Contents

1 VolaPG (vpg) Specification

1.1 TODO

- cache already decrypted messages

- decrypt links from registered users only by default; can be turned on for anonymous users (`WARNING`: they can change nicknames)

1.2 Conventions used in this document

- `?` – zero or one, i.e. optional
- `*` – zero or more
- `+` – one or more
- `[a-z]` – character set (i.e. "any of these characters")
- `<expr>{n,m}` – minimum of n expr, maximum of m expr

1.3 Definitions

1.3.1 user

- represents a volafile user, registered or not
- the user's name mustn't contain spaces and must be at least 3 and at most 12 characters long

1.3.1.1 Grammar

- `user_name := [a-zA-Z0-9]{3,12}`

1.3.2 alias

- usually, users will only have one public key but this doesn't have to be the case – that's why aliases are necessary
- an alias is a name for one of the public keys a user uses – it's not easy to remember a hexstring with 1000 characters
- no spaces are allowed in the alias' name
- it's a string produced by concatenating a user's name, `:` character and the alias' name
- a user's name can be an alias too (alias with the name `default` is special) – this is discussed later on in alias list

1.3.2.1 Grammar

- `alias := <user_name> | <user_name> ":" <alias_name>`
- `alias_name := [a-zA-z0-9]+`
- `user_name` – given above

1.3.2.2 Examples

#	alias
1.	`alice:default`
2.	`alice:work`
3.	`joe:home`
4.	`joe`
5.	`bob:shitposting`

1.3.3 alias table

- stores the public keys of a user along with their aliases
- a hash table with (key, value) being (`alias_name`, `pubkey`)
  - `alias_name` – name of the alias that's assigned to a specific public key
  - `pubkey` – the public key

1.3.3.1 Examples

#	user's name	aliases & public keys
1.	`bob`	`default: 135`, `shitposting: 246`
2.	`joe`	`home: 987`

1.3.4 user table

- stores other users' public keys used for encryption
- a table of users' names and their alias tables
- a hash table with (key, value) being (`user_name`, `alias_table`)
  - `user_name` – the name of the user
  - `alias_table` – the alias table of user

1.3.4.1 Examples

#	user's name	aliases & public keys
1.1	`alice`	`default: 123`, `work: 456`, `private: 789`
1.2	`bob`	`default: 135`, `shitposting: 246`
1.3	`joe`	`home: 987`

1.3.5 key pair list

- TODO: fingerprints
- the user running vpg can have multiple key pairs (identities) (analagous to other users having multiple public keys (identities))
- it is a list of (`pubkey`, `privkey`) pairs, each representing a key pair
  - `pubkey` – the public key of the key pair
  - `private` – the private key of the key pair

1.3.6 key list

- a list of public keys that will be used to encrypt a message
- it is generated from an alias list

1.3.7 alias list

- an alias list is used to define which public keys to use when encrypting a message, i.e. it is used to generate a key list from a user table
- it's a string of aliases concatenated with a `+` character
- aliases which don't exist won't be included in the key list
- the alias with the name `default` is special – if a user's name is given instead of an alias, the user's alias table is looked up for an alias with the name `default`
  - i.e. when specifying an alias, `bob` is the same as `bob:default`
  - it is merely a convenience and was introduced because ***most users* will have only one** public key and in that case typing out aliases isn't really useful (they're still used under the hood, however)
  - **`NOTE`: the `default` alias is not required to exist**
-
- TODO: addkey from registered users only by default
- TODO: renamekey
- TODO: filename length for ciphertext files
- TODO: key pair -> key pair

1.3.7.1 Grammar

- `alias_list := <alias> ("+" <alias>)*`
- `alias` – given above

1.3.7.2 Examples (using the user table given in example 1. of user table)

#	alias list	same as	key list
1.	`alice`	`alice:default`	123
2.	`alice+bob`	`alice:default+bob:default`	123, 135
3.	`bob:shitposting+joe:home`	n/a	246, 987
4.	`alice+joe`	`alice:default+joe:default`	123
5.	`joe`	`joe:default`	(empty)

1.3.8 vpg room

- a room to which the files containing ciphertext will be uploaded to

1.3.9 ciphertext file

- contains the ciphertext
- its filename is derived from
  - the users whose public keys were used to encrypt the message
  - time when the file was made (when the ciphertext was produced)

1.3.9.1 Grammar

- `filename := "~VPG~" " -- " <user_list> " -- " <unix_timestamp> ".asc"`
- `user_list := <user_name> (", " <user_name>)*`
- `unix_timestamp` – UNIX time
- `user_name` – given above

1.4 How it works

1. a user encrypts a piece of text with the recipient's public key
2. the ciphertext is put into a file and uploaded to volafile
3. the file has a special name and can be identified by the recipient as a vpg ciphertext file
4. the recipient downloads the file, gets the ciphertext and decrypts it using his private key
5. the recipient can now see the message the user sent him

1.5 Procedures

1.5.1 General

- encryption is triggered by the user via a command
- decryption is triggered automatically when a ciphertext file is identified

1.5.2 Decryption (triggered automatically)

1. the script detects that a ciphertext file was linked in the chat (via the filename)
2. if the filename contains our username
    1. if the ciphertext file hasn't been decrypted before (i.e. if it's not cached)
        1. the ciphertext file is downloaded if its size doesn't exceed 5 KiB
        2. the ciphertext file's content is decrypted with the user's private key
        3. the ciphertext file's filename and the plaintext are cached
    2. the file link in the chat is replaced by the plaintext

1.6 Commands

- commands are invoked with `/vpg <command> <param>*`
- parameters listed for each of the commands below appear in the order they're expected by the command
- `TODO`: move the description of commands somewhere else? section Command explanation?

1.6.1 encrypt

- encrypts a message with the specified keys

1.6.1.1 Parameters

- `alias_list` – defines which keys the message will be encrypted with
- `text` – the text to encrypt; quoted if it has spaces

1.6.1.2 Procedure

1. user runs the encryption command
2. the text is encrypted with the specified public keys
3. a ciphertext file is made
4. the ciphertext file is uploaded to the vpg room
5. a text message containing the link of the ciphertext file is output in the current room

1.6.2 addkey

- adds or updates a user's key in the user table

1.6.2.1 Parameters

- `file` – the file that contains the public key of the user (`TODO`: define what a file is, a full link or just an id like #fhsfhsd)
- `user` – the name of the user
- `alias?` (`default` by default) – the alias to use for this key

1.6.2.2 Procedure

1. the public key is extracted from the specified file
2. if the specified user already exists in our user table
    1. if the specified alias already exists in the specified user's alias table
        1. replace the existing alias with the new alias `alias`
    2. otherwise
        1. create a new alias `alias`
3. otherwise
    1. create a new user `user` and create a new alias `alias` for him

1.6.3 rmkey

- removes either a user's key or the whole user from the user table

1.6.3.1 Parameters

- `user` – the name of the user
- `alias?` (no default) – alias of the public key to remove

1.6.3.2 Procedure

1. if the user `user` exists
    1. if `alias` was specified
        1. if alias `alias` exists
            1. remove alias `alias` from user `user`'s alias table
    2. otherwise
        1. remove the whole user from the user table (`TODO`: confirmation maybe?)

1.6.4 newkeypair

- creates a new key pair (this is so users don't have to download a pgp implementation and do it themselves)

1.6.4.1 Parameters

- `name` – name assigned to the the key pair
- `email` – email assigned to the key pair (`TODO`: is email required?)
- `passphrase` – passphrase used for the private key
- `comment?` (`""` by default) – comment assigned to the key pair

1.6.4.2 Procedure

1. a new key pair is created with the specified parameters
2.

1.6.5 room

- sets a new vpg room

1.6.5.1 Parameters

- `room` – ID of the room that will be set as the vpg room

1.7 Possible implementation

1.7.1 Settings

- whitenames allowed
- vpg room
- user table
- self table
- encrypt with self key by default
- passphrase popup should give info about cert