub application
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it
values	Data blocks (Command parameters to use in the plugin or extension). NOTE: If you intend to send single data (not an array of parameters), use the other overload

5.12.3.9 SendContact()

```
void EncryptedMessaging.Messaging.SendContact (
            Contact contact,
            Contact toContact,
            bool directlyWithoutSpooler = false,
            bool purposeIsUpdateOnly = false,
            bool isLogin = false )
```

The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.

Parameters

contact	mp3 64 kbps file
toContact	The recipient
directlyWithoutSpooler	
purposeIsUpdateOnly	
isLogin	Flad used only for the login command to avoid a recursive loop

5.12.3.10 SendContactStatus()

```
void EncryptedMessaging.Messaging.SendContactStatus (
            Contact toContact )
```

Parameters

toContact	

5.12.3.11 SendData()

```
void EncryptedMessaging.Messaging.SendData (
            Contact toContact,
            bool directlyWithoutSpooler = false,
            bool encrypted = true,
            params byte values[][] )
```

Send array of bytes To read the data on the target client: var values = Functions.SplitData(false,data);

Parameters

toContact	The recipient
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it
values	Data blocks not exceeding 255 bytes each

5.12.3.12 SendDeclinedCall()

```
void EncryptedMessaging.Messaging.SendDeclinedCall (
            byte[] call,
            Contact toContact )
```

Decline a call from the recipent.

Parameters

call	
toContact	

5.12.3.13 SendEndCall()

```
void EncryptedMessaging.Messaging.SendEndCall (
            byte[] call,
            Contact toContact )
```

End a group call for the chat room.

Parameters

call	Audio/Video call
toContact	The recipient

5.12.3.14 SendInfo()

```
void EncryptedMessaging.Messaging.SendInfo (
            InformType inform,
            Contact toContact,
            ulong?  chatId = null,
            ulong[] toIdUsers = null,
            bool directlyWithoutSpooler = false,
            bool encrypted = true )
```

Send a info

Parameters

inform	Info type
toContact	The recipient
chatId	Set this value if toContact is null, that is, if the message is not encrypted
toIdUsers	Id of the members of the group the message is intended for. ToContact and toIdUsers cannot be set simultaneously
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it

5.12.3.15 SendKeyValueCollection() [1/2]

```
void EncryptedMessaging.Messaging.SendKeyValueCollection (
            Contact toContact,
```

```
            bool directlyWithoutSpooler = false,
            bool valueMustBeLessOf256Bytes = false,
            params Tuple< byte, byte[]>[] keyValue )
```

Sends a sequence of key-values, where the key is a byte and the value a byte array To read the data on the target client: var values = Functions.SplitData(data);

Parameters

toContact	The recipient
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
valueMustBeLessOf256Bytes	If true: Ideal for saving values no larger than 256 bytes
keyValue	Data blocks of data

5.12.3.16 SendKeyValueCollection() [2/2]

```
void EncryptedMessaging.Messaging.SendKeyValueCollection (
            Contact toContact,
            bool directlyWithoutSpooler = false,
            params Tuple< byte, byte[]>[] keyValue )
```

Sends a sequence of key-values, where the key is a byte and the value is an array of 256 bytes To read the data on the target client: var values = Functions.SplitData(data);

Parameters

toContact	The recipient
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
keyValue	Data blocks not exceeding 255 bytes each

5.12.3.17 SendKeyValueCollectionToCloud() [1/2]

```
void EncryptedMessaging.Messaging.SendKeyValueCollectionToCloud (
            bool directlyWithoutSpooler = false,
            bool valueMustBeLessOf256Bytes = false,
            params Tuple< byte, byte[]>[] keyValue )
```

Sends a sequence of key-values, where the key is a byte and the value is an array of bytes To read the data on the target client: var values = Functions.SplitData(data);

Parameters

directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost

Parameters

valueMustBeLessOf256Bytes	Limits the length of the saved values by saving communication bandwidth for data which is generally small
keyValue	Data blocks, not exceeding 255 bytes each, if specified by the valueMustBeLessOf256Bytes parameter

5.12.3.18 SendKeyValueCollectionToCloud() [2/2]

```
void EncryptedMessaging.Messaging.SendKeyValueCollectionToCloud (
            bool directlyWithoutSpooler = false,
            params Tuple< byte, byte[]>[] keyValue )
```

Sends a sequence of key-values, where the key is a byte and the value is an array of max 256 bytes To read the data on the target client: var values = Functions.SplitData(data);

Parameters

directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
keyValue	Data blocks not exceeding 255 bytes each

5.12.3.19 SendLocation()

```
void EncryptedMessaging.Messaging.SendLocation (
            double latitude,
            double longitude,
            Contact toContact,
            ulong?  replyToPostId = null )
```

Submit a geographic location

Parameters

latitude	
longitude	
toContact	
replyTo↩ PostId	

5.12.3.20 SendMessage()

```
void EncryptedMessaging.Messaging.SendMessage (
            MessageType type,
```

```
            byte[] data,
            Contact toContact,
            ulong?  replyToPostId = null,
            ulong?  chatId = null,
            ulong[] toIdUsers = null,
            bool directlyWithoutSpooler = false,
            bool encrypted = true,
            bool isLogin = false )
```

Send the message to all participants in the current chat room, and save a copy of the local storage

Parameters

type	The type of data
data	The body of the data in binary format
toContact	Recipient of the message. ToContact and toIdUsers cannot be set simultaneousl
replyToPostId	The post Id property of the message you want to reply to
chatId	Set this value if toContact is null, that is, if the message is not encrypted
toIdUsers	Id of the members of the group the message is intended for. ToContact and toIdUsers cannot be set simultaneously
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it
isLogin	Flad used only for the login command to avoid a recursive loop

5.12.3.21 SendPdfDocument()

```
void EncryptedMessaging.Messaging.SendPdfDocument (
            byte[] document,
            Contact toContact,
            ulong?  replyToPostId = null )
```

Send a document of the format pdf.

Parameters

document	file to send
toContact	The recipient
replyTo↩ PostId	

5.12.3.22 SendPhoneContact()

```
void EncryptedMessaging.Messaging.SendPhoneContact (
```

```
            byte[] phoneContact,
            Contact toContact,
            ulong?  replyToPostId = null )
```

Send a contact card to the recipent

Parameters

phoneContact	Contact card
toContact	The recipient
replyToPostId	

5.12.3.23 SendPicture()

```
void EncryptedMessaging.Messaging.SendPicture (
            byte[] png,
            Contact toContact,
            ulong?  replyToPostId = null )
```

For images, the only format allowed is PNG. The longest side of the image must not exceed 800 pixels, for example 800 X 480, or 480 X 800 (in no case any side must exceed 800 pixels in length)

Parameters

png	PNG file, maximum side size = 800 pix
toContact	The recipient
replyTo↩ PostId	The post Id property of the message you want to reply to

5.12.3.24 SendSmallData()

```
void EncryptedMessaging.Messaging.SendSmallData (
            Contact toContact,
            bool directlyWithoutSpooler = false,
            bool encrypted = true,
            params byte values[][] )
```

Send array of bytes, not exceeding 255 bytes To read the data on the target client: var values = Functions.Split↩ Data(true,data);

Parameters

toContact	The recipient
directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost

Parameters

encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it
values	Data blocks not exceeding 255 bytes each

5.12.3.25 SendSmallDataToCloud()

```
void EncryptedMessaging.Messaging.SendSmallDataToCloud (
            bool directlyWithoutSpooler = false,
            bool encrypted = true,
            params byte values[][] )
```

Send array of bytes, not exceeding 255 bytes To read the data on the target client: var values = Functions.Split↩Data(data);

Parameters

directlyWithoutSpooler	If this parameter is true, the data will be sent immediately without any reception check, if the recipient is not on-line they will be lost
encrypted	Clients are only able to receive encrypted messages. Non-encrypted messages are reserved for communications with cloud servers if the data is already encrypted and does not require a second encryption and if the message must be delivered to a server that does not have the client in the address book and therefore could not otherwise read it
values	Data blocks not exceeding 255 bytes each

5.12.3.26 SendStartAudioGroupCall()

```
void EncryptedMessaging.Messaging.SendStartAudioGroupCall (
            byte[] call,
            Contact toContact )
```

Start a group audio call for the chat room.

Parameters

call	Audio call
toContact	The recipients

5.12.3.27 SendStartVideoGroupCall()

```
void EncryptedMessaging.Messaging.SendStartVideoGroupCall (
            byte[] call,
            Contact toContact )
```

Start a group video call for the chat room.

Parameters

call	Video call
toContact	The recipients

5.12.3.28 SendText()

```
void EncryptedMessaging.Messaging.SendText (
            string text,
            Contact toContact,
            ulong?  replyToPostId = null )
```

Send a text

Parameters

text	The text to send
toContact	The recipient
replyTo↩ PostId	The post Id property of the message you want to reply to

5.12.3.29 SendVideoCall()

```
void EncryptedMessaging.Messaging.SendVideoCall (
            byte[] call,
            Contact toContact )
```

The only type of audio file allowed is mp3, with a speed of 64 k bps or lower.

Parameters

call	audio call
toContact	The recipient

5.12.3.30 SetMultipleChatModes()

```
void EncryptedMessaging.Messaging.SetMultipleChatModes (
            bool value )
```

By default set to false, as only one messaging chat is handled.

Parameters

value	Boolean

5.12.3.31 ShareEncryptedContent()

```
void EncryptedMessaging.Messaging.ShareEncryptedContent (
            Contact toContact,
            string contentType,
            byte[] privateKey,
            string description,
            string serverUrl = null )
```

Share encrypted content on the server with other contacts. Use the "<see cref="Message.GetShareEncrypted↩ ContentData(out string, out byte[], out string, out string)"/>" method of the Message class to read this command on the receiving device

Parameters

toContact	Recipient
contentType	Three characters describing the type of content being shared. Use the three characters of the file expansion, for example: MP4, DOC, PDF, ISO, etc ...
privateKey	The private key to decrypt the content
description	Literal description of content
serverUrl	The URL (max 256 char) of the server where the shared content resides. The name of the file is not necessary because it is obtained from the private key. If this value is allowed, then the server will be the default one

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Messaging.cs

5.13 EncryptedMessaging.My Class Reference

This class allows you to access all the information about the user who is using the application: the contact, if his cryptographic keys, the tokens of his device for notifications, the user id, etc.

Public Member Functions

- string GetPublicKey ()

  Return the public key of current user in base64 format
- byte[ ] GetPublicKeyBinary ()

  Return the public key of current user in binary format (array of byte)
- byte[ ] GetPrivatKeyBinary ()
- ulong GetId ()

  Get the unisgned long integer User Id from the CSP byte array.
- string GetPrivateKey ()

  Return the private key stored in the device,if not present, it generates one
- string GetPassphrase ()

  Gets the combination of words that allow you to recover the account using bitcoin technology
- byte[ ] GetAvatar ()

  Gets the avatar (the image of the user photo in the form of a byte array)
- void SetAvatar (byte[ ] png)

  Sets the avatar (the image of the user photo in the form of a byte array)

Properties

- Contact **Contact** `[get]`

  Gets my contact (the contact of the user using the application)
- CryptoServiceProvider? Csp `[get]`

  Return a CSP of current user
- bool **IsServer** `[get]`

  Boolean set for the Server parameter.
- string **Name** `[getset]`

  Set or set the current username, the data is saved on the cloud in an anonymous and encrypted way if an application usage limit is exceeded.
- string **FirebaseToken** `[get]`

  It is used by firebase, to send notifications to a specific device. The sender needs this information to make the notification appear to the recipient.
- string **DeviceToken** `[get]`

  In ios this is used to generate notifications for the device. Whoever sends the encrypted message needs this data to generate a notification on the device of who will receive the message.

5.13.1 Detailed Description

This class allows you to access all the information about the user who is using the application: the contact, if his cryptographic keys, the tokens of his device for notifications, the user id, etc.

5.13.2 Member Function Documentation

5.13.2.1 GetAvatar()

```
byte[] EncryptedMessaging.My.GetAvatar ( )
```

Gets the avatar (the image of the user photo in the form of a byte array)

Returns

5.13.2.2 GetId()

```
ulong EncryptedMessaging.My.GetId ( )
```

Get the unisgned long integer User Id from the CSP byte array.

Returns

5.13.2.3 GetPassphrase()

```
string EncryptedMessaging.My.GetPassphrase ( )
```

Gets the combination of words that allow you to recover the account using bitcoin technology

Returns

Passphrase

5.13.2.4 GetPrivateKey()

```
string EncryptedMessaging.My.GetPrivateKey ( )
```

Return the private key stored in the device,if not present, it generates one

Returns

5.13.2.5 GetPrivatKeyBinary()

```
byte[] EncryptedMessaging.My.GetPrivatKeyBinary ( )
```

Returns

5.13.2.6 GetPublicKey()

```
string EncryptedMessaging.My.GetPublicKey ( )
```

Return the public key of current user in base64 format

Returns

5.13.2.7 GetPublicKeyBinary()

```
byte[] EncryptedMessaging.My.GetPublicKeyBinary ( )
```

Return the public key of current user in binary format (array of byte)

Returns

5.13.2.8 SetAvatar()

```
void EncryptedMessaging.My.SetAvatar (
            byte[] png )
```

Sets the avatar (the image of the user photo in the form of a byte array)

Parameters

png	

5.13.3 Property Documentation

5.13.3.1 Csp

```
CryptoServiceProvider?  EncryptedMessaging.My.Csp  [get]
```

Return a CSP of current user

Returns

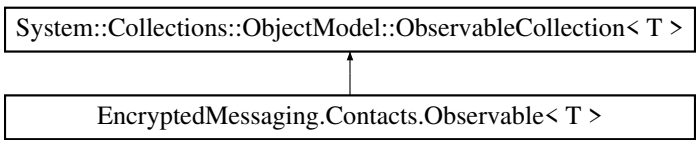
The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/My.cs

5.14 EncryptedMessaging.Contacts.Observable< T > Class Template Reference

Add new participants to the group.

Inheritance diagram for EncryptedMessaging.Contacts.Observable< T >:



Public Member Functions

- void Update (IEnumerable< T > list)
  Update the partipants list.

Protected Member Functions

- override void OnCollectionChanged (System.Collections.Specialized.NotifyCollectionChangedEventArgs e)
  Fucntion for change in the partipants in the group.

5.14.1 Detailed Description

Add new participants to the group.

Template Parameters

T	

5.14.2 Member Function Documentation

5.14.2.1 OnCollectionChanged()

```
override void EncryptedMessaging.Contacts.Observable< T >.OnCollectionChanged (
            System.Collections.Specialized.NotifyCollectionChangedEventArgs e )  [protected]
```

Fucntion for change in the partipants in the group.

Parameters

e	

5.14.2.2 Update()

```
void EncryptedMessaging.Contacts.Observable< T >.Update (
            IEnumerable< T > list )
```

Update the partipants list.

Parameters

list	list

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Contacts.cs

5.15 EncryptedMessaging.ContactMessage.Properties Class Reference

Strings used in this class.

Public Attributes

- string **Name**

NAme of the group.

- string **Language**

  Language.
- byte[ ] **Key**

  Byte array for key.
- Contact.RuntimePlatform **Os**

  Operating system.
- string **FirebaseToken**

  Firebase Token.
- string **DeviceToken**

  Device token for IOS.

5.15.1 Detailed Description

Strings used in this class.

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/ContactMessage.cs

5.16 EncryptedMessaging.Contact.RemoteReaded Class Reference

Add a timestamp to the message.

Public Member Functions

- **RemoteReaded** ()

  Add a timestamp to the message.

Public Attributes

- ulong **IdParticipant**

  unsigned integer value.
- long **TimeStamp**

  integer value for time.

Properties

- DateTime **DateTime** `[getset]`

  Update the timestamp on the message.

5.16.1 Detailed Description

Add a timestamp to the message.

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Contact.cs

5.17 EncryptedMessaging.Repository Class Reference

This library provides functions to retrieve messages on the server and store them locally.

Public Member Functions

- Repository (Context context)

  Display libraries used as read only.
- void AddPost (byte[ ] dataByteArray, ulong chatId, ref DateTime receptionDate)

  Add the encrypted post to local storage and return the reception date
- DateTime ReadPosts (ulong chatId, Action< byte[ ], DateTime > action=null, DateTime receprion↩ Antecedent=default, int? take=null, List< DateTime > exclude=null)

  Read all posts of a given chat and perform an action. If no action is specified, then the default action will be to show messages in the chat
- byte[ ] ReadLastPost (ulong chatId, out DateTime receptionDateTime)

  Get the last post written in a given chat
- Message GetLastMessageViewable (ulong chatId, out DateTime receptionDateTime)

  Gets the last visible post (which can be viewed in the chat, therefore system messages that do not produce anything visible are excluded)
- byte[ ] ReadPost (DateTime receptionDateTime, ulong chatId)

  Read a post in the chat having the time to receive
- byte[ ] ReadPostByPostId (ulong postId, ulong chatId)

  Read a post given its ID
- void DeletePost (DateTime receptionDateTime, Contact contact)

  Permanently delete a post saved in storage
- void DeletePostByPostId (ulong postId, ulong chatId)

  Delete a post given its identifier
- void DeletePostByPostId (ulong postId, Contact contact)

  Delete a post given its Post Id.
- void ClearPosts (Contact contact)

  Erase all content in a chat

Static Public Member Functions

- static DateTime GetDateTimeOfData (byte[ ] data)

  Get the Date and time of the data from the byte array.
- static int GetTimestampOfData (byte[ ] data)

  Get time stamp from the byte array.
- static ulong PostId (byte[ ] dataPost)

  Convert the byte array to Unsigned 64 integer if length is satsifed by the condition.

Static Public Attributes

- const int **MaxPostLength** = 20971520

  Set maximum post length to 20 MB.

5.17.1 Detailed Description

This library provides functions to retrieve messages on the server and store them locally.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 Repository()

```
EncryptedMessaging.Repository.Repository (
            Context context )
```

Display libraries used as read only.

Parameters

context	

5.17.3 Member Function Documentation

5.17.3.1 AddPost()

```
void EncryptedMessaging.Repository.AddPost (
            byte[] dataByteArray,
            ulong chatId,
            ref DateTime receptionDate )
```

Add the encrypted post to local storage and return the reception date

Parameters

dataByteArray	
chatId	
receptionDate	return reception date

Returns

5.17.3.2 ClearPosts()

```
void EncryptedMessaging.Repository.ClearPosts (
            Contact contact )
```

Erase all content in a chat

Parameters

contact	The contact representing the chat you want to reset

5.17.3.3 DeletePost()

```
void EncryptedMessaging.Repository.DeletePost (
            DateTime receptionDateTime,
            Contact contact )
```

Permanently delete a post saved in storage

Parameters

receptionDateTime	The date of receipt of the post you want to delete
contact	The conversation group (contact) in which the post was written

5.17.3.4 DeletePostByPostId() [1/2]

```
void EncryptedMessaging.Repository.DeletePostByPostId (
            ulong postId,
            Contact contact )
```

Delete a post given its Post Id.

Parameters

postId	Delete a post given its identifier
contact	The chat in which to search for the post, identified by the contact

5.17.3.5 DeletePostByPostId() [2/2]

```
void EncryptedMessaging.Repository.DeletePostByPostId (
            ulong postId,
            ulong chatId )
```

Delete a post given its identifier

Parameters

postId	
chatId	The chat in which to search for the post, identified by the chat ID

5.17.3.6 GetDateTimeOfData()

```
static DateTime EncryptedMessaging.Repository.GetDateTimeOfData (
            byte[] data )  [static]
```

Get the Date and time of the data from the byte array.

Parameters

data	

Returns

5.17.3.7 GetLastMessageViewable()

```
Message EncryptedMessaging.Repository.GetLastMessageViewable (
            ulong chatId,
            out DateTime receptionDateTime )
```

Gets the last visible post (which can be viewed in the chat, therefore system messages that do not produce anything visible are excluded)

Parameters

chatId	The ID of the chat for which you want to get the last message
receptionDateTime	It also returns the date and time when the message was received

Returns

5.17.3.8 GetTimestampOfData()

```
static int EncryptedMessaging.Repository.GetTimestampOfData (
            byte[] data )  [static]
```

Get time stamp from the byte array.

Parameters

data	

Returns

5.17.3.9 PostId()

```
static ulong EncryptedMessaging.Repository.PostId (
            byte[] dataPost )  [static]
```

Convert the byte array to Unsigned 64 integer if length is satsifed by the condition.

Parameters

dataPost	

Returns

5.17.3.10 ReadLastPost()

```
byte[] EncryptedMessaging.Repository.ReadLastPost (
            ulong chatId,
            out DateTime receptionDateTime )
```

Get the last post written in a given chat

Parameters

chatId	The ID of the chat for which you want to get the last post
receptionDateTime	It also returns the date and time when the post was received

Returns

5.17.3.11 ReadPost()

```
byte[] EncryptedMessaging.Repository.ReadPost (
            DateTime receptionDateTime,
            ulong chatId )
```

Read a post in the chat having the time to receive

Parameters

receptionDateTime	The reception time is used as the identifier of the message to formulate the request
chatId	The chat in which you want to search

Returns

5.17.3.12 ReadPostByPostId()

```
byte[] EncryptedMessaging.Repository.ReadPostByPostId (
            ulong postId,
            ulong chatId )
```

Read a post given its ID

Parameters

postId	The identifier of the post you want to read
chatId	The chat in which to search for the post, identified by the chat ID

Returns

If the post you are looking for is not found, null is returned

5.17.3.13 ReadPosts()

```
DateTime EncryptedMessaging.Repository.ReadPosts (
            ulong chatId,
            Action< byte[], DateTime > action = null,
            DateTime receprionAntecedent = default,
            int?  take = null,
            List< DateTime > exclude = null )
```

Read all posts of a given chat and perform an action. If no action is specified, then the default action will be to show messages in the chat

Parameters

chatId	The id of the chat whose posts you want to read
action	Action to be performed for each post, the byte[] is binary data of the encrypted post that is read from the repository, DateTime is the time the post was received which you can use as a unique ID (you can use the ticks property of DateTime as a unique id)
receprionAntecedent	If set, consider only posts that are dated before the value indicated. It is useful for paginating messages in the chat view, or for telling the loading of messages in blocks. How to use this parameter: You need to store the date of the oldest message that is displayed in the chat, when you want to load a second block of messages you have to pass this date in order to get the next block
take	Limit the number of messages to take. If not set, the value set in the Context.Setting.MessagePagination settings will be used. Pass the Context.Setting.KeepPost value to process all messages!
exclude	List of posts to exclude using the received date as a filter

Returns

Returns the date of arrival of the oldest message processed by the function. Use this value to page further requests by passing the "receprionAntecedent" parameter.

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Repository.cs

5.18 EncryptedMessaging.Setting Class Reference

Configuration functions of setting message saving and deleting based on user input.

Public Member Functions

- Setting (Context context)

  Load the settigs of the chats.

Properties

- int **PostPersistenceDays** `[getset]`

  Number of days messages must be kept in memory before being automatically deleted
- int **KeepPost** `[getset]`

  Number of messages to be saved for each chat
- int **MessagePagination** `[getset]`

  Number of messages for each chat page: The chat is divided into pages to speed up the loading and not to weigh down the memory

5.18.1 Detailed Description

Configuration functions of setting message saving and deleting based on user input.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 Setting()

```
EncryptedMessaging.Setting.Setting (
            Context context )
```

Load the settigs of the chats.

Parameters

context	Context

The documentation for this class was generated from the following file:

- C:/Data/AntiGitCode/CryptoMessenger/Crypto-Messenger/EncryptedMessaging/Setting.cs

Index