



Πανεπιστήμιο Πειραιά
Τμήμα Ψηφιακών Συστημάτων

GPGVote

**Ανάπτυξη ενός διαδικτυακού συστήματος
λογισμικού ηλεκτρονικής διεξαγωγής
ψηφοφοριών, βασιζόμενου στο σύστημα
κρυπτογράφησης PGP**

Φοιτητής:
Πέτρος Μωυσιάδης
(Ε/03126)

Επιβλέπων Καθηγητής:
Δρ. Σωκράτης Κάτσικας

Σεπτέμβριος 2010

Περίληψη

Στην παρούσα πτυχιακή εργασία χρησιμοποιούμε την γλώσσα προγραμματισμού *Python* και την πλατφόρμα ανάπτυξης εφαρμογών Παγκόσμιου Ιστού *Django* για να αναπτύξουμε ένα διαδικτυακό σύστημα ηλεκτρονικής διεξαγωγής ψηφοφοριών.

Οι εξουσιοδοτημένοι χρήστες του συστήματος αποκτούν την δυνατότητα να δημιουργούν νέες ψηφοφορίες, να συμμετέχουν ως ψηφοφόροι σε όσες ψηφοφορίες έχουν το δικαίωμα ψήφου, καθώς και να εξακριβώνουν την καταμέτρηση της ψήφου τους στα τελικά αποτελέσματα των ψηφοφοριών.

Για την ικανοποίηση των απαιτήσεων ασφάλειας του συστήματος, όπως για παράδειγμα στη ταυτοποίηση και αυθεντικοποίηση των χρηστών, χρησιμοποιούμε τα κρυπτογραφικά μοντέλα και τις μεθόδους του λογισμικού κρυπτογράφησης GnuPG, το οποίο υλοποιεί πλήρως το πρότυπο OpenPGP.

Περιεχόμενα

Περιεχόμενα	1
1 Εισαγωγή	2
1.1 Η ανάγκη για διαδικτυακές ψηφοφορίες	2
1.2 Σκοπός και στόχοι της εργασίας	3
1.3 Το όνομα του έργου	3
2 Προδιαγραφές	4
2.1 Τεχνολογία Λογισμικού	4
2.2 Απαιτήσεις Ασφάλειας	6
2.3 Επιπλέον ζητήματα Ασφάλειας	7
3 Οργάνωση και Λειτουργίες	8
3.1 Οργάνωση του έργου	8
3.2 Λειτουργίες του συστήματος	10
3.2.1 Λειτουργίες για μη αυθεντικοποιημένους χρήστες	10
3.2.2 Λειτουργίες για αυθεντικοποιημένους χρήστες	13
3.3 Στιγμιότυπα του συστήματος σε λειτουργία	18
4 Συμπεράσματα	26
A Κώδικας	27
A.1 Στατιστικά κώδικα	27
A.2 Κώδικας γενικών λειτουργιών	28
A.3 Εφαρμογή gpgauth	40
A.4 Εφαρμογή polls	50
A.5 python-gnupg patch	77
Βιβλιογραφία	79

Κεφάλαιο 1

Εισαγωγή

1.1 Η ανάγκη για διαδικτυακές ψηφοφορίες

Το Διαδίκτυο, με την ταχεία ανάπτυξη και την ευρεία εξάπλωσή του τα τελευταία χρόνια, έχει αποτελέσει ένα σημαντικό μέσο επικοινωνίας. Στους κόλπους του σχηματίζονται συνεχώς κοινότητες ανθρώπων με κοινά ενδιαφέροντα και στόχους, ενώ παράλληλα αναπτύσσονται νέοι μηχανισμοί συνεργασίας και λήψης κοινών αποφάσεων.

Ωστόσο, η διάσταση απόψεων που συχνά εμφανίζεται σε μικρότερο ή μεγαλύτερο βαθμό ανάμεσα στα μέλη μιας κοινότητας μπορεί να εμποδίζει την δράση της και να διακόπτει την ομαλή της λειτουργία. Κατά συνέπεια, δημιουργείται η ανάγκη για δημιουργία διαδικασιών που διευθετούν τις όποιες διαφωνίες και καταδεικνύουν λύσεις με την μεγαλύτερη δυνατή αποδοχή για τα προβλήματα που προκύπτουν. Βασική επιδίωξη για την ικανοποίηση αυτής της ανάγκης αποτελεί η εξασφάλιση της δημοκρατικότητας των διαδικασιών που οδηγούν στην λήψη αποφάσεων.

Η διεξαγωγή ψηφοφοριών, ως μία από τις πλέον διαδεδομένες διαδικασίες, μπορεί να εξασφαλίσει, υπό προϋποθέσεις, την μεγαλύτερη δυνατή συμμετοχή των μελών μιας κοινότητας στην λήψη αποφάσεων. Προς αυτή την κατεύθυνση, οι ενδιαφερόμενες κοινότητες μπορούν να αξιοποιήσουν τις δυνατότητες του Διαδικτύου και να υιοθετήσουν διαδικασίες ψηφοφοριών μέσω αυτού. Επιπλέον, οι διαδικασίες ψηφοφοριών μέσω Διαδικτύου παρουσιάζουν ξεχωριστό ενδιαφέρον για τις κοινότητες που λόγω της διασποράς των μελών τους έχουν ως βασικό ή αποκλειστικό μέσο επικοινωνίας το Διαδίκτυο, καθώς και για μέλη κοινοτήτων που για κάποιον λόγο δεν είναι εφικτό να συμμετέχουν σε διαδικασίες ψηφοφοριών με τη φυσική τους παρουσία (π.χ. άτομα με ειδικές ανάγκες).

1.2 Σκοπός και στόχοι της εργασίας

Σκοπός αυτής της εργασίας είναι η υλοποίηση ενός συστήματος διεξαγωγής ψηφοφοριών μέσω Διαδικτύου, το οποίο θα μπορεί να χρησιμοποιηθεί για τις ανάγκες μιας κοινότητας στην λήψη αποφάσεων, την ανάδειξη ρόλων, καθώς και στην προβολή των κυρίαρχων απόψεων σχετικά με τα ζητήματα που την αφορούν.

Βασικός στόχος αποτελεί η κάλυψη των απαιτήσεων ασφάλειας που επιλέγουμε για το σύστημα, οι οποίες αναλύονται στην συνέχεια σύμφωνα με την σχετική βιβλιογραφία.

Επιπλέον, στόχος του έργου ανάπτυξης του συστήματος είναι η υιοθέτηση σύγχρονων τεχνολογιών και μεθόδων ανάπτυξης εφαρμογών Παγκόσμιου Ιστού που διευκολύνουν την περαιτέρω ανάπτυξη και επεκτασιμότητα του έργου.

Τέλος, για τις ανάγκες του συστήματος, έχουμε αναπτύξει μια αυτοτελή μονάδα λογισμικού (software module) εξουσιοδότησης (authorization) χρηστών, θέτοντας ως στόχο αυτή να μπορεί να χρησιμοποιηθεί εκ νέου σε άλλα έργα ανάπτυξης λογισμικού.

1.3 Το όνομα του έργου

Όπως συνηθίζεται σε κάθε λογισμικό να αποδίδεται ένα όνομα, έτσι κι εμείς αποδώσαμε στο λογισμικό που αναπτύξαμε το όνομα **GPGVote**. Το πρώτο συστατικό του ονόματος παραπέμπει στο πρόγραμμα GnuPG (ή GPG) στο οποίο, ως επί το πλείστον, βασίζουμε τις παραμέτρους ασφάλειας του συστήματος και το δεύτερο συστατικό σημαίνει “ψήφος” στην αγγλική γλώσσα.

Κεφάλαιο 2

Προδιαγραφές

2.1 Τεχνολογία Λογισμικού

Για την ανάπτυξη του συστήματος χρησιμοποιούμε, επεκτείνουμε και ενσωματώνουμε λογισμικό, πλατφόρμες ανάπτυξης και μονάδες λογισμικού που έχουν αναπτυχθεί από τρίτους. Στην συνέχεια παραθέτουμε συνοπτικές περιγραφές αυτών των τεχνολογιών και τους λόγους επιλογής τους.

Django

Το *Django* είναι μια πλατφόρμα ανάπτυξης εφαρμογών Παγκόσμιου Ιστού (Web development framework). Είναι ελεύθερο λογισμικό που ενθαρρύνει την ταχεία και καθαρή ανάπτυξη διαδικτυακών εφαρμογών σε γλώσσα προγραμματισμού υψηλού επιπέδου *Python*. Ακολουθεί την διαδεδομένη αρχιτεκτονική σχεδιασμού και ανάπτυξης λογισμικού MVC (Model-View-Controller), σύμφωνα με την οποία διαχωρίζεται η λογική της εφαρμογής από την παρουσίαση των διεπαφών χρήστη. Τα χαρακτηριστικά του Django καθιστούν την εφαρμογή μας εύκολα επεκτάσιμη, παραμετροποιήσιμη και αποδοτική. Επίσης, η φιλοσοφία του *django*, όπως αποτυπώνεται στους αυτοματισμούς και τις προγραμματιστικές διεπαφές που παρέχει, μας γλιτώνει από περιττές επαναλήψεις του ίδιου κώδικα σε πολλά σημεία. Χαρακτηριστική είναι η δυνατότητα του Django να μετατρέπει τα μοντέλα των διαφόρων οντοτήτων που δημιουργούμε για τις ανάγκες της εφαρμογής και τις σχέσεις μεταξύ τους, από την γλώσσα *Python* που τα ορίζουμε σε καταχωρήσεις στην βάση δεδομένων που χρησιμοποιούμε, ενώ, παράλληλα, μας παρέχει μια δυναμική προγραμματιστική διεπαφή (API) για την προσπέλαση τους (Object-relational mapper).

GnuPG

Το *GnuPG* (ή *GPG*) αποτελεί ελεύθερο λογισμικό που υλοποιεί πλήρως το πρότυπο κρυπτογράφησης *OpenPGP*. Επιτρέπει την κρυπτογράφηση και την ηλεκτρονική υπογραφή δεδομένων και μηνυμάτων και παρέχει ένα ευέλικτο σύστημα διαχείρισης κλειδιών. Η λειτουργία του ως πρόγραμμα γραμμής εντολών μας επιτρέπει να το ενσωματώνουμε σε άλλες εφαρμογές για να αξιοποιήσουμε τις κρυπτογραφικές μεθόδους του, χωρίς να χρειάζεται να χρησιμοποιήσουμε περίπλοκες βιβλιοθήκες κρυπτογραφίας.

python-gnupg

Η μονάδα λογισμικού *python-gnupg* επιτρέπει την αξιοποίηση των δυνατοτήτων του προγράμματος *GnuPG* με χρήση κώδικα *Python*. Για τις ανάγκες του *GPGVote*, επεκτείναμε την *python-gnupg* με την προσθήκη κάποιων επιπλέον μεθόδων. Στο παράρτημα της εργασίας, εκτός απ' τον κώδικα του *GPGVote* παρέχουμε και το βελτιωτικό αρχείο (patch file) για την *python-gnupg*.

django-simple-captcha

Για να προστατεύσουμε το σύστημα από ανεπιθύμητες καταχωρήσεις από προγράμματα που επιχειρούν επαναληπτικά αυτοματοποιημένες εγγραφές (bots), χρησιμοποιούμε τη μονάδα λογισμικού *django-simple-captcha* η οποία παρουσιάζει στα επίμαχα σημεία ως απαραίτητο πεδίο την εισαγωγή συμβολοσειρών από εικόνες, αναγνωρίσιμες μόνο από ανθρώπους (captcha images).

Στο σημείο αυτό κρίνουμε σκόπιμο να αναφέρουμε πως εξαιτίας της διαπλατφορμικής (cross-platform) αρχιτεκτονικής των εργαλείων και των λογισμικών μονάδων που χρησιμοποιούμε, το *GPGVote* δύναται να εγκατασταθεί σε αρκετά περιβάλλοντα λειτουργικών συστημάτων και συστημάτων βάσεων δεδομένων. Ωστόσο, εμείς έχουμε εργαστεί και δοκιμάσει το σύστημα αποκλειστικά σε περιβάλλον *Unix* (*FreeBSD* και *GNU/Linux*), ενώ ως σύστημα βάσης δεδομένων χρησιμοποιήσαμε το αξιόπιστο σύστημα *PostgreSQL*.

Τέλος, εγκαθιστούμε την εφαρμογή σε περιβάλλον εξυπηρετητή Παγκόσμιου Ιστού (Web Server) *Apache* με χρήση της λειτουργικής μονάδας *mod_wsgi* για την εξυπηρέτηση εφαρμογών *Django/Python*, ενώ ενεργοποιούμε, επίσης, τη λειτουργική μονάδα *mod_ssl* για την προστασία του συστήματός μας με ένα επιπλέον στρώμα ασφάλειας σε επίπεδο ανταλλαγής μηνυμάτων και δεδομένων με τους πελάτες Παγκόσμιου Ιστού (Web Clients) των χρηστών.

2.2 Απαιτήσεις Ασφάλειας

Σύμφωνα με την σχετική βιβλιογραφία, υπάρχει ένα σύνολο απαιτήσεων ασφάλειας που είναι κοινό σε όλα τα συστήματα ηλεκτρονικών ψηφοφοριών και εξασφαλίζει την συμμόρφωση του συστήματος με τις αρχές διεξαγωγής ψηφοφοριών που υιοθετούνται και τους ισχύοντες κανονισμούς. Για τις ανάγκες διεξαγωγής ψηφοφοριών στους κόλπους μιας κοινότητας, το πλαίσιο αυτό μπορεί να καθοριστεί από τα ίδια τα μέλη της κοινότητας, όπως συμβαίνει συνήθως. Παρακάτω περιγράφουμε τις απαιτήσεις ασφάλειας που έχουμε ως στόχο να πληρεί το σύστημα GPGVote, θεωρώντας πως αυτές ανταποκρίνονται στις απαιτήσεις που μπορεί να έχει γενικά μια κοινότητα.

Δημοκρατικότητα: Σε ένα “δημοκρατικό” σύστημα επιτρέπεται να ψηφίσουν μόνο όσοι έχουν το δικαίωμα ψήφου και κάθε ψηφοφόρος μπορεί να ψηφίσει μόνο μία φορά.

Μυστικότητα: Σύμφωνα με αυτήν την απαίτηση κανείς δεν πρέπει να μπορεί να ανακαλύψει τι ψήφισε κάποιος από τους ψηφοφόρους.

Ακρίβεια: Η ακρίβεια ή ορθότητα σε μια ψηφοφορία απαιτεί το ανακοινωθέν αποτέλεσμα της ψηφοφορίας να ταυτίζεται με το πραγματικό αποτέλεσμα. Αυτό σημαίνει πως κανείς δεν μπορεί να αλλάξει την ψήφο κάποιου άλλου και πως όλες οι μη άκυρες ψήφοι καταμετρούνται στο τελικό αποτέλεσμα.

Αμεροληψία: Η αμεροληψία έχει να κάνει με το ότι κανείς δεν δύναται να μάθει το τρέχον αποτέλεσμα της ψηφοφορίας πριν αυτή ολοκληρωθεί και επομένως να χρησιμοποιήσει αυτή τη γνώση για να επηρεάσει υπέρ συγκεκριμένων επιλογών την κρίση των υπόλοιπων ψηφοφόρων που δεν έχουν ψηφίσει ακόμα.

Αποτροπή της εκμαίευσης ψήφων: Σύμφωνα με αυτήν την απαίτηση κανένας παράγοντας δεν πρέπει να μπορεί να εξαναγκάσει κάποιον ψηφοφόρο να αποκαλύψει την ψήφο του.

Επαληθευσιμότητα: Η επαληθευσιμότητα του αποτελέσματος της ψηφοφορίας αφορά την ύπαρξη μηχανισμών που επιτρέπουν σε κάθε ψηφοφόρο να επιβεβαιώσει τόσο την ορθότητα στην καταμέτρηση των ψήφων όσο και το ότι η δική του ψήφος έχει καταμετρηθεί στο τελικό αποτέλεσμα.

Στη συνέχεια, κατά την περιγραφή των λειτουργιών του GPGVote, εξηγούμε πώς και υπό ποιες προϋποθέσεις το σύστημα επιχειρεί να καλύψει τις παραπάνω απαιτήσεις.

2.3 Επιπλέον ζητήματα Ασφάλειας

Στο σημείο αυτό πρέπει να επισημάνουμε πως η κάλυψη των απαιτήσεων ασφάλειας εξαρτάται σε μεγάλο βαθμό από το περιβάλλον στο οποίο εγκαθίσταται το σύστημα καθώς και από αυτούς που αναλαμβάνουν την διαχείριση τόσο του ίδιου του συστήματος όσο και του περιβάλλοντος εγκατάστασης. Επομένως, η υιοθέτηση του συστήματος GPGVote για τις ανάγκες μιας κοινότητας προϋποθέτει πως η ίδια η κοινότητα έχει αναπτύξει μηχανισμούς σύστασης ομάδων υψηλής εμπιστοσύνης για την εγκατάσταση και διαχείριση των εμπλεκόμενων συστημάτων, ενώ έχει επίσης φροντίσει για την ύπαρξη μηχανισμών ελέγχου των ομάδων αυτών. Παράλληλα, ιδιαίτερη μέριμνα χρειάζεται για την ασφάλεια του συστήματος από εξωτερικούς κινδύνους.

Όσον αφορά το τεχνικό σκέλος, λύσεις μπορούν να δοθούν εξωτερικά του συστήματος GPGVote με την δημιουργία ιεραρχημένων ρόλων και δικαιωμάτων πρόσβασης σε επίπεδο λειτουργικού συστήματος, συστήματος βάσεων δεδομένων και διακομιστή ιστοσελίδων, καθώς επίσης και με την ανάπτυξη μηχανισμών ασφάλειας για την προστασία από εσωτερικές και εξωτερικές απειλές και επιθέσεις (π.χ. καθορισμός πολιτικών ασφάλειας, παραμετροποίηση τειχών προστασίας (firewalls) από μη ασφαλή μηνύματα και δίκτυα, ανάπτυξη διαδικασιών καταγραφής των εκτελούμενων ενεργειών κτλ.).

Επιπλέον, εξίσου σημαντικά ζητήματα αποτελούν η διατήρηση της ακεραιότητας των δεδομένων, η όσο το δυνατόν μεγαλύτερη διάρκεια απρόσκοπτης λειτουργίας των συστημάτων και η ύπαρξη μηχανισμών επαναφοράς των συστημάτων σε πλήρη λειτουργία και χωρίς επιπλοκές μετά από απρόβλεπτη διακοπή (π.χ. με την ανάπτυξη συστημάτων δημιουργίας, επικαιροποίησης και επαναφοράς αντιγράφων ασφάλειας).

Στο σύνολό τους αυτά τα ζητήματα συμβάλλουν στην **ανθεκτικότητα** ενός συστήματος ηλεκτρονικών ψηφοφοριών, η οποία μελετάται συχνά στην βιβλιογραφία ως μία ξεχωριστή απαίτηση ασφάλειας. Ωστόσο, στην παρούσα εργασία εστιάζουμε στις λειτουργίες του συστήματος GPGVote που αναπτύξαμε και δεν ασχολούμαστε με αυτά τα ζητήματα παρά μόνο αναφορικά, όπου κρίνουμε σκόπιμο.

Κεφάλαιο 3

Οργάνωση και Λειτουργίες

3.1 Οργάνωση του έργου

Το GPGVote αποτελείται από τις εξής, δύο εφαρμογές Django:

- Την βοηθητική εφαρμογή **gpgauth** για την εγγραφή, αυθεντικοποίηση, εξουσιοδότηση και είσοδο των χρηστών του συστήματος, καθώς και για τη διαχείριση των αντίστοιχων PGP κλειδιών
- Την κύρια εφαρμογή **polls** που υλοποιεί τις βασικές λειτουργίες του συστήματος για την δημιουργία, τροποποίηση και διεξαγωγή των ψηφοφοριών

Οι παραπάνω δύο εφαρμογές χρησιμοποιούν κοινή βάση δεδομένων όπου καταχωρούν και επεξεργάζονται εγγραφές σε πίνακες που δημιουργούνται κατά αντιστοιχία των μοντέλων που ορίζονται σε κάθε εφαρμογή.

Επίσης, τα αρχεία τόσο του έργου όσο και των επιμέρους εφαρμογών του οργανώνονται στην μορφή μονάδας λογισμικού Python, με την εξής δενδρική μορφή:

- **Κατάλογος gpgauth:** Η εφαρμογή gpgauth
 - Αρχείο `__init__.py`: Κενό αρχείο, απαραίτητο για κάθε μονάδα λογισμικού Python
 - Αρχείο `models.py`: Ορίζει τα μοντέλα της εφαρμογής και τις μεταξύ τους σχέσεις
 - Αρχείο `views.py`: Υλοποιεί τις λειτουργίες της εφαρμογής
 - Αρχείο `forms.py`: Ορίζει τις φόρμες εισαγωγής στοιχείων που χρησιμοποιούνται στην εφαρμογή

- Αρχείο `tests.py`: Μπορεί να περιέχει κώδικα δοκιμών, χρήσιμων για την ανάπτυξη της εφαρμογής
- **Κατάλογος `polls`:** Η εφαρμογή `polls`
 - Αρχείο `__init__.py`: Κενό αρχείο, απαραίτητο για κάθε μονάδα λογισμικού Python
 - Αρχείο `models.py`: Ορίζει τα μοντέλα της εφαρμογής και τις μεταξύ τους σχέσεις
 - Αρχείο `views.py`: Υλοποιεί τις λειτουργίες της εφαρμογής
 - Αρχείο `forms.py`: Ορίζει τις φόρμες εισαγωγής στοιχείων που χρησιμοποιούνται στην εφαρμογή
 - Αρχείο `tests.py`: Μπορεί να περιέχει κώδικα δοκιμών, χρήσιμων για την ανάπτυξη της εφαρμογής
- **Κατάλογος `templates`:** Περιέχει αρχεία για την παρουσίαση των διεπαφών χρήστη
 - *Κατάλογος `media`:*
 - * *Κατάλογος `css`:* Αρχεία μορφοποίησης και ορισμού στυλ αντικειμένων HTML
 - * *Κατάλογος `js`:* Αρχεία Javascript
 - * *Κατάλογος `images`:* Αρχεία γραφικών που χρησιμοποιούνται στο σύστημα
 - * Αρχείο `gpgvote_authority_public_key.asc`: Το δημόσιο PGP κλειδί της αρχής ελέγχου και διαχείρισης του GPGVote
 - *Django Templates:* Αρχεία HTML με ενσωματωμένο κώδικα προτυποποίησης (templating) που εκτελεί το Django για την παρουσίαση των διεπαφών χρήστη
- Αρχείο `__init__.py`: Κενό αρχείο, απαραίτητο για κάθε μονάδα λογισμικού Python
- Αρχείο `manage.py`: Υλοποιεί μεθόδους διαχείρισης του έργου
- Αρχείο `settings.py`: Ορίζει τις ρυθμίσεις του συστήματος
- Αρχείο `urls.py`: Αντιστοιχίζει URLs με λειτουργίες του συστήματος
- Αρχείο `views.py`: Υλοποιεί γενικές λειτουργίες του συστήματος

3.2 Λειτουργίες του συστήματος

Οι λειτουργίες του συστήματος διαχωρίζεται σε λειτουργίες για μη αυθεντικοποιημένους χρήστες και λειτουργίες για αυθεντικοποιημένους χρήστες. Μη αυθεντικοποιημένοι χρήστες είναι οι επισκέπτες του δικτυακού τόπου του συστήματος, ενώ αυθεντικοποιημένοι χρήστες είναι οι χρήστες που έχουν εισαχθεί επιτυχώς στο σύστημα.

3.2.1 Λειτουργίες για μη αυθεντικοποιημένους χρήστες

Οι λειτουργίες αυτές υλοποιούνται στην εφαρμογή grgauth και περιλαμβάνουν:

- **Εγγραφή (Register)**

1. Υποβολή PGP κλειδιού (Προστατευόμενη από bots με captcha).
2. Το PGP κλειδί εισάγεται στη βάση κλειδιών GnuPG του συστήματος (*PGP keyring*).

Αν υπάρξει κάποιο πρόβλημα κατά την εισαγωγή, η διαδικασία εγγραφής διακόπτεται.

Το σύστημα επιτρέπει την εισαγωγή αποκλειστικά μοναδικών κλειδιών, δηλαδή κλειδιών που δεν περιέχουν υποκλειδιά (subkeys).

3. Αν το PGP κλειδί έχει λήξει, διαγράφεται από το PGP keyring και η διαδικασία διαγραφής διακόπτεται.
4. Το σύστημα ελέγχει αν υπάρχει εγγραφή χρήστη με username το e-mail του κλειδιού. (Το e-mail χρησιμοποιείται ως username του χρήστη).

Αν δεν υπάρχει εγγραφή χρήστη, δημιουργείται νέα καταχώρηση χρήστη με απενεργοποιημένο κωδικό και συμπληρώνονται τα στοιχεία για το αντίστοιχο κλειδί στην βάση, σύμφωνα με τα στοιχεία του καταχωρημένου κλειδιού στο PGP keyring.

Αν υπάρχει εγγεγραμμένος χρήστης, διαγράφεται το υποβαλλόμενο PGP κλειδί από το PGP keyring και η διαδικασία εγγραφής διακόπτεται.

5. Ο χρήστης ενημερώνεται με μήνυμα για το αποτέλεσμα της διαδικασίας.

- **Είσοδος (Login)**

Για την είσοδο των χρηστών στο σύστημα χρησιμοποιούμε τις μεθόδους αυθεντικοποίησης χρηστών του django με την διαφορά πως ο κωδικός πρόσβασης αλλάζει σε κάθε προσπάθεια εισόδου. Ο μεταβλητός κωδικός εισόδου συνδέεται κρυπτογραφικά με το δημόσιο κλειδί του χρήστη δίνοντας σε αυτόν την δυνατότητα να συνδεθεί στο σύστημα χωρίς να γνωρίζει εκ των προτέρων τον κωδικό, αλλά να τον αποκρυπτογραφήσει (με το ιδιωτικό του κλειδί) και να τον χρησιμοποιήσει ακριβώς την στιγμή που χρειάζεται, ενώ μετά την χρήση του ο κωδικός καθίσταται άκυρος για επόμενη προσπάθεια σύνδεσης. Αναλυτικά, η διαδικασία έχει ως εξής:

Βήμα 1:

- Υποβολή e-mail ως username (Προστατευόμενη με captcha).
- Το σύστημα ελέγχει αν υπάρχει εγγραφή χρήστη με username ίδιο με αυτό που υποβάλλεται, αν η εγγραφή είναι ενεργοποιημένη, αν το αντίστοιχο PGP κλειδί έχει λήξει και αν διαθέτει το απαιτούμενο επίπεδο εμπιστοσύνης (trust level).
- Αν υπάρχει εγγραφή χρήστη, αν αυτή είναι ενεργοποιημένη και αν το PGP κλειδί δεν έχει λήξει και διαθέτει το προκαθορισμένο επίπεδο εμπιστοσύνης, δημιουργείται αυτόματα κωδικός χρήστη, καταχωρείται στην εγγραφή χρήστη κρυπτογραφημένος με συμμετρική κωδικοποίηση και ενεργοποιείται. Ο ίδιος κωδικός κρυπτογραφείται επίσης με το αντίστοιχο PGP κλειδί και εμφανίζεται κρυπτογραφημένος στο επόμενο βήμα.
- Η είσοδος δεν επιτρέπεται αν δεν υπάρχει εγγραφή χρήστη ή αν δεν είναι ενεργοποιημένη ή αν το επίπεδο εμπιστοσύνης είναι μικρότερο από το προκαθορισμένο. Ο χρήστης ενημερώνεται κατάλληλα με μήνυμα.
- Αν υπάρχει ενεργοποιημένη εγγραφή χρήστη με το απαιτούμενο επίπεδο εμπιστοσύνης, αλλά το αντίστοιχο PGP κλειδί έχει λήξει, ο χρήστης μεταφέρεται σε διαδικασία ανανέωσης του PGP κλειδιού.

Βήμα 2:

- Εμφανίζεται ένα δεύτερο πεδίο για εισαγωγή του παραγόμενου κωδικού. Ο Χρήστης μπορεί να αποκρυπτογραφήσει τον κωδικό που εμφανίζεται χρησιμοποιώντας το private key που αντιστοιχεί στο PGP κλειδί με το οποίο έχει εγγραφεί στο σύστημα και το οποίο αντιστοιχεί στο όνομα χρήστη με το οποίο προσπαθεί να συνδεθεί.

- Τέλος, ο χρήστης υποβάλλει τον αντίστοιχο παραγόμενο κωδικό αποκρυπτογραφημένο και αν συμπίπτει με τον καταχωρημένο κωδικό, ο χρήστης αποκτάει πρόσβαση στο μενού επιλογών για αυθεντικοποιημένους χρήστες και ο κωδικός χρήστη απενεργοποιείται. Διαφορετικά, ο χρήστης ενημερώνεται με μήνυμα πως ο κωδικός που εισήγαγε δεν είναι έγκυρος.

- **Ανανέωση PGP κλειδιού (Renew PGP Key)**

Η λειτουργία αυτή παρέχεται κατά την είσοδο, αν το αντίστοιχο κλειδί έχει λήξει. Αναλυτικά, η διαδικασία έχει ως εξής:

1. Ελέγχεται αν ο χρήστης πληρεί τις εξής προϋποθέσεις για την ανανέωση του PGP κλειδιού:
 - (a) Έγκυρο username (έγκυρη διεύθυνση e-mail)
 - (b) Ενεργοποιημένη εγγραφή χρήστη στη βάση
 - (c) Αντίστοιχο PGP κλειδί που έχει λήξει
2. Υποβολή νέου PGP κλειδιού. Η υποβολή προστατεύεται με κωδικό ο οποίος, αν είναι λάθος, ανανεώνεται και στέλνεται με e-mail στον αντίστοιχο χρήστη. Η διαδικασία προστατεύεται επιπλέον με ένα captcha για να αποφευχθεί η μαζική αποστολή e-mail (spam) από κάποιο κακόβουλο script που εισάγει συνεχώς λάθος κωδικούς.
3. Εισαγωγή του PGP κλειδιού στο PGP keyring του συστήματος. Αν υπάρξει κάποιο πρόβλημα κατά την εισαγωγή, η διαδικασία ανανέωσης διακόπτεται. Το σύστημα επιτρέπει την εισαγωγή αποκλειστικά μοναδικών κλειδιών, δηλαδή κλειδιών που δεν περιέχουν υποκλειδιά.
4. Αν το εισαγόμενο PGP κλειδί έχει λήξει και δεν συμπίπτει με κάποιο υπάρχον κλειδί στο σύστημα, διαγράφεται από το PGP keyring και η διαδικασία ανανέωσης διακόπτεται.
5. Έλεγχος του PGP key fingerprint στην καταχώρηση της βάσης με το fingerprint του νεοεισαχθέντος κλειδιού στο PGP keyring. Αν διαφέρουν, ελέγχεται αν το e-mail στο εισαγόμενο κλειδί χρησιμοποιείται από κάποιον καταχωρημένο χρήστη. Αν δεν χρησιμοποιείται, διαγράφεται το παλιό PGP κλειδί απ' το PGP keyring και ενημερώνεται η βάση του συστήματος σύμφωνα με τα στοιχεία του νέου PGP κλειδιού. Διαφορετικά, η διαδικασία ανανέωσης διακόπτεται.
6. Ο χρήστης ενημερώνεται για το αποτέλεσμα της διαδικασίας ανανέωσης του PGP κλειδιού.

3.2.2 Λειτουργίες για αυθεντικοποιημένους χρήστες

Όλες οι λειτουργίες για αυθεντικοποιημένους χρήστες εκτελούν στην αρχή έλεγχο για το αν ο χρήστης που τις καλεί είναι αυθεντικοποιημένος από το σύστημα και αναλόγως επιτρέπουν ή όχι την συνέχιση της εκτέλεσής τους.

- **Δημιουργία ψηφοφορίας (Create Poll)**

Εμφάνιση φόρμας με τα εξής πεδία:

- Πεδίο ερώτησης - αντικειμένου ψηφοφορίας.
- Πεδίο προσθήκης επιλογής και λίστα προστιθέμενων επιλογών. (παρέχονται κουμπιά για την προσθήκη και αφαίρεση επιλογών).
- Πεδία ελάχιστου και μέγιστου επιτρεπόμενου αριθμού επιλογών ανά ψήφο.
- Πεδία καθορισμού των χρονικών στιγμών έναρξης και λήξης της ψηφοφορίας.
- Λίστα με όλους τους ενεργοποιημένους χρήστες με εμπιστευόμενο (από την αρχή λειτουργίας του συστήματος) PGP κλειδί, για την επιλογή των ψηφοφόρων (με κουμπιά για επιλογή όλων ή κανενός).

Το σύστημα, για την ορθότητα των εισαγόμενων δεδομένων κατά την υποβολή, διενεργεί τους εξής ελέγχους:

1. Η λίστα επιλογών πρέπει να περιλαμβάνει τουλάχιστον δύο, διαφορετικές μεταξύ τους επιλογές.
2. Ο μέγιστος και ο ελάχιστος επιτρεπόμενος αριθμός επιλογών πρέπει να είναι ακέραιοι μεγαλύτεροι ή ίσοι με 1.
3. Ο μέγιστος επιτρεπόμενος αριθμός επιλογών πρέπει να είναι ίσος ή μεγαλύτερος του ελάχιστου επιτρεπόμενου αριθμού και μικρότερος ή ίσος του αριθμού επιλογών της ψηφοφορίας.
4. Στην λίστα ψηφοφόρων πρέπει να έχουν επιλεγθεί τουλάχιστον δύο ψηφοφόροι.
5. Η χρονική στιγμή έναρξης της ψηφοφορίας πρέπει να είναι μεγαλύτερη από τον τρέχοντα χρόνο του συστήματος τουλάχιστον κατά ένα χρονικό διάστημα που ορίζεται (σε λεπτά) στις ρυθμίσεις του συστήματος και μικρότερη από την χρονική στιγμή λήξης κατά ένα χρονικό διάστημα που ορίζεται επίσης (σε λεπτά) στις ρυθμίσεις του συστήματος. Έτσι, κάθε ψηφοφορία διαρκεί τουλάχιστον ένα εύλογο χρονικό διάστημα, ενώ ένα επίσης εύλογο χρονικό διάστημα μπορεί να αξιοποιηθεί για τυχόν αλλαγές ή ακύρωση (διαγραφή) της ψηφοφορίας πριν την έναρξη αυτής.

Στο σημείο αυτό κρίνουμε σκόπιμο να αναφέρουμε, ως μία επιπλέον σημείωση για τις απαιτήσεις ασφάλειας του συστήματος GPGVote, πως οι έλεγχοι που γίνονται σχετικά με τις χρονικές στιγμές έναρξης και λήξης της κάθε ψηφοφορίας απαιτούν να διατηρείται πάντοτε συγχρονισμένο το ρολόι του λειτουργικού συστήματος στο περιβάλλον του οποίου εγκαθίσταται το GPGVote. Για τον σκοπό αυτό μπορεί να χρησιμοποιηθεί το πρωτόκολλο *NTP*, το οποίο επιτρέπει τον συγχρονισμό με διεθνή ρολόγια υψηλής ακρίβειας μέσω Διαδικτύου.

- **Πίνακας ψηφοφοριών (My Polls)**

Σε αυτήν την λειτουργία εμφανίζονται όλες οι ψηφοφορίες στις οποίες δικαιούται να συμμετέχει ή έχει συμμετάσχει στο παρελθόν ο χρήστης, είτε ως ψηφοφόρος είτε ως δημιουργός τους. Οι ψηφοφορίες διαχωρίζονται σε αυτές που βρίσκονται σε εξέλιξη (για τους δημιουργούς περιλαμβάνονται επιπλέον και οι ψηφοφορίες που δεν έχουν ξεκινήσει ακόμα) και σε αυτές που έχουν λήξει.

Για κάθε ψηφοφορία, εμφανίζεται πίνακας με τα εξής πεδία:

- Η ερώτηση ή το αντικείμενο της ψηφοφορίας.
- Ο δημιουργός της ψηφοφορίας με σύνδεσμο εμφάνισης σε ξεχωριστό πίνακα των εξής αναλυτικών στοιχείων για αυτόν:
 - * Όνομα
 - * Διεύθυνση E-mail
 - * Αποτύπωμα (Fingerprint) του δημόσιου PGP κλειδιού του
 - * Ένδειξη για το αν η αρχή λειτουργίας του συστήματος τον εμπιστεύεται
- Σύνδεσμος εμφάνισης σε ξεχωριστό πίνακα της λίστας των ψηφοφόρων της ψηφοφορίας.
- Η χρονική στιγμή έναρξης της ψηφοφορίας.
- Η χρονική στιγμή λήξης της ψηφοφορίας.
- Λίστα των λειτουργιών επί της ψηφοφορίας που ενεργοποιούνται ανάλογα με την εξέλιξη της και τα δικαιώματα του χρήστη.

Οι λειτουργίες επί της ψηφοφορίας έχουν ως εξής:

- **Επεξεργασία της ψηφοφορίας (Edit Poll)**

Η λειτουργία αυτή ενεργοποιείται εφόσον ο χρήστης είναι δημιουργός της ψηφοφορίας και η ψηφοφορία δεν έχει ξεκινήσει.

Εκτελούνται τα ίδια βήματα που εκτελούνται και στην λειτουργία της δημιουργίας ψηφοφορίας με την διαφορά ότι τα πεδία της φόρμας συμπληρώνονται με τις αντίστοιχες τιμές της καταχωρημένης στη βάση δεδομένων ψηφοφορίας, ενώ, κατά την υποβολή, οι καταχωρημένες επιλογές της ψηφοφορίας διαγράφονται από την βάση και προστίθενται εκ νέου οι νέες (επεξεργασμένες) επιλογές.

Επίσης, ο έλεγχος για το αν η ψηφοφορία έχει ήδη ξεκινήσει επαναλαμβάνεται κατά την υποβολή, προς αποφυγή της αλλαγής στοιχείων σε ψηφοφορία που βρίσκεται ήδη σε εξέλιξη.

- **Διαγραφή της ψηφοφορίας (Delete Poll)**

Η λειτουργία αυτή ενεργοποιείται εφόσον ο χρήστης είναι δημιουργός της ψηφοφορίας και η ψηφοφορία δεν έχει ξεκινήσει.

Ο χρήστης καλείται με μήνυμα διαλόγου να επιβεβαιώσει την επιθυμία του για διαγραφή της ψηφοφορίας και σε ενδεχόμενη θετική απάντηση, εφόσον πληρούνται οι παραπάνω προϋποθέσεις, η ψηφοφορία διαγράφεται.

- **Κατάθεση ψήφου (Vote)**

Η λειτουργία αυτή ενεργοποιείται εφόσον ο χρήστης έχει δικαίωμα ψήφου στην ψηφοφορία, δεν έχει ήδη ψηφίσει και η ψηφοφορία βρίσκεται σε εξέλιξη [*κάλυψη της απαίτησης ασφάλειας για δημοκρατικότητα της ψηφοφορίας*].

Αν πληρούνται οι παραπάνω προϋποθέσεις, εμφανίζεται η ερώτηση ή το αντικείμενο της ψηφοφορίας και μια φόρμα με τις αντίστοιχες επιλογές. Η σχεδίαση της φόρμας επιτρέπει την μονή (με radio buttons) ή πολλαπλή επιλογή (με checkboxes), ανάλογα του επιτρεπόμενου αριθμού επιλογών της ψηφοφορίας.

Κατά την υποβολή της ψήφου διενεργούνται οι εξής έλεγχοι:

- Έλεγχος για το αν οι υποβληθείσες επιλογές αποτελούν έγκυρες επιλογές που ανήκουν στην ψηφοφορία.
- Έλεγχος για το αν ο αριθμός των υποβληθέντων επιλογών ξεπερνάει τον μέγιστο επιτρεπτό αριθμό επιλογών της ψηφοφορίας.
- Έλεγχος για το αν ο αριθμός των υποβληθέντων επιλογών είναι μικρότερος από τον ελάχιστο επιτρεπόμενο αριθμό επιλογών της ψηφοφορίας.
- Έλεγχος για τυχόν ενσωμάτωση της ίδιας επιλογής πολλαπλές φορές στην ψήφο.

Αν οι έλεγχοι αποτύχουν εμφανίζεται το κατάλληλο μήνυμα λάθους και η λειτουργία διακόπτεται. Διαφορετικά [*κάλυψη της απαίτησης για εγκυρότητα της ψήφου (ως μέρος της απαίτησης ασφάλειας για την ακρίβεια της ψηφοφορίας)*], εκτελούνται τα εξής βήματα:

- Δημιουργείται μια μοναδική, τυχαία συμβολοσειρά ως *ετικέτα ψήφου (vote tag)*.
- Δημιουργείται μήνυμα ως απόδειξη της κατάθεσης ψήφου του ψηφοφόρου, το οποίο περιλαμβάνει την ετικέτα ψήφου και τις επιλογές του ψηφοφόρου. Το μήνυμα αυτό κρυπτογραφείται με το δημόσιο PGP κλειδί του ψηφοφόρου και υπογράφεται με το κλειδί της αρχής λειτουργίας του συστήματος.
- Κάθε επιλογή της ψήφου του ψηφοφόρου συνδυάζεται με την ετικέτα ψήφου και καταχωρείται στην βάση δεδομένων του συστήματος.
- Ο ψηφοφόρος προστίθεται στην λίστα των ψηφοφόρων που έχουν ήδη ψηφίσει για την ψηφοφορία.

Στη συνέχεια, εμφανίζεται στον ψηφοφόρο η απόδειξη της κατάθεσης της ψήφου του, στην κρυπτογραφημένη και υπογεγραμμένη απ' την αρχή λειτουργίας του συστήματος μορφή (υπό την μορφή PGP μηνύματος). Ο χρήστης παροτρύνεται να φυλάξει (κρυπτογραφημένο) το εν λόγω μήνυμα, ως αποδεικτικό κατάθεσης της ψήφου του.

• Προβολή αποτελεσμάτων (Results)

Η λειτουργία αυτή ενεργοποιείται εάν η ψηφοφορία έχει ήδη λήξει [*κάλυψη της απαίτησης ασφάλειας για αμεροληψία της ψηφοφορίας*] και εφόσον ο χρήστης έχει δικαίωμα ψήφου στην ψηφοφορία ή είναι ο δημιουργός της ψηφοφορίας.

Αν πληρούνται οι παραπάνω προϋποθέσεις, εκτελείται καταμέτρηση των ψήφων και παρουσιάζεται πίνακας με τις επιλογές της ψηφοφορίας, τον αριθμό των ψήφων που συγκέντρωσε κάθε επιλογή, καθώς και το ποσοστό % επί του συνόλου των ψήφων. Αν δεν έχει κατατεθεί καμία ψήφος (ολική αποχή), εμφανίζεται σχετικό μήνυμα.

Τέλος, ο χρήστης ενημερώνεται για την δυνατότητα επιβεβαίωσης του αποτελέσματος και της καταμέτρησης της δικής του ψήφου και παρέχεται σύνδεσμος προς την επόμενη λειτουργία:

- **Λίστα ψήφων (Votes List)**

Η λειτουργία αυτή ενεργοποιείται εάν η ψηφοφορία έχει ήδη λήξει και εφόσον ο χρήστης έχει δικαίωμα ψήφου στην ψηφοφορία ή είναι ο δημιουργός της ψηφοφορίας.

Αν πληρούνται οι παραπάνω προϋποθέσεις, εμφανίζεται πίνακας με τις ετικέτες ψήφου για κάθε ψήφο και τις αντίστοιχες επιλογές ανά ετικέτα ψήφου. Ο χρήστης δύναται να ελέγξει αν προσμετράται σωστά στο τελικό αποτέλεσμα η δική του ψήφος καταμετρώντας όλες τις ψήφους και ελέγχοντας στον πίνακα την γραμμή της ετικέτας ψήφου που του είχε αποδοθεί κατά την κατάθεση της ψήφου του. Επίσης, σε ενδεχόμενη καταγγελία μη σωστής προσμέτρησης της ψήφου κάποιου χρήστη στο τελικό αποτέλεσμα της ψηφοφορίας, η αρχή λειτουργίας του συστήματος δύναται να ελέγξει την αυθεντικότητα του PGP μηνύματος σύμφωνα με το οποίο γίνεται η καταγγελία, το οποίο πρέπει να φέρει την υπογραφή της αρχής *[κάλυψη της απαίτησης ασφάλειας για επαληθευσσιμότητα του αποτελέσματος της ψηφοφορίας]*.

Στο σημείο αυτό πρέπει να σημειωθεί πως η λίστα των ψήφων μιας ψηφοφορίας είναι ανώνυμη και δεν μπορεί ο χρήστης να μάθει από αυτήν τι ψήφισε κάποιος άλλος χρήστης *[εξασφάλιση της μυστικότητας της ψηφοφορίας]*. Αυτό ισχύει επειδή η ταύτιση ενός ψηφοφόρου με την ετικέτα ψήφου της ψήφου του μπορεί να γίνει μονάχα με την αποκρυπτογράφηση του PGP μηνύματος που δημιουργείται κατά την κατάθεση της ψήφου. Ωστόσο, η αποκρυπτογράφηση αυτή απαιτεί το ιδιωτικό PGP κλειδί του ψηφοφόρου. Μάλιστα, η απαίτηση για μυστικότητα της ψηφοφορίας (όπως και όλων των λειτουργιών του συστήματος) καλύπτεται και κατά την στιγμή της κατάθεσης της ψήφου μέσω της ισχυρής κρυπτογράφησης των σχετικών μηνυμάτων που ανταλλάσσονται μεταξύ του χρήστη και του συστήματος (διασφάλιση του στρώματος μεταφοράς δεδομένων σύμφωνα με τα κρυπτογραφικά πρωτόκολλα TLS/SSL).

Επίσης, ο εξαναγκασμός για αποκάλυψη της ψήφου κάποιου ψηφοφόρου από κάποιον τρίτο δεν είναι εύκολος, αφού απαιτεί είτε φυσική συνάντηση μεταξύ των δύο πλευρών ή αποκάλυψη του ιδιωτικού κλειδιού του ψηφοφόρου *[μερική κάλυψη της απαίτησης ασφάλειας για αποτροπή της εκμείωσης ψήφων]*.

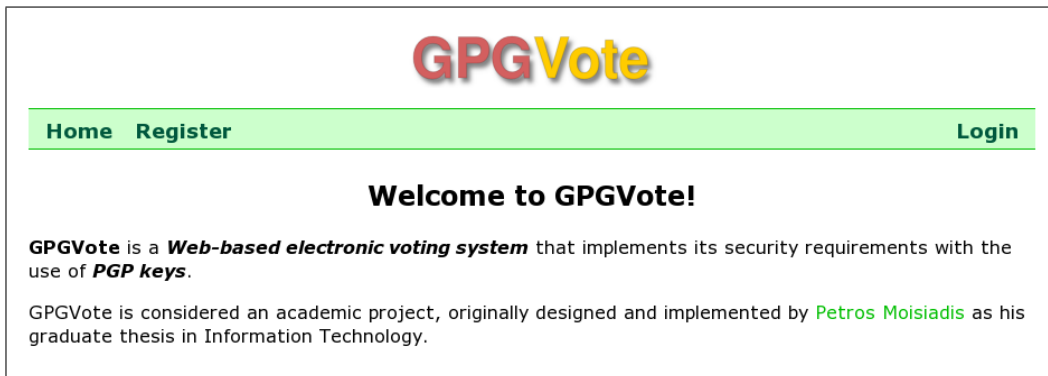
Τέλος, παρέχεται η λειτουργία:

- **Αποσύνδεση (Logout)**

Η λειτουργία αυτή ενεργοποιείται εφόσον ο χρήστης έχει εισαχθεί επιτυχώς στο σύστημα και του παρέχει την δυνατότητα εξόδου από αυτό.

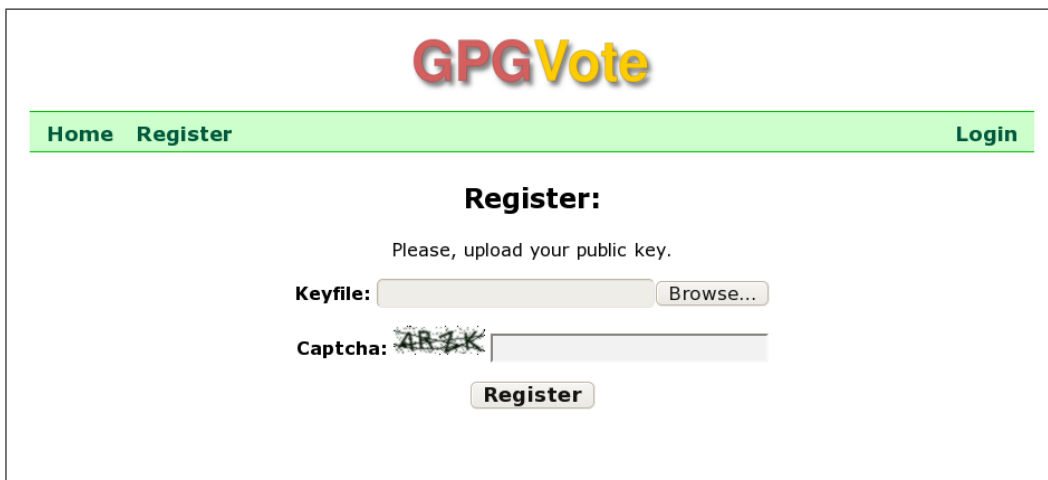
3.3 Στιγμιότυπα του συστήματος σε λειτουργία

Παραθέτουμε μερικά στιγμιότυπα (screenshots) του συστήματος σε λειτουργία:




The screenshot shows the GPGVote Home page. At the top is the GPGVote logo. Below it is a navigation bar with links for Home, Register, and Login. The main content area has a heading 'Welcome to GPGVote!' followed by a paragraph explaining that GPGVote is a Web-based electronic voting system using PGP keys. It also mentions that the system is an academic project designed by Petros Moisiadis.

Κεντρική σελίδα (για επισκέπτες)



The screenshot shows the GPGVote Register page. It features the GPGVote logo and a navigation bar with Home, Register, and Login links. The main heading is 'Register:'. Below this, it asks the user to upload their public key. There is a text input field for the 'Keyfile:' and a 'Browse...' button. Below that is a 'Captcha:' field with a distorted image of the letters '4B4K' and an empty input field. At the bottom is a 'Register' button.

Εγγραφή χρήστη




[Home](#) [Register](#)
[Login](#)

Login:

Username:

Είσοδος χρήστη (Βήμα 1)



[Home](#) [Register](#)
[Login](#)

Login:

Username:

Decrypt the following PGP-encrypted password with your Private Key and use it in the password field below:

```

-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.14 (FreeBSD)

hQEMAzwoQBzmpcuWAQgAj kDxdjCYun5iH/SWzTdCa9wSPyfxLcG8kn0++QjoU2wd
7D5etTDJdY5NV5Q7g5MGT9RGjdHGaasxR4mNCNqKL+us7FEz3xCRfZd28rK/aTz
hB7+wkSNL6T/IKv+jJccN9petEh4497HLsdyVcBcGi jXC6kRPvH1gbs9N4BkSHjd
9I+uARLc8XvqQnLxMo35U/DSqo516n5RqeeRACyNRKBj7M6Esh6zhd9nBF0xj4I/
e2uB1MIoziKr3k/1+FqNnP2bxPIL/oy1Uu7GE TTo20Y7zuJaNM+/V1DvDj07I3ir
zLDWjLJKxaRFuPJ4snwLowhVweJwzSHC6R6cPSbR9JFAMPQjCelWdhHi2eSwv9r
NuFVVsPj tqY8SWehHTC175PapR7/+OX1CdS+dyer0+zY17Y7s0b0HbBpaZ9QT/
5zeDzPH9
=Vfrb
-----END PGP MESSAGE-----

```

Password:

Είσοδος χρήστη (Βήμα 2)

GPGVote

[Home](#)

logged in as: test_user2@test.org

[Logout](#)

Welcome to GPGVote!

GPGVote is a **Web-based electronic voting system** that implements its security requirements with the use of **PGP keys**.

GPGVote is considered an academic project, originally designed and implemented by [Petros Moisiadis](#) as his graduate thesis in Information Technology.

[My Polls](#)[Create Poll](#)

Κεντρική σελίδα (για αυθεντικοποιημένους χρήστες)

Create your Poll:

Question:

Choice:

Add Choice

Added Choices:

Remove Choice(s)

Min. number of choices per vote:

Max. number of choices per vote:

Starts:

Date:

Time: (in "hh:mm" format)

Ends:

Date:

Time: (in "hh:mm" format)

Select Voters:

test user1 <test_user1@test.org>
test user2 <test_user2@test.org>
test user3 <test_user3@test.org>

None

All

Create Poll

Φόρμα δημιουργίας ψηφοφορίας

GPGVote

[Home](#)

logged in as: test_user3@test.org

[Logout](#)

Pending Polls:

Poll Question	What is your favorite color?
Created by:	test user3 <test_user3@test.org>
Allowed Voters:	Show list of allowed voters
Started:	July 15, 2010, 7:50 p.m.
Ends:	Sept. 2, 2010, noon
Actions:	 Vote

Ended Polls:

Poll Question	What is your favorite drink?
Created by:	test user2 <test_user2@test.org>
Allowed Voters:	Show list of allowed voters
Started:	July 15, 2010, 9:18 p.m.
Ended:	Aug. 15, 2010, 10 p.m.
Actions:	 Results

Πίνακας ψηφοφοριών

GPGVote

[Home](#)

logged in as: test_user3@test.org

[Logout](#)

Vote:

What is your favorite color?

Green ☒

Red ☐

[Vote](#)

Φόρμα υποβολής ψήφου

GPGVote

[Home](#)

logged in as: test_user3@test.org

[Logout](#)

You have successfully voted for the poll:

"What is your favorite color?",
Created by **test user3** <test_user3@test.org>

Your vote corresponds to the following, encrypted vote receipt:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.14 (FreeBSD)

hQEMAxXCFLPPze+cAQgAgu06ib57rTCCg7aex2V9TbnF4x0Zk8d+jLvl8FccWkrY
us3w7Q8US7t8wqIu0Se0mccZ9+uK33wTJ0gKMEnc58bE5xeYZDjCwfZvj0j5a4U
gcyMapMy3gLOU+UpK86t/rD9rxBmXbqMK+nfbZB+F6k7VNPpPD7w0GX1THtZYVKK
I0LNK+Fjvs90enCrv2MBEwmjTsqijN3iToWf1I8vMpKLLPolIWqVUFMT4a/Ei2y6
jmjVcom0mZxAwapfW6nwrU53McZ80eU1VurAbDJDbx7GCToaohAgj7TDexkKYol
xb1jAqhuC5PC02Lr6ITYq8KLyGLFVvfgCR7HfE+LfDlpAfZxbxCJBG+bJH5bzrsC
auMhyksFgm0Iz2EDdvhqef7P5ffY+8BmKI1dqXZYT19M190mTRaYpsV0afqp3BV9
poVwDL0qiFA7ENy8eErowU1f0jih4TiRT9g8f30n7P7GczqFUQYRyJw6F18s54FU
1rr2VUWpbMJJPPrcv7xrFQtyH0tqEWJCAxXMMNhsVZmwxuMARBqvFZPL6Mw6dtXo
10E9I0q7ZpPCWxFNvgU08gsGmRze1cFoSEixgLLdeHLSfUn0PqVevRSgH9TLAVfo
SJHF4LxSS23CyaBcIHL38YmXbEdbj4cLkesdVpKvKq/ve6GEFW2eX+2kfKENCno
4JunFC99n1w8kd5GT7qUT0Kzr/X3VfXZG1byscYci0UT0NSFi26+uU5wS6m01BsU
3PfXrriHoGwtTABGJ9+PAMWkjIcgDBfz0XRqI4ChmcIrWvzj1An8imTxU4nj3USbJ
bqWri3t0BZAhFAY0g3EWpV1na7e8UC5qxzfzWuLZJX8vMHZaZh2y+k7+Bpyqo/oNk
FmgRTiW4kTf5/vLuXqpQ/BoRvVJ5Xxyr9nHD1aW1C+mH3hmCeok7gcHthYK6aMW
LrDbZ273s/+ouG055PaPPnawdCJ/wkKfsdzL5NJX1ixAqtEjyET/aFSzLiEYQUh6
Fz8AI65b/1d7wo30XRI1icFDccPqe5b/5G6xEqL98tEmVt9nTpy82apDGJHhba9j
yvGomqAICouJXU19fvjVkrBnmRNBXMKiuc3Pq+ZFB6wVManZw==
--ZeIB
-----END PGP MESSAGE-----
```

You should keep a copy of the above PGP message!

You may decrypt this vote receipt with your private key and use it to verify that your vote has been correctly calculated in the final tally when this poll ends.
You should not store the decrypted form of the vote receipt anywhere.


The above PGP message is signed by the GPGVote System Authority.
You can download the public key of the GPGVote System Authority from [here](#).

Αποδεικτικό κατάθεσης ψήφου

Pending Polls:

Poll Question	What is your favorite color?
Created by:	test user3 <test_user3@test.org>
Allowed Voters:	Show list of allowed voters
Started:	July 15, 2010, 7:50 p.m.
Ends:	Sept. 2, 2010, noon
Actions:	<i>You have already voted. Please, wait for the results.</i>

Ended Polls:

Poll Question	What is your favorite drink?
Created by:	test user2 <test_user2@test.org>
Allowed Voters:	Show list of allowed voters
Started:	July 15, 2010, 9:18 p.m.
Ended:	Aug. 15, 2010, 10 p.m.
Actions:	 Results

Πίνακας ψηφοφοριών (μετά την κατάθεση ψήφου)

GPGVote

[Home](#)logged in as: test_user3@test.org [Logout](#)

Results:

Poll: "Who is the world's best soccer player?"Created by **test user1** <test_user1@test.org>

Choice	Votes	%
Lionel Messi	2	66.66
Andres Iniesta	1	33.33
Christian Ronaldo	0	0
Arjen Robben	0	0

You may verify the results by viewing the anonymous [Votes List](#).

Each individual voter may check that her vote has been calculated in the final results by verifying in the list the corresponding vote tag supplied to her when she casted her vote.

Αποτελέσματα ψηφοφορίας

GPGVote

[Home](#)logged in as: test_user3@test.org [Logout](#)

Votes List:

Poll: "Who is the world's best soccer player?"Created by **test user1** <test_user1@test.org>

Vote Tag	Choices
BS518PM8G8L1SPABBNXH905R9SIFACJPLE4	• Lionel Messi
LJE11J16GLRINEP0SUKO2AS82269K5MTCRZ	• Andres Iniesta
S1IT64BRJRHQMEF6B7GQYVJGCIEMKCY5AGA	• Lionel Messi

Λίστα ψήφων ψηφοφορίας

Κεφάλαιο 4

Συμπεράσματα

Η ανάπτυξη του συστήματος GPGVote αποτέλεσε για μας μια ευχάριστη και εποικοδομητική εργασία. Η πλατφόρμα ανάπτυξης εφαρμογών Παγκόσμιου Ιστού Django που χρησιμοποιήσαμε αποδείχθηκε ιδιαίτερα ευέλικτη, εύχρηστη και δυναμική, ικανή να καλύψει τεχνικά τις απαιτήσεις ενός συστήματος με πληθώρα λειτουργιών και ιδιαίτερες απαιτήσεις ασφάλειας.

Σύμφωνα με όσα γνωρίζουμε, δεν είχε υλοποιηθεί προηγουμένως κάποιο διαδικτυακό σύστημα διεξαγωγής ψηφοφοριών ως αυτοτελής εφαρμογή Παγκόσμιου Ιστού, βασιζόμενη στο λογισμικό κρυπτογράφησης GnuPG. Κατά συνέπεια, η καταλληλότητα της επιλογής των συγκεκριμένων τεχνολογιών για την κάλυψη των προδιαγραφών ασφάλειας που θέτουμε αποτελεί αντικείμενο προς διερεύνηση. Επίσης, το σύστημα GPGVote, έχοντας μόλις ολοκληρωθεί, δεν έχει προς το παρόν δοκιμαστεί σε πραγματικές συνθήκες· δηλαδή, δεν έχει χρησιμοποιηθεί ακόμη για τις ανάγκες κάποιας κοινότητας. Επομένως, ένα πιθανό αντικείμενο κάποιας μεταγενέστερης σχετικής μελέτης θα ήταν η υιοθέτηση του συστήματος GPGVote από μία ή περισσότερες κοινότητες και η αξιολόγησή του σε σχέση με τις προδιαγραφές που έχουμε ορίσει στην παρούσα εργασία.

Ευελπιστούμε πως θα υπάρξουν κοινότητες που θα ενδιαφερθούν για το έργο της εργασίας μας, θα επιχειρήσουν να το υιοθετήσουν δοκιμαστικά και ενδεχομένως να συμβάλλουν στην περαιτέρω ανάπτυξή του. Ωστόσο, ακόμα και αν αμφισβητηθεί η ικανότητα ανταπόκρισης του συστήματος GPGVote στις ανάγκες μιας κοινότητας, αντιλαμβανόμαστε ως επιτυχία του έργου την ανάπτυξη της εφαρμογής Django *grgauth* ως μία ανεξάρτητη μονάδα λογισμικού η οποία μπορεί να χρησιμοποιηθεί σε οποιοδήποτε έργο Django, όπως επίσης και την μικρή βελτίωση της μονάδας λογισμικού *python-gnupg*, η οποία χρησιμοποιείται ήδη σε αρκετά άλλα έργα λογισμικού.

Τέλος, αναζητώντας κάποιον «ιερό» σκοπό στις προσπάθειες για ανάπτυξη συστημάτων όπως το GPGVote, ας κλείσουμε το κείμενο αυτής της εργασίας με την ευχή το Διαδίκτυο να αποτελεί πάντοτε όχημα της Δημοκρατίας.

Παράρτημα Α

Κώδικας

27

Α.1 Στατιστικά κώδικα

Στατιστικά για τον κώδικα σύμφωνα με το πρόγραμμα cloc (<http://cloc.sourceforge.net>):

Language	files	blank	comment	code	scale	3rd gen. equiv
Python	11	159	90	805 x	4.20 =	3381.00
HTML	12	41	0	480 x	1.90 =	912.00
CSS	1	33	0	189 x	1.00 =	189.00
Javascript	1	7	0	61 x	1.48 =	90.28
SUM:	25	240	90	1535 x	2.98 =	4572.28

A.2 Κώδικας γενικών λειτουργιών

settings.py

28

```
1  # Django settings for gpgvote project.
2
3  DEBUG = False
4  TEMPLATE_DEBUG = False
5
6  ADMINS = (
7      # ('Your Name', 'your_email@domain.com'),
8  )
9
10 MANAGERS = ADMINS
11
12 DATABASES = {
13     'default': {
14         'ENGINE': 'django.db.backends.postgresql_psycopg2', # Add 'postgresql_psycopg2', 'postgresql', 'mysql', 'sqlite3'
15         # or 'oracle'.
16         'NAME': 'gpgvote', # Or path to database file if using sqlite3.
17         'USER': 'gpgvote', # Not used with sqlite3.
18         'PASSWORD': 'somepass', # Not used with sqlite3.
19         'HOST': '', # Set to empty string for localhost. Not used with sqlite3.
20         'PORT': '', # Set to empty string for default. Not used with sqlite3.
21     }
22 }
23
24 # Local time zone for this installation. Choices can be found here:
25 # http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
26 # although not all choices may be available on all operating systems.
27 # On Unix systems, a value of None will cause Django to use the same
28 # timezone as the operating system.
29 # If running in a Windows environment this must be set to the same as your
30 # system time zone.
31 TIME_ZONE = 'Europe/Athens'
32
33 # Language code for this installation. All choices can be found here:
34 # http://www.i18nguy.com/unicode/language-identifiers.html
35 LANGUAGE_CODE = 'en-us'
```

```

35
36 SITE_ID = 1
37
38 # If you set this to False, Django will make some optimizations so as not
39 # to load the internationalization machinery.
40 USE_I18N = True
41
42 # If you set this to False, Django will not format dates, numbers and
43 # calendars according to the current locale
44 USE_L10N = True
45
46 # Make this unique, and don't share it with anybody.
47 SECRET_KEY = 'd6hgy5nthm2v^53748jkg47csd@tet54hzgk*jghu9zwbm^gkj=iycy67dq'
48
49 # List of callables that know how to import templates from various sources.
50 TEMPLATE_LOADERS = (
51     'django.template.loaders.filesystem.Loader',
52     'django.template.loaders.app_directories.Loader',
53     # 'django.template.loaders.eggs.Loader',
54 )
55
56 MIDDLEWARE_CLASSES = (
57     'django.middleware.common.CommonMiddleware',
58     'django.contrib.sessions.middleware.SessionMiddleware',
59     'django.middleware.csrf.CsrfViewMiddleware',
60     'django.contrib.auth.middleware.AuthenticationMiddleware',
61     'django.contrib.messages.middleware.MessageMiddleware',
62     'django.middleware.transaction.TransactionMiddleware',
63 )
64
65 ROOT_URLCONF = 'gpgvote.urls'
66
67 TEMPLATE_DIRS = (
68     "/path/to/gpgvote/templates"
69 )
70
71 INSTALLED_APPS = (
72     'django.contrib.auth',
73     'django.contrib.contenttypes',
74     'django.contrib.sessions',

```

```

75     'django.contrib.messages',
76     'gpgvote.gpgauth',
77     'gpgvote.polls',
78     'captcha'
79 )
80
81 #SESSION_COOKIE_AGE = 300
82 GNUPGBINARY = '/path/to/gpg'
83 GNUPGHOME = '/path/to/gpgvote/gnupg_keyring'
84 SYSTEM_KEY_FINGERPRINT = 'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' # key of the system authority, used to sign PGP
    messages
85 SYSTEM_KEY_PASSWD = 'keypass' # Since this is secret, you MUST make this file readable ONLY by trusted users
86     # and, of course, make sure it is NOT accessible from the web clients
87 TRUST_LEVELS = ('f', 'u') # Trust only f-ully trusted and u-ltimately trusted keys
88 POLL_START_TIME_THRESHOLD = 5 # in minutes
89 POLL_MIN_DURATION = 60 # in minutes

```

urls.py

```
1 from django.conf.urls.defaults import *
2
3 urlpatterns = patterns('',
4     url(r'^captcha/', include('captcha.urls')),
5     (r'^$', 'gpgvote.views.main'),
6     (r'^userinfo/(?P<user_id>\d+)/$', 'gpgvote.views.userinfo'),
7     (r'^register/$', 'gpgvote.gpgauth.views.register'),
8     (r'^renew/(?P<username>.*)$', 'gpgvote.gpgauth.views.renew'),
9     (r'^login/$', 'gpgvote.gpgauth.views.login_view'),
10    (r'^logout/$', 'gpgvote.gpgauth.views.logout_view'),
11    (r'^createpoll/$', 'gpgvote.polls.views.createpoll'),
12    (r'^editpoll/(?P<poll_id>\d+)/$', 'gpgvote.polls.views.editpoll'),
13    (r'^deletepoll/(?P<poll_id>\d+)/$', 'gpgvote.polls.views.deletepoll'),
14    (r'^mypolls/$', 'gpgvote.polls.views.mypolls'),
15    (r'^voters_list/(?P<poll_id>\d+)/$', 'gpgvote.polls.views.voters_list'),
16    (r'^vote/(?P<poll_id>\d+)/$', 'gpgvote.polls.views.vote'),
17    (r'^results/(?P<poll_id>\d+)/$', 'gpgvote.polls.views.results'),
18    (r'^results/(?P<poll_id>\d+)/votes_list/$', 'gpgvote.polls.views.votes_list')
19 )
```

views.py

```
1 from django.shortcuts import render_to_response
2 from django.http import HttpResponseRedirect, Http404
3 from django.template import RequestContext
4 from django.contrib.auth.models import User
5
6 def main(request):
7     logged_in = False
8     user = ''
9     if request.user.is_authenticated():
10         user = request.user.username
11         logged_in = True
12     return render_to_response('main.html', { 'logged_in': logged_in, 'user': user })
13
```

```

14 def userinfo(request, user_id):
15     if not request.user.is_authenticated():
16         return HttpResponseRedirect('/')
17     else:
18         logged_in = True
19
20     try:
21         user = User.objects.get(pk = user_id)
22     except User.DoesNotExist:
23         raise Http404
24
25     return render_to_response('userinfo.html',
26                               {
27                                   'user': request.user.username,
28                                   'requested_user': user,
29                                   'logged_in': logged_in }, context_instance = RequestContext(request))

```

Django template: base.html

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <link rel="stylesheet" href="/site_media/css/style.css" />
6     <script src="/site_media/js/gpgvote.js" language="javascript" type="text/javascript"></script>
7     <title>{% block title %}GPGVote{% endblock %}</title>
8   </head>
9
10  <body>
11
12    <div id="header">
13      
14    </div>
15
16    <div id="menubar">
17      <div id="site_actions">
18        <ul id="menulist">
19          <li><a href="/">Home</a></li>
20          {% if not logged_in %}

```

```

21         <li><a href="/register/">Register</a></li>
22     {% endif %}
23 </ul>
24 </div>
25
26 <div id="login_actions">
27     <ul id="menulist">
28         {% if not logged_in %}
29             <li><a href="/login/">Login</a></li>
30         {% endif %}
31         {% if logged_in %}
32             <li><b>logged in as: </b>{{ user }}</li>
33             <li><a href="/logout/">Logout</a></li>
34         {% endif %}
35     </ul>
36 </div>
37 </div>
38
39 <div id="content">
40     {% block content %}{% endblock %}
41 </div>
42
43 </body>
44 </html>

```

Django template: main.html

```

1  {% extends "base.html" %}
2
3  {% block title %}Welcome{% endblock %}
4
5  {% block content %}
6      <h2>Welcome to GPGVote!</h2>
7      <p>
8          <b>GPGVote</b> is a <b><i>Web-based electronic voting system</i></b> that implements its security requirements
          with the use
9          of <b><i>PGP keys</i></b>.
10     </p>

```

```

11     <p>
12     GPGVote is considered an academic project, originally designed and implemented by
13     <a href="mailto:ernest0x@dtips.unipi.gr">Petros Moisiadis</a> as his graduate thesis in Information Technology.
14     <p>
15     {% if logged_in %}
16     <div id="main_functions">
17         <table cellspacing="40" cellpadding="25" align="center">
18             <tr>
19                 <td><a href="./mypolls/"></a><br /><b>My Polls</b></td>
20                 <td><a href="./createpoll/"></a><br /><b>Create Poll</b></td>
21             </tr>
22         </table>
23     </div>
24     {% endif %}
25 {% endblock %}

```

CSS: style.css

```

1 body
2 {
3     width: 900px;
4     margin-left: auto;
5     margin-right: auto;
6     margin-top: 0px;
7     margin-bottom: 0px;
8     font-family: Verdana, sans-serif;
9     background-color: white;
10 }
11
12 a:link { color: #00BF00 }
13 a:visited { color: #005200 }
14 a { text-decoration: none }
15
16 img
17 {
18     border: 0px;
19 }
20

```

```
21 h2
22 {
23     text-align: center;
24 }
25
26 .textmiddle { vertical-align:middle; }
27
28 #header
29 {
30     text-align: center;
31     height: 70px;
32     padding-top: 15px;
33 }
34
35 #site_actions
36 {
37     float: left;
38     padding-left: 15px;
39 }
40
41 #menubar
42 {
43     background-color: #CCFFCC;
44     height: 35px;
45     font-size: small;
46     border-top: 1px solid #00BF00;
47     border-bottom: 1px solid #00BF00;
48 }
49
50 #menubar a
51 {
52     color: #00583D;
53     font-size: large;
54     font-weight: bold;
55     text-decoration: none;
56 }
57
58 #menubar a:hover
59 {
60     color: #00BF00;
```

```
61  }
62
63  #login_actions
64  {
65      float: right;
66  }
67
68  #content
69  {
70      clear: both;
71      padding-top: 5px;
72  }
73
74  #menulist
75  {
76      padding-left: 0px;
77      margin-top: 0px;
78      margin-bottom: 0px;
79      padding-top: 7px;
80  }
81
82  #menulist li
83  {
84      padding-right: 15px;
85      display: inline;
86      list-style-type: none;
87  }
88
89  #main_functions
90  {
91      text-align: center;
92      text-decoration: none;
93      color: #00583D;
94  }
95
96  #main_functions img:hover
97  {
98      background-color: #CCFFCC;
99      border: 1px dotted #00BF00;
100 }
```

```
101
102 #poll_edit th
103 {
104     background-color: #FBFFFB;
105     border-top: 1px solid #00BF00;
106     border-bottom: 1px solid #00BF00;
107 }
108
109 #poll_edit textarea, select, input[type=text]
110 {
111     background-color: #F2F2F2;
112 }
113
114 #poll_edit textarea:focus, select:focus, input[type=text]:focus
115 {
116     background-color: #FFF0B3;
117 }
118
119 #id_choices, #id_choice, #id_question
120 {
121     min-width: 450px;
122 }
123
124 #id_choices
125 {
126     min-height: 200px;
127     max-height: 300px;
128 }
129
130 #id_allowed_voters
131 {
132     width: 350px;
133     min-height: 80px;
134     max-height: 200px;
135 }
136
137 #error_td
138 {
139     max-width: 350px;
140 }
```

```
141
142 #poll_table
143 {
144     width: 800px;
145 }
146
147 #poll_table td
148 {
149     text-align: center;
150     padding: 10px;
151     background-color: #FEF4F4;
152     border-top: 1px solid #FEC0C0;
153     border-bottom: 1px solid #FEC0C0;
154 }
155
156 #poll_table th
157 {
158     background-color: #E8FFE8;
159     border-top: 1px solid #00BF00;
160     border-bottom: 1px solid #00BF00;
161     width: 150px;
162 }
163
164 #userinfo_table td
165 {
166     text-align: center;
167     padding: 10px;
168     background-color: #FEF4F4;
169     border-top: 1px solid #FEC0C0;
170     border-bottom: 1px solid #FEC0C0;
171 }
172
173 #userinfo_table th
174 {
175     background-color: #E8FFE8;
176     border-top: 1px solid #00BF00;
177     border-bottom: 1px solid #00BF00;
178     padding: 10px;
179 }
180
```



```
181 #results
182 {
183     text-align: right;
184     max-width: 150px;
185 }
186
187 #results_first_col
188 {
189     text-align: left;
190     max-width: 450px;
191 }
192
193 .results_th
194 {
195     background-color: #E8FFE8;
196     border-top: 1px solid #00BF00;
197     border-bottom: 1px solid #00BF00;
198     padding: 10px;
199 }
200
201 .results_td_even
202 {
203     background-color: #FEF4F4;
204     padding: 10px;
205 }
206
207 .results_td_odd
208 {
209     background-color: #FEE9E9;
210     padding: 10px;
211 }
212
213 #vote_receipt
214 {
215     text-align: left;
216     margin-left: auto;
217     margin-right: auto;
218     width: 35em;
219     background-color: #E8FFE8;
220     border: 1px dotted #00BF00;
```

```
221 padding: 10px;
222 }
```

A.3 Εφαρμογή gpgauth

models.py

40

```
1 from django.db import models
2 from django.conf import settings
3 from django.contrib.auth.models import User
4 from gnupg import GPG
5
6 class PGPkey(models.Model):
7     user = models.OneToOneField(User)
8     name = models.CharField(max_length=100)
9     fingerprint = models.CharField(unique=True, max_length=50)
10    renew_passwd = models.CharField(max_length=35)
11    is_trusted = models.BooleanField(default=False)
12
13    def __init__(self, *args, **kwargs):
14        super(PGPkey, self).__init__(*args, **kwargs)
15        self.clean()
16
17    def clean(self):
18        gpg = GPG(gpgbinary=settings.GNUPGBINARY, gnupghome=settings.GNUPGHOME)
19        key = gpg.get_key(self.fingerprint)
20        if key['ownertrust'] in settings.TRUST_LEVELS:
21            self.is_trusted = True
22        else:
23            self.is_trusted = False
24        self.save()
25
26    def __unicode__(self):
27        return self.fingerprint
```

forms.py

```
1 from django import forms
2 from captcha.fields import CaptchaField
3
4 class RegisterForm(forms.Form):
5     keyfile = forms.FileField()
6     captcha = CaptchaField()
7
8 class RenewForm(forms.Form):
9     keyfile = forms.FileField()
10    password = forms.CharField(widget=forms.PasswordInput)
11    captcha = CaptchaField()
12
13 class LoginForm(forms.Form):
14     username = forms.EmailField()
15     password = forms.CharField(widget=forms.PasswordInput)
```

41

views.py

```
1 from django.shortcuts import render_to_response, redirect
2 from django.http import HttpResponseRedirect
3 from django.template import RequestContext
4 from django.conf import settings
5 from django.core.exceptions import ObjectDoesNotExist
6 from django.core.validators import validate_email
7 from django.contrib.auth.models import User
8 from django.contrib.auth import authenticate, login, logout
9 from gpgvote.gpgauth.forms import RegisterForm, RenewForm, LoginForm
10 from gpgvote.gpgauth.models import PGKey
11 from gnupg import GPG
12 from hashlib import md5
13
14
15 # Utility Functions
16 def delete_keys(gpg, fingerprints, del_existed=False):
17     # Make sure that fingerprint is a list
```

```

18 if type(fingerprints).__name__ != 'list':
19     fingerprints = [fingerprints]
20
21 for fp in fingerprints:
22     # Delete key only if it does not exist in database or del_existed is True
23     try:
24         key = PGPkey.objects.get(fingerprint = fp)
25         if del_existed:
26             gpg.delete_keys(fp)
27     except ObjectDoesNotExist:
28         gpg.delete_keys(fp)
29
30
31 def key_import(gpg, keyfile):
32     if keyfile.size > 100000: # accept files of a normal size
33         error = 'Key file size is too big'
34     else:
35         error = ''
36         try:
37             import_result = gpg.import_keys(keyfile.read())
38         except UnicodeDecodeError:
39             error = 'There was an error in importing your key'
40             return error, None
41
42     if import_result.count == 0:
43         error = 'There was an error in importing your key'
44
45     if import_result.count > 1: # accept only single-key files
46         error = 'Your key file includes more than one keys'
47         delete_keys(gpg, import_result.fingerprints)
48
49     if import_result.count == 1:
50         fp = import_result.fingerprints[0]
51         if gpg.key_is_expired(fp):
52             error = 'Your key is expired'
53             delete_keys(gpg, import_result.fingerprints)
54         else:
55             return error, gpg.get_key(fp)
56
57     return error, None

```

```

58
59 def login_common_checks(username):
60     try:
61         user = User.objects.get(username = username)
62         if user.is_active:
63             gpg = GPG(gpgbinary=settings.GNUPGBINARY, gnupghome=settings.GNUPGHOME)
64             if not gpg.key_is_expired(user.pgpkey.fingerprint):
65                 key = gpg.get_key(user.pgpkey.fingerprint)
66                 if key['ownertrust'] in settings.TRUST_LEVELS:
67                     error = False
68                 else:
69                     error = 'PGP key for user \'%s\' is not trusted (yet)' % username
70             else:
71                 error = 'PGP key for user \'%s\' has expired' % username
72         else:
73             error = 'Account for user \'%s\' is disabled' % username
74         gpg = None
75     except ObjectDoesNotExist:
76         error = 'User \'%s\' does not exist' % username
77         user = None
78         gpg = None
79
80     return user, error, gpg
81
82
83 # Views
84
85 def register(request):
86     if request.user.is_authenticated():
87         return HttpResponseRedirect('/')
88     error = ''
89     success = ''
90     if request.POST:
91         form = RegisterForm(request.POST, request.FILES)
92         if form.is_valid():
93             keyfile = request.FILES['keyfile']
94             gpg = GPG(gpgbinary=settings.GNUPGBINARY, gnupghome=settings.GNUPGHOME)
95             (error, imported_key) = key_import(gpg, keyfile)
96             if not error:
97                 # check for user existence in database to accept registration only for new users

```

```

98         try:
99             user = User.objects.get(email = imported_key['email'])
100             error = 'User \'%s\' is already registered' % imported_key['email']
101             if user.pgpkey.fingerprint != imported_key['fingerprint']:
102                 delete_keys(gpg, imported_key['fingerprint'])
103         except ObjectDoesNotExist:
104             newuser = User.objects.create_user(username = imported_key['email'],
105                                                 email = imported_key['email'],
106                                                 password = '')
107             newuser.set_unusable_password()
108             newuser.save()
109             pgpkey = PGPkey(user = newuser, name = imported_key['name'], fingerprint = imported_key['fingerprint'])
110             pgpkey.save()
111             success = 'You are now registered'
112
113     else:
114         form = RegisterForm()
115
116     return render_to_response('register.html',
117                               {
118                                   'form': form,
119                                   'error': error,
120                                   'success': success }, context_instance = RequestContext(request))
121
122 def renew(request, username):
123     if request.user.is_authenticated():
124         return HttpResponseRedirect('/')
125     error = ''
126     success = ''
127     try:
128         validate_email(username)
129     except:
130         error = 'Invalid username'
131     if not error:
132         (user, error, gpg) = login_common_checks(username)
133         if not error:
134             error = 'Your key is not expired yet'
135         if error.endswith('expired'):
136             error = ''
137

```

```

138 if request.POST:
139     form = RenewForm(request.POST, request.FILES)
140     if form.is_valid():
141         if user.pgpkey.renew_passwd != md5(form.cleaned_data['password']).hexdigest():
142             passwd = User.objects.make_random_password()
143             user.pgpkey.renew_passwd = md5(passwd).hexdigest()
144             user.pgpkey.save()
145             msg = 'Use the following password to renew your key:\n' + passwd
146             user.email_user('Renew Key', msg)
147             error = 'Wrong Password. The correct password has been sent to: %s' % username
148         else:
149             keyfile = request.FILES['keyfile']
150             (error, imported_key) = key_import(gpg, keyfile)
151         if not error:
152             if imported_key['fingerprint'] == user.pgpkey.fingerprint:
153                 error = 'The uploaded key already exists'
154             else:
155                 update_user = False
156                 # Check if the email of the uploaded key is already used by a different user
157                 try:
158                     user = User.objects.get(username = imported_key['email'])
159                     if username != imported_key['email']:
160                         error = 'There is another user with username: \'%s\' ' % imported_key['email']
161                         delete_keys(gpg, imported_key['fingerprint'])
162                     else:
163                         update_user = True
164                 except:
165                     update_user = True
166
167             if update_user:
168                 delete_keys(gpg, user.pgpkey.fingerprint, del_existed=True)
169                 user.pgpkey.fingerprint = imported_key['fingerprint']
170                 user.pgpkey.name = imported_key['name']
171                 user.pgpkey.save()
172                 user.username = imported_key['email']
173                 user.email = imported_key['email']
174                 user.save()
175                 success = 'Your key was successfully renewed'
176
177     else:

```

```

178     form = RenewForm()
179
180     return render_to_response('renew.html',
181                               {
182                                   'form': form,
183                                   'error': error,
184                                   'success': success,
185                                   'username': username }, context_instance = RequestContext(request))
186
187 def login_view(request):
188     if request.user.is_authenticated():
189         return HttpResponseRedirect('/')
190     error = ''
191     password = ''
192     try:
193         goto_stage = request.POST['stage']
194     except:
195         goto_stage = 'password'
196     if request.POST:
197         form = LoginForm(request.POST)
198         if goto_stage == 'password':
199             try:
200                 validate_email(request.POST['username'])
201             except:
202                 error = 'Invalid username'
203         if not error:
204             (user, error, gpg) = login_common_checks(request.POST['username'])
205             if not error:
206                 password = User.objects.make_random_password()
207                 user.set_password(password)
208                 user.save()
209                 password = gpg.encrypt(password, user.pgpkey.fingerprint, always_trust=True) # Trust level is checked earlier
210                 if password.ok:
211                     goto_stage = 'afterpass'
212             else:
213                 user.set_unusable_password()
214                 user.save()
215                 error = 'Encryption error (%s)' % password.status
216         elif error.endswith('expired'):
217             return redirect('/renew/%s' % request.POST['username'])

```


47

```
218         else:
219             pass
220
221     elif goto_stage == 'afterpass':
222         if form.is_valid():
223             # Run common checks again to disappoint those who try to bypass first login step
224             (user, error, gpg) = login_common_checks(form.cleaned_data['username'])
225             if not error:
226                 user = authenticate(username=form.cleaned_data['username'], password = form.cleaned_data['password'] )
227                 if user is not None:
228                     login(request, user)
229                     return HttpResponseRedirect('/')
230             else:
231                 error = 'Wrong password'
232                 form = ''
233             else:
234                 goto_stage = 'password'
235         else:
236             error = 'Invalid username or password'
237             goto_stage = 'password'
238
239     else:
240         form = LoginForm()
241
242     return render_to_response('login.html', {
243         'form': form,
244         'goto_stage': goto_stage,
245         'error': error,
246         'password': password    }, context_instance = RequestContext(request))
247
248 def logout_view(request):
249     logout(request)
250     return HttpResponseRedirect('/')
```

Django template: register.html

```

1 {% extends "base.html" %}
2
3 {% block title %}Register{% endblock %}
4
5 {% block content %}
6     <div style="text-align:center">
7         <h2>Register:</h2>
8         <p>Please, upload your public key.</p>
9         {% if not success %}
10        {% if error %}<p style="color:red"><b>Error:</b> {{ error }}</p>{% endif %}
11        <form enctype="multipart/form-data" action="/register/" method="post">
12            {% csrf_token %}
13            <b>{{ form.as_p }}</b>
14            <input style="font-weight:bold;font-size:large" type="submit" value="Register" />
15        </form>
16        {% endif %}
17        {% if success %}
18        <p><b>Success:</b> {{ success }}</p>
19        {% endif %}
20    </div>
21 {% endblock %}

```

Django template: login.html

```

1 {% extends "base.html" %}
2
3 {% block title %}Login{% endblock %}
4
5 {% block content %}
6     <h2>Login:</h2>
7     <div style="text-align:center">
8         {% if error %}
9         <p style="color:red"><b>Error: </b>{{ error }}</p>
10        {% endif %}
11        {% if form %}

```

```

12     <form action="/login/" method="post">
13         {% csrf_token %}
14         <p><b>Username:</b> {{ form.username }}</p>
15         {% if goto_stage = "afterpass" %}
16         <p>Decrypt the following PGP-encrypted password with your Private Key and use it in the password field below:</p>
17         <div id="vote_receipt"><pre>{{ password }}</pre></div>
18         <p><b>Password:</b> {{ form.password }}</p>
19         {% endif %}
20         <input type="hidden" name="stage" value="{{ goto_stage }}" />
21         <input style="font-weight:bold;font-size:large" type="submit" value="Login" />
22     </form>
23     {% endif %}
24 </div>
25 {% endblock %}

```

Django template: renew.html

49

```

1  {% extends "base.html" %}
2
3  {% block title %}Renew Key{% endblock %}
4
5  {% block content %}
6      <h2>Renew your key:</h2>
7      {% if not success %}
8          {% if error %}<p style="text-align:center;color:red"><b>Error:</b> {{ error }}</p>
9          {% else %}
10             <p style="text-align:center">Your key has expired. You can renew your key here.
11             <br/>The required password has been sent to your email address.</p>
12             <p style="text-align:center">Please, upload your public key.<br /><br />
13             <form style="text-align:center" enctype="multipart/form-data" action="/renew/{{ username }}" method="post">
14                 {% csrf_token %}
15                 {{ form.as_p }}
16                 <input type="submit" value="Renew" />
17             </form>
18             </p>
19         {% endif %}
20     {% endif %}
21     {% if success %}

```

```
22         <p style="text-align:center"><b>Success:</b> {{ success }}</p>
23         {% endif %}
24     {% endblock %}
```

A.4 Εφαρμογή polls

models.py

50

```
1  from django.db import models
2  from django.conf import settings
3  from django.contrib.auth.models import User
4
5  class Poll(models.Model):
6      creator = models.ForeignKey(User)
7      question = models.TextField()
8      min_choices = models.PositiveIntegerField()
9      max_choices = models.PositiveIntegerField()
10     allowed_voters = models.TextField('voters allowed to vote')
11     who_voted = models.TextField('voters who have already voted')
12     starts = models.DateTimeField()
13     ends = models.DateTimeField()
14
15     def __unicode__(self):
16         return self.question
17
18     def has_voted(self, voter):
19         if voter in self.who_voted.split(';'):
20             return True
21         else:
22             return False
23
24     def add_voter(self, voter, To):
25         if To == 'allowed_voters':
26             self.allowed_voters = self.allowed_voters + voter + ';'
27         elif To == 'who_voted':
```

```

28         if voter in self.allowed_voters.split(';'):
29             self.who_voted = self.who_voted + voter + ';'
30
31     def remove_voter(self, voter, From):
32         if From == 'allowed_voters':
33             self.allowed_voters = self.allowed_voters.replace(voter+';', ' ')
34         elif From == 'who_voted':
35             self.who_voted = self.who_voted.replace(voter+';', ' ')
36
37     def is_allowed_voter(self, voter):
38         if voter in self.allowed_voters.split(';'):
39             return True
40         else:
41             return False
42
43     class Choice(models.Model):
44         poll = models.ForeignKey(Poll)
45         choice = models.CharField(max_length=255)
46
47     def __unicode__(self):
48         return self.choice
49
50     class Vote(models.Model):
51         choice = models.ForeignKey(Choice)
52         tag = models.CharField(max_length=35)
53
54     def __unicode__(self):
55         return '\\' + self.tag + '\\' vote tag includes choice: ' + '\\' + str(self.choice) + '\\'

```

forms.py

```

1 from django import forms
2 from django.forms import TextInput, Textarea
3 from django.forms.extras.widgets import SelectDateWidget
4
5 class PollForm(forms.Form):
6     question = forms.CharField(widget=Textarea(attrs={'rows': '3'}))
7     min_choices = forms.IntegerField(widget=TextInput(attrs={'size': '2'}))

```

```

8     max_choices = forms.IntegerField(widget=TextInput(attrs={'size':'2'}))
9     starts_date = forms.DateField(widget=SelectDateWidget())
10    starts_time = forms.TimeField(widget=TextInput(attrs={'size':'2'}))
11    ends_date = forms.DateField(widget=SelectDateWidget())
12    ends_time = forms.TimeField(widget=TextInput(attrs={'size':'2'}))
13    allowed_voters = forms.MultipleChoiceField([])
14
15    def __init__(self, allowed_voters, *args, **kwargs):
16        super(PollForm, self).__init__(*args, **kwargs)
17        self.fields['allowed_voters'].choices = allowed_voters

```

views.py

```

1  from django.shortcuts import render_to_response
2  from django.db.models import Q
3  from django.http import HttpResponseRedirect, Http404
4  from django.template import RequestContext
5  from django.conf import settings
6  from django.forms import Form
7  from gpgvote.polls.forms import PollForm
8  from django.contrib.auth.models import User
9  from gpgvote.gpgauth.models import PGPkey
10 from gpgvote.polls.models import Poll
11 from gpgvote.polls.models import Choice
12 from gpgvote.polls.models import Vote
13 from gnupg import GPG
14 import datetime
15 import random
16 import string
17 import operator
18
19
20 # Utility functions
21 def list_has_duplicates(mylist):
22     if len(mylist) > len(list(set(mylist))):
23         return True
24     else:
25         return False

```

```

26
27 def dates_check(starts_date, starts_time, ends_date, ends_time):
28     error = ''
29     starts_datetime = datetime.datetime(
30         year    = starts_date.year,
31         month   = starts_date.month,
32         day     = starts_date.day,
33         hour    = starts_time.hour,
34         minute  = starts_time.minute,
35         second  = starts_time.second )
36
37     ends_datetime = datetime.datetime(
38         year    = ends_date.year,
39         month   = ends_date.month,
40         day     = ends_date.day,
41         hour    = ends_time.hour,
42         minute  = ends_time.minute,
43         second  = ends_time.second )
44
45     if starts_datetime < datetime.datetime.now() + datetime.timedelta(minutes = settings.POLL_START_TIME_THRESHOLD):
46         error = '<ul><li>Poll must start at least ' + str(settings.POLL_START_TIME_THRESHOLD) + ' minutes in the future</li>
47             ></ul>'
48
49     duration = ends_datetime - starts_datetime
50     if duration < datetime.timedelta(minutes = settings.POLL_MIN_DURATION):
51         if duration < datetime.timedelta(microseconds=0):
52             error = '<ul><li>Poll must end at least ' + str(settings.POLL_MIN_DURATION) + ' minutes after the start</li></ul>'
53         else:
54             error = '<ul><li>Poll must last for at least ' + str(settings.POLL_MIN_DURATION) + ' minutes</li></ul>'
55
56     return error, starts_datetime, ends_datetime
57
58 def num_of_choices_check(num_of_choices, min_choices, max_choices):
59     error = ''
60     if max_choices < min_choices:
61         error = '<ul><li>Max. must be bigger than or equal to Min.</li></ul>'
62
63     if (max_choices < 1) or (min_choices < 1):
64         error = '<ul><li>Max. and Min. must be bigger than or equal to 1</li></ul>'

```

```

65     if (not error) and (max_choices > num_of_choices):
66         error = '<ul><li>Max. and Min. must be smaller than or equal to the number of choices</li></ul>'
67
68     return error
69
70 # Views
71 def poll(request, action, poll_id):
72     if not request.user.is_authenticated():
73         return HttpResponseRedirect('/')
74     else:
75         logged_in = True
76
77     choices_error = ''
78     dates_error = ''
79     num_of_choices_error = ''
80     allowed_voters_error = ''
81     success = ''
82     poll_data = {}
83     poll_choices = []
84
85     if action == 'edit':
86         try:
87             poll = Poll.objects.get(pk = poll_id)
88         except Poll.DoesNotExist:
89             raise Http404
90         # allow edit only for creator and only if the poll has not started yet
91         if (request.user != poll.creator) or (poll.starts < datetime.datetime.now()):
92             return HttpResponseRedirect('/')
93         poll_data = {
94             'question': poll.question,
95             'min_choices': poll.min_choices,
96             'max_choices': poll.max_choices,
97             'starts_date': poll.starts.date(),
98             'starts_time': poll.starts.time(),
99             'ends_date': poll.ends.date(),
100             'ends_time': poll.ends.time() }
101         poll_choices = Choice.objects.filter(poll = poll).order_by('id')
102
103     # Clean PGKey records to correct is_trusted field
104     for key in PGKey.objects.all():
105         key.clean()

```



```

105
106 # Get trusted users and create allowed voters choices
107 trusted_users = User.objects.filter(pgpkey__is_trusted=True).order_by('pgpkey__name')
108 allowed_voters = ()
109 for user in trusted_users:
110     allowed_voters = allowed_voters + ( (user.username, user.pgpkey.name + ' <' + user.username + '>' ), )
111
112 if request.POST:
113     # allow edit only if the poll has not started yet
114     if (action == 'edit'):
115         if (poll.starts < datetime.datetime.now()):
116             return HttpResponseRedirect('/')
117
118 form = PollForm(allowed_voters, request.POST, initial = poll_data)
119 if form.is_valid():
120     poll_choices = request.POST.getlist('choices')
121     if len(poll_choices) < 2:
122         choices_error = '<ul><li>You must add at least 2 choices</li></ul>'
123     else:
124         for choice in poll_choices:
125             if len(choice) > 255:
126                 choices_error = '<ul><li>Each choice must be up to 255 characters in length</li></ul>'
127
128     (dates_error, starts_datetime, ends_datetime) = dates_check(
129                                     form.cleaned_data['starts_date'],
130                                     form.cleaned_data['starts_time'],
131                                     form.cleaned_data['ends_date'],
132                                     form.cleaned_data['ends_time'] )
133     num_of_choices_error = num_of_choices_check(len(poll_choices),
134                                                 form.cleaned_data['min_choices'],
135                                                 form.cleaned_data['max_choices'])
136     if len(form.cleaned_data['allowed_voters']) < 2:
137         allowed_voters_error = '<ul><li>You must select at least 2 voters</li></ul>'
138
139     if not (choices_error or dates_error or num_of_choices_error or allowed_voters_error):
140         if action == 'create':
141             poll = Poll(
142                 creator = request.user,
143                 question = form.cleaned_data['question'],
144                 min_choices = form.cleaned_data['min_choices'],

```

```

145         max_choices = form.cleaned_data['max_choices'],
146         allowed_voters = '',
147         who_voted = '',
148         starts = starts_datetime,
149         ends = ends_datetime )
150     else:
151         poll.question = form.cleaned_data['question']
152         poll.min_choices = form.cleaned_data['min_choices']
153         poll.max_choices = form.cleaned_data['max_choices']
154         poll.allowed_voters = ''
155         poll.starts = starts_datetime
156         poll.ends = ends_datetime
157
158     for voter in form.cleaned_data['allowed_voters']:
159         poll.add_voter(voter, To = 'allowed_voters')
160     poll.save()
161
162     # Delete old choices before adding their new versions
163     if action == 'edit':
164         Choice.objects.filter(poll = poll).delete()
165     for choice in poll_choices:
166         choice = Choice(poll = poll, choice = choice)
167         choice.save()
168
169     if action == 'create':
170         success = 'You have successfully created a new poll'
171     else:
172         success = 'You have successfully edited the poll'
173 else:
174     form = PollForm(allowed_voters, initial = poll_data)
175
176 if action == 'edit':
177     poll_id = poll.id
178 else:
179     poll_id = ''
180
181 return render_to_response('poll.html',
182     {
183         'form': form,
184         'action': action,
185         'poll_id': poll_id,

```

```

185         'choices': poll_choices,
186         'choices_error': choices_error,
187         'dates_error': dates_error,
188         'num_of_choices_error': num_of_choices_error,
189         'allowed_voters_error': allowed_voters_error,
190         'success': success,
191         'user': request.user.username,
192         'logged_in': logged_in }, context_instance = RequestContext(request))
193
194 def createpoll(request):
195     return poll(request, 'create', None)
196
197 def editpoll(request, poll_id):
198     return poll(request, 'edit', poll_id)
199
200 def deletepoll(request, poll_id):
201     if not request.user.is_authenticated():
202         return HttpResponseRedirect('/')
203     else:
204         logged_in = True
205
206     try:
207         poll = Poll.objects.get(pk = poll_id)
208     except Poll.DoesNotExist:
209         raise Http404
210
211     # Delete poll only if it has not started yet and the user is the creator of the poll
212     if (poll.creator == request.user) and (poll.starts > datetime.datetime.now()):
213         poll.delete()
214
215     return HttpResponseRedirect('/mypolls')
216
217 def mypolls(request):
218     if not request.user.is_authenticated():
219         return HttpResponseRedirect('/')
220     else:
221         logged_in = True
222
223     # Polls that are created by the user and polls for which the user is allowed to vote
224     mypolls_query = Poll.objects.filter(Q(creator = request.user)

```



```

265             'logged_in': logged_in }, context_instance = RequestContext(request))
266
267 def vote(request, poll_id):
268     if not request.user.is_authenticated():
269         return HttpResponseRedirect('/')
270     else:
271         logged_in = True
272
273     error = ''
274     success = ''
275     try:
276         poll = Poll.objects.get(pk = poll_id)
277     except Poll.DoesNotExist:
278         raise Http404
279
280     username = request.user.username
281     if (not poll.is_allowed_voter(username)) \
282         or poll.has_voted(username) \
283         or (poll.starts > datetime.datetime.now()) \
284         or (poll.ends < datetime.datetime.now()):
285         return HttpResponseRedirect('/mypolls')
286
287     poll_choices = Choice.objects.filter(poll = poll)
288     choice_type = "radio"
289     if poll.max_choices > 1:
290         choice_type = "checkbox"
291
292     vote_tag = ''
293     vote_receipt_encrypted = ''
294
295     if request.POST:
296         form = Form(request.POST)
297         if form.is_valid():
298             choices = request.POST.getlist('choices')
299
300             # Check that the submitted choices exist and belong to the poll
301             for choice in choices:
302                 try:
303                     c = Choice.objects.get(pk = choice, poll = poll)
304                 except Choice.DoesNotExist:

```

```

305         error = "The submitted choices are not valid choices of the poll"
306
307     # Check that the submitted choices are between min and max number of choices allowed for the poll
308     if len(choices) > poll.max_choices:
309         error = 'You cannot vote for more than ' + str(poll.max_choices) + ' choices'
310     if len(choices) < poll.min_choices:
311         error = 'You must vote for at least ' + str(poll.min_choices) + ' choices'
312     if poll.max_choices == 1: # a better error message for single choice polls
313         error = 'You must select a choice'
314     if list_has_duplicates(choices):
315         error = 'Each choice can be selected only once'
316
317     if not error:
318         # Construct a unique, random string to use as a vote tag
319         while not vote_tag:
320             vote_tag = ''.join(random.choice(string.ascii_uppercase + string.digits) for x in range(35))
321         try:
322             v = Vote.objects.get(tag = vote_tag)
323             vote_tag = ''
324         except Vote.DoesNotExist: # our random string is unique so we can use it as a vote tag
325             # Encrypt the vote tag with user's public pgp key and sign it with the key of the system authority
326             gpg = GPG(gpgbinary=settings.GNUPGBINARY, gnupghome=settings.GNUPGHOME)
327             vote_receipt = ""GPGVote: Vote Receipt
328             -----
329
330         You are voter:
331             %s
332
333         You voted for Poll:
334             \'%s\'
335
336         Created by:
337             %s
338
339         Your Vote Tag is: %s
340
341         You made the following choices:"" % (request.user.pgpkey.name + ' <' + request.user.username + '>', poll.question, \
342                                             poll.creator.pgpkey.name + ' <' + poll.creator.username + '>', vote_tag)
343
344         for choice in choices:

```

```

345         choice = Choice.objects.get(pk = choice, poll = poll)
346         vote_receipt = vote_receipt + '\n * %s' % choice.choice
347
348         vote_receipt_encrypted = gpg.encrypt(vote_receipt, request.user.pgpkey.fingerprint, always_trust = True,
349                                             sign = settings.SYSTEM_KEY_FINGERPRINT,
350                                             passphrase = settings.SYSTEM_KEY_PASSWD)
351         # Create the actual vote records in database
352         for choice in choices:
353             vote = Vote(choice = Choice.objects.get(id = choice), tag = vote_tag)
354             vote.save()
355         poll.add_voter(voter = username, To = 'who_voted')
356         poll.save()
357
358         success = 'You have successfully voted for the poll'
359
360     return render_to_response('vote.html',
361                             {
362                                 'user': username,
363                                 'poll': poll,
364                                 'choices': poll_choices,
365                                 'choice_type': choice_type,
366                                 'error': error,
367                                 'success': success,
368                                 'vote_receipt': vote_receipt_encrypted,
369                                 'logged_in': logged_in }, context_instance = RequestContext(request))
370
371 def results(request, poll_id):
372     if not request.user.is_authenticated():
373         return HttpResponseRedirect('/')
374     else:
375         logged_in = True
376
377     try:
378         poll = Poll.objects.get(pk = poll_id)
379     except Poll.DoesNotExist:
380         raise Http404
381
382     username = request.user.username
383     if ((not poll.is_allowed_voter(username)) and (poll.creator != request.user)) or (poll.ends > datetime.datetime.now()):
384         return HttpResponseRedirect('/mypolls')

```

```

385 total_abstention = False
386 try:
387     votes = Vote.objects.filter(choice__poll = poll)
388 except Vote.DoesNotExist:
389     total_abstention = True
390
391 if not votes: total_abstention = True
392
393 results = {}
394 if not total_abstention:
395     choices = Choice.objects.filter(poll = poll)
396     for choice in choices:
397         results[choice.choice] = (0, 0)
398     for vote in votes:
399         votes_count = results[vote.choice.choice][0] + 1
400         votes_percent = str(float(votes_count) / float(len(votes)) * 100)[0:5]
401         results[vote.choice.choice] = (votes_count, votes_percent)
402
403     results = sorted(results.iteritems(), key = operator.itemgetter(1))
404     results.reverse()
405
406 return render_to_response('results.html',
407                           {
408                               'user': username,
409                               'poll': poll,
410                               'results': results,
411                               'total_abstention': total_abstention,
412                               'logged_in': logged_in }, context_instance = RequestContext(request))
413
414 def votes_list(request, poll_id):
415     if not request.user.is_authenticated():
416         return HttpResponseRedirect('/')
417     else:
418         logged_in = True
419
420     try:
421         poll = Poll.objects.get(pk = poll_id)
422     except Poll.DoesNotExist:
423         raise Http404
424
425     username = request.user.username

```


69

```

425     if ((not poll.is_allowed_voter(username)) and (poll.creator != request.user)) or (poll.ends > datetime.datetime.now()):
426         return HttpResponseRedirect('/mypolls')
427
428     total_abstention = False
429     try:
430         votes = Vote.objects.filter(choice__poll = poll).order_by('tag')
431     except Vote.DoesNotExist:
432         total_abstention = True
433
434     vote_tags = {}
435     if not votes:
436         total_abstention = True
437     else:
438         for vote in votes:
439             try:
440                 vote_tags[vote.tag]
441             except KeyError:
442                 vote_tags[vote.tag] = []
443                 vote_tags[vote.tag] = vote_tags[vote.tag] + [vote.choice]
444
445     vote_tags = sorted(vote_tags.iteritems(), key = operator.itemgetter(1))
446
447     return render_to_response('votes_list.html',
448                             {
449                                 'user': username,
450                                 'poll': poll,
451                                 'votes': vote_tags,
452                                 'total_abstention': total_abstention,
453                                 'logged_in': logged_in }, context_instance = RequestContext(request))
454
455 def voters_list(request, poll_id):
456     if not request.user.is_authenticated():
457         return HttpResponseRedirect('/')
458     else:
459         logged_in = True
460
461     try:
462         poll = Poll.objects.get(pk = poll_id)
463     except Poll.DoesNotExist:
464         raise Http404

```

```

465     username = request.user.username
466     if ((not poll.is_allowed_voter(username)) and (poll.creator != request.user)):
467         return HttpResponseRedirect('/mypolls')
468
469     allowed_voters = poll.allowed_voters.split(';')
470
471     qobject = Q()
472     for voter in allowed_voters:
473         if voter == '':
474             continue
475         else:
476             qobject = qobject | Q(username=voter)
477
478     voters = User.objects.filter(qobject).order_by('pgpkey__name')
479
480     return render_to_response('voters_list.html',
481                             {
482                                 'user': username,
483                                 'poll': poll,
484                                 'voters': voters,
485                                 'logged_in': logged_in }, context_instance = RequestContext(request))

```

Django template: poll.html

```

1  {% extends "base.html" %}
2
3  {% block title %}Create Poll{% endblock %}
4
5  {% block content %}
6      <h2>{{ action|capfirst }} your Poll:</h2>
7      {% if not success %}
8      {% if action == "create" %}
9      <form action="/createpoll/" method="post">
10     {% else %}
11     <form action="/editpoll/{{ poll_id }}/" method="post">
12     {% endif %}
13     {% csrf_token %}
14     <br />
15     <table cellpadding="10" align="center">

```

```

16 <tr>
17 <td valign="top" style="border-right: 1px dashed green">
18
19 <table id="poll_edit">
20 <tr><th>Question:</th></tr>
21 <tr><td>{{ form.question }}</td></tr>
22 <tr><td id="error_td">{{ form.question.errors }}</td></tr>
23 <tr><td colspan="2">&nbsp;</td></tr>
24 <tr><th>Choice:</th></tr>
25 <tr><td><textarea id="id_choice" rows="2" name="choice"></textarea></td></tr>
26 <tr>
27 <td align="center">
28 <input type="button" value="Add Choice" name="add_choice" onclick="addChoice('id_choice', '
    id_choices')" />
29 </td>
30 </tr>
31 <tr><td colspan="2">&nbsp;</td></tr>
32 <tr><th>Added Choices:</th></tr>
33 <tr>
34 <td>
35 <select multiple="multiple" id="id_choices" name="choices">
36 <option value="{{ choice }}">{{ choice}}</option>
37 <option value="{{ choice }}">{{ choice}}</option>
38 <option value="{{ choice }}">{{ choice}}</option>
39 </select>
40 </td>
41 </tr>
42 <tr><td id="error_td">{{ choices_error|safe }}</td></tr></tr>
43 <tr>
44 <td align="center">
45 <input type="button" value="Remove Choice(s)" name="remove_choice" onclick="removeChoices('
    id_choices')" />
46 </td>
47 </tr>
48 </table>
49
50 </td>
51 <td valign="top">
52
53 <table id="poll_edit">

```

```

54         <tr><th align="left">Min. number of choices per vote:</th></tr><tr><td>{{ form.min_choices }}</td></tr>
55     <tr><th align="left">Max. number of choices per vote:</th></tr><tr><td>{{ form.max_choices }}</td></tr>
56     <tr><td colspan="2" id="error_td">{{ form.max_choices.errors }} {{ num_of_choices_error|safe }}</td></tr>
57         <tr><td colspan="2">&nbsp;</td></tr>
58     <tr><th colspan="2">Starts:</th></tr>
59     <tr><td colspan="2">Date: {{ form.starts_date }}</td></tr>
60     <tr><td colspan="2" id="error_td">{{ form.starts_date.errors }} {{ dates_error|safe }}</td></tr>
61     <tr><td colspan="2">Time: {{ form.starts_time }} <small><i> (in "hh:mm" format)</i></small></td></tr>
62     <tr><td colspan="2" id="error_td">{{ form.starts_time.errors }}</td></tr>
63     <tr><td colspan="2">&nbsp;</td></tr>
64     <tr><th colspan="2">Ends:</th></tr>
65     <tr><td colspan="2">Date: {{ form.ends_date }}</td></tr>
66     <tr><td colspan="2" id="error_td">{{ form.ends_date.errors }} {{ dates_error|safe }}</td></tr>
67     <tr><td colspan="2">Time: {{ form.ends_time }} <small><i> (in "hh:mm" format)</i></small></td></tr>
68     <tr><td colspan="2" id="error_td">{{ form.ends_time.errors }}</td></tr>
69     <tr><td colspan="2">&nbsp;</td></tr>
70     <tr><th colspan="2">Select Voters:</th></tr>
71     <tr><td colspan="2">{{ form.allowed_voters }}</td></tr>
72     <tr><td colspan="2" id="error_td">{{ form.allowed_voters.errors }} {{ allowed_voters_error|safe }}</td></tr>
73     <tr>
74         <td colspan="2" align="center">
75             <input type="button" value="None" name="select_none" onclick="selectChoices('none', '
76                 id_allowed_voters')" />
77             <input type="button" value="All" name="select_all" onclick="selectChoices('all', 'id_allowed_voters
78                 ')" />
79         </td>
80     </tr>
81     </td>
82 </tr>
83 <tr><td colspan="2"><hr /></td></tr>
84 </table>
85 <div align="center">
86     <input style="font-size: large; font-weight: bold" type="submit" value="{{ action|capfirst }}" Poll" onclick="
87         selectChoices('all', 'id_choices')" />
88 </div>
</form>

```

```

89     {% else %}
90     <p>{{ success }}</p>
91     {% endif %}
92 {% endblock %}

```

Javascript: gpgvote.js

67

```

1 function addChoice(choice_id, choices_id)
2 {
3     var choice = document.createElement('option');
4     var choice_to_add = document.getElementById(choice_id);
5     if (/^\s*$/.test(choice_to_add.value)) return;
6     var choices = document.getElementById(choices_id);
7     for (i = 0; i < choices.length; i++)
8         if (choices.options[i].value == choice_to_add.value) return;
9     choice.text = choice_to_add.value;
10    choice.value = choice_to_add.value;
11    choices.add(choice, null);
12 }
13
14 function removeChoices(choices_id)
15 {
16     var choices = document.getElementById(choices_id);
17
18     for (i = choices.length-1; i >= 0; i--)
19     {
20         if(choices.options[i].selected)
21         {
22             choices.remove(i);
23         }
24     }
25 }
26
27
28 function selectChoices(choice_set, choices_id)
29 {
30     var choices = document.getElementById(choices_id);
31     for (i = 0; i < choices.length; i++)

```

```
32  {
33      if (choice_set == 'all')
34          choices.options[i].selected = true;
35      else if (choice_set == 'none')
36          choices.options[i].selected = false;
37      else
38          choice_set = choice_set.split(';')
39          for (choice in choice_set)
40              if (choice == choices.options[i].value)
41                  choices.options[i].selected = true;
42  }
43 }
44
45 function delconfirm(poll_id)
46 {
47     if (! poll_id) poll_id="";
48     var del = confirm("Are you sure you want to delete this poll?");
49     if (del == true)
50         location.replace("/deletetpoll/" + poll_id);
51 }
52
53 function checkChoices(choice_id, max)
54 {
55     var choices = document.getElementsByName("choices");
56     var checked_choices = 0;
57     for (i=0; i < choices.length; i++)
58     {
59         if (choices[i].checked) checked_choices++;
60     }
61
62     if (checked_choices > max)
63     {
64         alert("You cannot vote for more than " + max + " choices");
65         var choice = document.getElementById(choice_id);
66         choice.checked = false;
67     }
68 }
```

Django template: mypolls.html

69

```
1 {% extends "base.html" %}
2
3 {% block title %}My Polls{% endblock %}
4
5 {% block content %}
6     {% if pending_polls %}<h2>Pending Polls:</h2>{% endif %}
7     {% for p in pending_polls %}
8         {% if p.allowed_actions %}
9             <table id="poll_table" align="center">
10                 <tr>
11                     <th>Poll Question</th>
12                     <td><b>{{ p.poll.question }}</b></td>
13                 </tr>
14                 <tr>
15                     <th>Created by:</th>
16                     <td>
17                         <a href="/userinfo/{{ p.poll.creator.id }}">
18                             {{ p.poll.creator.pgkey.name }} &lt;{{ p.poll.creator.username }}&gt;
19                         </a>
20                     </td>
21                 </tr>
22                 <tr>
23                     <th>Allowed Voters:</th>
24                     <td>
25                         <a href="/voters_list/{{ p.poll.id }}">Show list of allowed voters</a>
26                     </td>
27                 </tr>
28                 <tr>
29                     <th>{% if p.allowed_actions == 'edit' %}Starts:{% else %}Started:{% endif %}</th>
30                     <td>{{ p.poll.starts }}</td>
31                 </tr>
32                 <tr>
33                     <th>Ends:</th>
34                     <td>{{ p.poll.ends }}</td>
35                 </tr>
36                 <tr>
37                     <th>Actions:</th>
38                     <td>
```

```

39         <ul id="menulist">
40             {% if p.allowed_actions == 'edit' %}
41                 <li>
42                     <a href="/editpoll/{ p.poll.id }"> Edit</a>
44                 </li>
45                 <li>
46                     <a href="#" onclick="delconfirm('{ p.poll.id }')">
47                          Delete
48                     </a>
49                 </li>
50             {% endif %}
51             {% if p.allowed_actions == 'vote' %}
52                 <li>
53                     <a href="/vote/{ p.poll.id }"> Vote
54                     </a>
55                 </li>
56             {% endif %}
57             {% if p.allowed_actions == 'results' %}
58                 <li>
59                     <a href="/results/{ p.poll.id }"> Results</a>
61                 </li>
62             {% endif %}
63             {% if p.allowed_actions == 'wait' %}<i>You have already voted. Please, wait for the results.</i>{%
64                 endif %}
65             {% if p.allowed_actions == 'wait_creator' %}<i>Please, wait for the results.</i>{% endif %}
66         </ul>
67     </td>
68 </tr>
69 </table>
70 <br />
71 {% endif %}
72 {% for p in ended_polls %}
73     <table id="poll_table" align="center">

```



```

75         <tr>
76             <th>Poll Question</th>
77             <td><b>{{ p.poll.question }}</b></td>
78         </tr>
79         <tr>
80             <th>Created by:</th>
81             <td>
82                 <a href="/userinfo/{{ p.poll.creator.id }}">
83                     {{ p.poll.creator.pgkey.name }} &lt;{{ p.poll.creator.username }}&gt;
84                 </a>
85             </td>
86         </tr>
87         <tr>
88             <th>Allowed Voters:</th>
89             <td>
90                 <a href="/voters_list/{{ p.poll.id }}">Show list of allowed voters</a>
91             </td>
92         </tr>
93         <tr>
94             <th>Started:</th>
95             <td>{{ p.poll.starts }}</td>
96         </tr>
97         <tr>
98             <th>Ended:</th>
99             <td>{{ p.poll.ends }}</td>
100        </tr>
101        <tr>
102            <th>Actions:</th>
103            <td>
104                <ul id="menulist">
105                    <li>
106                        <a href="/results/{{ p.poll.id }}">
107                            Results</a>
108                    </li>
109                </ul>
110            </td>
111        </tr>
112    </table>
113    <br />
    {% endfor %}

```

```
114
115 {% endblock %}
```

Django template: userinfo.html

72

```
1  {% extends "base.html" %}
2
3  {% block title %}My Polls{% endblock %}
4
5  {% block content %}
6      <h2>User Info:</h2>
7      <table id="userinfo_table" align="center">
8          <tr>
9              <th>Name:</th>
10             <td><b>{{ requested_user.pgpkey.name }}</b></td>
11         </tr>
12         <tr>
13             <th>E-mail:</th>
14             <td><a href="mailto:{{ requested_user.email }}">{{ requested_user.email }}</a></td>
15         </tr>
16         <tr>
17             <th>Public PGP key fingerprint:</th>
18             <td>{{ requested_user.pgpkey }}</td>
19         </tr>
20         <tr>
21             <th>Trusted by the Authority:</th>
22             <td>
23                 {% if requested_user.pgpkey.is_trusted %}
24                     <span style="color:green">Yes</span>
25                 {% else %}
26                     <span style="color:red">No</span>
27                 {% endif %}
28             </td>
29         </tr>
30     </table>
31
32 {% endblock %}
```

Django template: voters_list.html

```
1 {% extends "base.html" %}
2
3 {% block title %}Voters{% endblock %}
4
5 {% block content %}
6
7     <h2>Voters List:</h2>
8     <p style="text-align:center">
9         <b>Poll: &quot;{{ poll.question }}&quot;</b><br />
10        Created by <b>{{ poll.creator.pgpkey.name }}</b> &lt;{{ poll.creator }}&gt;<br /><br />
11    </p>
12    <table align="center">
13        <tr><th class="results_th">Allowed Voters</th></tr>
14        {% for voter in voters %}
15            <tr>
16                <td class="{% cycle 'results_td_even' 'results_td_odd' %}">
17                    <a href="/userinfo/{{ voter.id }}">{{ voter.pgpkey.name }} &lt;{{ voter.username }}&gt;</a>
18                </td>
19            </tr>
20        {% endfor %}
21    </table>
22
23 {% endblock %}
```

Django template: vote.html

```
1 {% extends "base.html" %}
2
3 {% block title %}Vote{% endblock %}
4
5 {% block content %}
6     {% if not success %}
7         <h2>Vote:</h2>
8         <form action="/vote/{{ poll.id }}" method="post">
9             {% csrf_token %}
```

```

10     <table align="center">
11     <tr><th colspan="2">{{ poll.question }}</th></tr>
12     <tr><td colspan="2" height="10"></td></tr>
13     {% for choice in choices %}
14     <tr>
15         <td>{{ choice.choice }}</td>
16         <td>
17             <input type="{{ choice_type }}" name="choices" id="{{ choice.id }}" value="{{ choice.id }}"
18                 onchange="checkChoices('{{ choice.id }}', {{ poll.max_choices }})"/>
19         </td>
20     </tr>
21     {% endfor %}
22 </table>
23 <br />
24 {% if error %}<p style="text-align:center;color:red"><b>Error: </b><i>{{ error }}</i></p><br />{% endif %}
25 <p style="text-align:center"><input style="font-weight:bold;font-size:large" type="submit" value="Vote" /></p>
26 </form>
27 {% else %}
28 <p style="text-align:center">
29     {{ success }}:<br /><br />
30     <b>&quot;{{ poll.question }}&quot;</b><br />
31     Created by <b>{{ poll.creator.pgpkey.name }}</b> &lt;{{ poll.creator }}&gt;<br /><br />
32     Your vote corresponds to the following, encrypted vote receipt: <br />
33     <div id="vote_receipt">
34         <b><pre>{{ vote_receipt }}</pre></b>
35     </div>
36 </p>
37 <p style="text-align:center"><i>You should keep a copy of the above PGP message!</i></p>
38 <p style="text-align:justify">
39     You may decrypt this vote receipt with your private key and use it to verify that your vote has been
40     correctly calculated in the final tally when this poll ends.<br />
41     You should not store the decrypted form of the vote receipt anywhere.
42 </p>
43 <p style="text-align:justify">
44     The above PGP message is signed by the GPGVote System Authority.<br />
45     You can download the public key of the GPGVote System Authority from
46     <a href="/site_media/gpgvote_authority_public_key.asc">here</a>.
47 </p>
48 {% endif %}
49

```

50 {% endblock %}

Django template: results.html

```
1 {% extends "base.html" %}
2
3 {% block title %}Results{% endblock %}
4
5 {% block content %}
6     <h2>Results:</h2>
7     <p style="text-align:center">
8         <b>Poll: &quot;{{ poll.question }}&quot;</b><br />
9         Created by <b>{{ poll.creator.pgkey.name }}</b> &lt;{{ poll.creator }}&gt;<br /><br />
10    </p>
11    {% if not total_abstention %}
12        <table align="center" cellpadding="10">
13            <tr>
14                <th id="results_first_col" class="results_th">Choice</th>
15                <th id="results" class="results_th">Votes</th>
16                <th id="results" class="results_th">%</th>
17            </tr>
18            {% for result in results %}
19                <tr>
20                    <td class="{% cycle 'results_td_even' 'results_td_odd' %}" id="results_first_col">{{ result.0 }}</td>
21                    <td class="{% cycle 'results_td_even' 'results_td_odd' %}" id="results">{{ result.1.0 }}</td>
22                    <td class="{% cycle 'results_td_even' 'results_td_odd' %}" id="results">
23                        
24                        <br />{{ result.1.1 }}
25                    </td>
26                </tr>
27            {% endfor %}
28        </table>
29        <p style="text-align:justify">
30            You may verify the results by viewing the anonymous <a href="/results/{{ poll.id }}/votes_list">Votes List</a>
31            <br />Each individual voter may check that her vote has been calculated in the final results by verifying in
32            the
33            list the corresponding vote tag supplied to her when she casted her vote.
```

```

33         </p>
34     {% else %}
35     <p style="text-align:center">Total Abstention</p>
36     {% endif %}
37
38 {% endblock %}

```

Django template: votes_list.html

```

1  {% extends "base.html" %}
2
3  {% block title %}Results{% endblock %}
4
5  {% block content %}
6
7      <h2>Votes List:</h2>
8      <p style="text-align:center">
9          <b>Poll: &quot;{{ poll.question }}&quot;</b><br />
10         Created by <b>{{ poll.creator.pgpkey.name }}</b> &lt;{{ poll.creator }}&gt;<br /><br />
11     </p>
12     {% if not total_abstention %}
13         <table align="center">
14             <tr><th class="results_th">Vote Tag</th><th class="results_th">Choices</th></tr>
15             {% for tag in votes %}
16                 <tr>
17                     <td class="{% cycle 'results_td_even' 'results_td_odd' %}">{{ tag.0 }}</td>
18                     <td class="{% cycle 'results_td_even' 'results_td_odd' %}">
19                         <ul>
20                             {% for vote in tag.1 %}
21                                 <li>{{ vote }}</li>
22                             {% endfor %}
23                         </ul>
24                     </td>
25                 </tr>
26             {% endfor %}
27         </table>
28     {% else %}
29         <p style="text-align:center">Total Abstention</p>

```

```
30         {% endif %}
31
32     {% endblock %}
```

A.5 python-gnupg patch

gnupg.py.patch

77

```
1  --- gnupg.py      2010-08-01 10:18:50.000000000 +0300
2  +++ gnupg-patched.py      2010-08-01 10:20:08.000000000 +0300
3  @@ -50,6 +50,8 @@
4      import logging
5      import os
6      import socket
7  +import re
8  +from time import time
9      from subprocess import Popen
10     from subprocess import PIPE
11     import threading
12  @@ -135,6 +137,7 @@
13         self.gnupghome = gnupghome
14         self.verbose = verbose
15         self.encoding = locale.getpreferredencoding()
16  +
17         self.r = re.compile('(.*) <(.*@.*)>') # pattern for key owner name/email pair
18         if gnupghome and not os.path.isdir(self.gnupghome):
19             os.makedirs(self.gnupghome,0x1C0)
20         p = self._open_subprocess(["--version"])
21  @@ -423,6 +426,32 @@
22         getattr(result, keyword)(L)
23         return result
24
25  +
26  +
27  +
28  +
29  +
30  +
31  +
32  +
33  +
34  +
35  +
36  +
37  +
38  +
39  +
40  +
41  +
42  +
43  +
44  +
45  +
46  +
47  +
48  +
49  +
50  +
51  +
52  +
53  +
54  +
55  +
56  +
57  +
58  +
59  +
60  +
61  +
62  +
63  +
64  +
65  +
66  +
67  +
68  +
69  +
70  +
71  +
72  +
73  +
74  +
75  +
76  +
77  +
78  +
79  +
80  +
81  +
82  +
83  +
84  +
85  +
86  +
87  +
88  +
89  +
90  +
91  +
92  +
93  +
94  +
95  +
96  +
97  +
98  +
99  +
100 +
101 +
102 +
103 +
104 +
105 +
106 +
107 +
108 +
109 +
110 +
111 +
112 +
113 +
114 +
115 +
116 +
117 +
118 +
119 +
120 +
121 +
122 +
123 +
124 +
125 +
126 +
127 +
128 +
129 +
130 +
131 +
132 +
133 +
134 +
135 +
136 +
137 +
138 +
139 +
140 +
141 +
142 +
143 +
144 +
145 +
146 +
147 +
148 +
149 +
150 +
151 +
152 +
153 +
154 +
155 +
156 +
157 +
158 +
159 +
160 +
161 +
162 +
163 +
164 +
165 +
166 +
167 +
168 +
169 +
170 +
171 +
172 +
173 +
174 +
175 +
176 +
177 +
178 +
179 +
180 +
181 +
182 +
183 +
184 +
185 +
186 +
187 +
188 +
189 +
190 +
191 +
192 +
193 +
194 +
195 +
196 +
197 +
198 +
199 +
200 +
201 +
202 +
203 +
204 +
205 +
206 +
207 +
208 +
209 +
210 +
211 +
212 +
213 +
214 +
215 +
216 +
217 +
218 +
219 +
220 +
221 +
222 +
223 +
224 +
225 +
226 +
227 +
228 +
229 +
230 +
231 +
232 +
233 +
234 +
235 +
236 +
237 +
238 +
239 +
240 +
241 +
242 +
243 +
244 +
245 +
246 +
247 +
248 +
249 +
250 +
251 +
252 +
253 +
254 +
255 +
256 +
257 +
258 +
259 +
260 +
261 +
262 +
263 +
264 +
265 +
266 +
267 +
268 +
269 +
270 +
271 +
272 +
273 +
274 +
275 +
276 +
277 +
278 +
279 +
280 +
281 +
282 +
283 +
284 +
285 +
286 +
287 +
288 +
289 +
290 +
291 +
292 +
293 +
294 +
295 +
296 +
297 +
298 +
299 +
300 +
301 +
302 +
303 +
304 +
305 +
306 +
307 +
308 +
309 +
310 +
311 +
312 +
313 +
314 +
315 +
316 +
317 +
318 +
319 +
320 +
321 +
322 +
323 +
324 +
325 +
326 +
327 +
328 +
329 +
330 +
331 +
332 +
333 +
334 +
335 +
336 +
337 +
338 +
339 +
340 +
341 +
342 +
343 +
344 +
345 +
346 +
347 +
348 +
349 +
350 +
351 +
352 +
353 +
354 +
355 +
356 +
357 +
358 +
359 +
360 +
361 +
362 +
363 +
364 +
365 +
366 +
367 +
368 +
369 +
370 +
371 +
372 +
373 +
374 +
375 +
376 +
377 +
378 +
379 +
380 +
381 +
382 +
383 +
384 +
385 +
386 +
387 +
388 +
389 +
390 +
391 +
392 +
393 +
394 +
395 +
396 +
397 +
398 +
399 +
400 +
401 +
402 +
403 +
404 +
405 +
406 +
407 +
408 +
409 +
410 +
411 +
412 +
413 +
414 +
415 +
416 +
417 +
418 +
419 +
420 +
421 +
422 +
423 +
424 +
425 +
426 +
427 +
428 +
429 +
430 +
431 +
432 +
433 +
434 +
435 +
436 +
437 +
438 +
439 +
440 +
441 +
442 +
443 +
444 +
445 +
446 +
447 +
448 +
449 +
450 +
451 +
452 +
453 +
454 +
455 +
456 +
457 +
458 +
459 +
460 +
461 +
462 +
463 +
464 +
465 +
466 +
467 +
468 +
469 +
470 +
471 +
472 +
473 +
474 +
475 +
476 +
477 +
478 +
479 +
480 +
481 +
482 +
483 +
484 +
485 +
486 +
487 +
488 +
489 +
490 +
491 +
492 +
493 +
494 +
495 +
496 +
497 +
498 +
499 +
500 +
501 +
502 +
503 +
504 +
505 +
506 +
507 +
508 +
509 +
510 +
511 +
512 +
513 +
514 +
515 +
516 +
517 +
518 +
519 +
520 +
521 +
522 +
523 +
524 +
525 +
526 +
527 +
528 +
529 +
530 +
531 +
532 +
533 +
534 +
535 +
536 +
537 +
538 +
539 +
540 +
541 +
542 +
543 +
544 +
545 +
546 +
547 +
548 +
549 +
550 +
551 +
552 +
553 +
554 +
555 +
556 +
557 +
558 +
559 +
560 +
561 +
562 +
563 +
564 +
565 +
566 +
567 +
568 +
569 +
570 +
571 +
572 +
573 +
574 +
575 +
576 +
577 +
578 +
579 +
580 +
581 +
582 +
583 +
584 +
585 +
586 +
587 +
588 +
589 +
590 +
591 +
592 +
593 +
594 +
595 +
596 +
597 +
598 +
599 +
600 +
601 +
602 +
603 +
604 +
605 +
606 +
607 +
608 +
609 +
610 +
611 +
612 +
613 +
614 +
615 +
616 +
617 +
618 +
619 +
620 +
621 +
622 +
623 +
624 +
625 +
626 +
627 +
628 +
629 +
630 +
631 +
632 +
633 +
634 +
635 +
636 +
637 +
638 +
639 +
640 +
641 +
642 +
643 +
644 +
645 +
646 +
647 +
648 +
649 +
650 +
651 +
652 +
653 +
654 +
655 +
656 +
657 +
658 +
659 +
660 +
661 +
662 +
663 +
664 +
665 +
666 +
667 +
668 +
669 +
670 +
671 +
672 +
673 +
674 +
675 +
676 +
677 +
678 +
679 +
680 +
681 +
682 +
683 +
684 +
685 +
686 +
687 +
688 +
689 +
690 +
691 +
692 +
693 +
694 +
695 +
696 +
697 +
698 +
699 +
700 +
701 +
702 +
703 +
704 +
705 +
706 +
707 +
708 +
709 +
710 +
711 +
712 +
713 +
714 +
715 +
716 +
717 +
718 +
719 +
720 +
721 +
722 +
723 +
724 +
725 +
726 +
727 +
728 +
729 +
730 +
731 +
732 +
733 +
734 +
735 +
736 +
737 +
738 +
739 +
740 +
741 +
742 +
743 +
744 +
745 +
746 +
747 +
748 +
749 +
750 +
751 +
752 +
753 +
754 +
755 +
756 +
757 +
758 +
759 +
760 +
761 +
762 +
763 +
764 +
765 +
766 +
767 +
768 +
769 +
770 +
771 +
772 +
773 +
774 +
775 +
776 +
777 +
778 +
779 +
780 +
781 +
782 +
783 +
784 +
785 +
786 +
787 +
788 +
789 +
790 +
791 +
792 +
793 +
794 +
795 +
796 +
797 +
798 +
799 +
800 +
801 +
802 +
803 +
804 +
805 +
806 +
807 +
808 +
809 +
810 +
811 +
812 +
813 +
814 +
815 +
816 +
817 +
818 +
819 +
820 +
821 +
822 +
823 +
824 +
825 +
826 +
827 +
828 +
829 +
830 +
831 +
832 +
833 +
834 +
835 +
836 +
837 +
838 +
839 +
840 +
841 +
842 +
843 +
844 +
845 +
846 +
847 +
848 +
849 +
850 +
851 +
852 +
853 +
854 +
855 +
856 +
857 +
858 +
859 +
860 +
861 +
862 +
863 +
864 +
865 +
866 +
867 +
868 +
869 +
870 +
871 +
872 +
873 +
874 +
875 +
876 +
877 +
878 +
879 +
880 +
881 +
882 +
883 +
884 +
885 +
886 +
887 +
888 +
889 +
890 +
891 +
892 +
893 +
894 +
895 +
896 +
897 +
898 +
899 +
900 +
901 +
902 +
903 +
904 +
905 +
906 +
907 +
908 +
909 +
910 +
911 +
912 +
913 +
914 +
915 +
916 +
917 +
918 +
919 +
920 +
921 +
922 +
923 +
924 +
925 +
926 +
927 +
928 +
929 +
930 +
931 +
932 +
933 +
934 +
935 +
936 +
937 +
938 +
939 +
940 +
941 +
942 +
943 +
944 +
945 +
946 +
947 +
948 +
949 +
950 +
951 +
952 +
953 +
954 +
955 +
956 +
957 +
958 +
959 +
960 +
961 +
962 +
963 +
964 +
965 +
966 +
967 +
968 +
969 +
970 +
971 +
972 +
973 +
974 +
975 +
976 +
977 +
978 +
979 +
980 +
981 +
982 +
983 +
984 +
985 +
986 +
987 +
988 +
989 +
990 +
991 +
992 +
993 +
994 +
995 +
996 +
997 +
998 +
999 +
1000 +
1001 +
1002 +
1003 +
1004 +
1005 +
1006 +
1007 +
1008 +
1009 +
1010 +
1011 +
1012 +
1013 +
1014 +
1015 +
1016 +
1017 +
1018 +
1019 +
1020 +
1021 +
1022 +
1023 +
1024 +
1025 +
1026 +
1027 +
1028 +
1029 +
1030 +
1031 +
1032 +
1033 +
1034 +
1035 +
1036 +
1037 +
1038 +
1039 +
1040 +
1041 +
1042 +
1043 +
1044 +
1045 +
1046 +
1047 +
1048 +
1049 +
1050 +
1051 +
1052 +
1053 +
1054 +
1055 +
1056 +
1057 +
1058 +
1059 +
1060 +
1061 +
1062 +
1063 +
1064 +
1065 +
1066 +
1067 +
1068 +
1069 +
1070 +
1071 +
1072 +
1073 +
1074 +
1075 +
1076 +
1077 +
1078 +
1079 +
1080 +
1081 +
1082 +
1083 +
1084 +
1085 +
1086 +
1087 +
1088 +
1089 +
1090 +
1091 +
1092 +
1093 +
1094 +
1095 +
1096 +
1097 +
1098 +
1099 +
1100 +
1101 +
1102 +
1103 +
1104 +
1105 +
1106 +
1107 +
1108 +
1109 +
1110 +
1111 +
1112 +
1113 +
1114 +
1115 +
1116 +
1117 +
1118 +
1119 +
1120 +
1121 +
1122 +
1123 +
1124 +
1125 +
1126 +
1127 +
1128 +
1129 +
1130 +
1131 +
1132 +
1133 +
1134 +
1135 +
1136 +
1137 +
1138 +
1139 +
1140 +
1141 +
1142 +
1143 +
1144 +
1145 +
1146 +
1147 +
1148 +
1149 +
1150 +
1151 +
1152 +
1153 +
1154 +
1155 +
1156 +
1157 +
1158 +
1159 +
1160 +
1161 +
1162 +
1163 +
1164 +
1165 +
1166 +
1167 +
1168 +
1169 +
1170 +
1171 +
1172 +
1173 +
1174 +
1175 +
1176 +
1177 +
1178 +
1179 +
1180 +
1181 +
1182 +
1183 +
1184 +
1185 +
1186 +
1187 +
1188 +
1189 +
1190 +
1191 +
1192 +
1193 +
1194 +
1195 +
1196 +
1197 +
1198 +
1199 +
1200 +
1201 +
1202 +
1203 +
1204 +
1205 +
1206 +
1207 +
1208 +
1209 +
1210 +
1211 +
1212 +
1213 +
1214 +
1215 +
1216 +
1217 +
1218 +
1219 +
1220 +
1221 +
1222 +
1223 +
1224 +
1225 +
1226 +
1227 +
1228 +
1229 +
1230 +
1231 +
1232 +
1233 +
1234 +
1235 +
1236 +
1237 +
1238 +
1239 +
1240 +
1241 +
1242 +
1243 +
1244 +
1245 +
1246 +
1247 +
1248 +
1249 +
1250 +
1251 +
1252 +
1253 +
1254 +
1255 +
1256 +
1257 +
1258 +
1259 +
1260 +
1261 +
1262 +
1263 +
1264 +
1265 +
1266 +
1267 +
1268 +
1269 +
1270 +
1271 +
1272 +
1273 +
1274 +
1275 +
1276 +
1277 +
1278 +
1279 +
1280 +
1281 +
1282 +
1283 +
1284 +
1285 +
1286 +
1287 +
1288 +
1289 +
1290 +
1291 +
1292 +
1293 +
1294 +
1295 +
1296 +
1297 +
1298 +
1299 +
1300 +
1301 +
1302 +
1303 +
1304 +
1305 +
1306 +
1307 +
1308 +
1309 +
1310 +
1311 +
1312 +
1313 +
1314 +
1315 +
1316 +
1317 +
1318 +
1319 +
1320 +
1321 +
1322 +
1323 +
1324 +
1325 +
1326 +
1327 +
1328 +
1329 +
1330 +
1331 +
1332 +
1333 +
1334 +
1335 +
1336 +
1337 +
1338 +
1339 +
1340 +
1341 +
1342 +
1343 +
1344 +
1345 +
1346 +
1347 +
1348 +
1349 +
1350 +
1351 +
1352 +
1353 +
1354 +
1355 +
1356 +
1357 +
1358 +
1359 +
1360 +
1361 +
1362 +
1363 +
1364 +
1365 +
1366 +
1367 +
1368 +
1369 +
1370 +
1371 +
1372 +
1373 +
1374 +
1375 +
1376 +
1377 +
1378 +
1379 +
1380 +
1381 +
1382 +
1383 +
1384 +
1385 +
1386 +
1387 +
1388 +
1389 +
1390 +
1391 +
1392 +
1393 +
1394 +
1395 +
1396 +
1397 +
1398 +
1399 +
1400 +
1401 +
1402 +
1403 +
1404 +
1405 +
1406 +
1407 +
1408 +
1409 +
1410 +
1411 +
1412 +
1413 +
1414 +
1415 +
1416 +
1417 +
1418 +
1419 +
1420 +
1421 +
1422 +
1423 +
1424 +
1425 +
1426 +
1427 +
1428 +
1429 +
1430 +
1431 +
1432 +
1433 +
1434 +
1435 +
1436 +
1437 +
1438 +
1439 +
1440 +
1441 +
1442 +
1443 +
1444 +
1445 +
1446 +
1447 +
1448 +
1449 +
1450 +
1451 +
1452 +
1453 +
1454 +
1455 +
1456 +
1457 +
1458 +
1459 +
1460 +
1461 +
1462 +
1463 +
1464 +
1465 +
1466 +
1467 +
1468 +
1469 +
1470 +
1471 +
1472 +
1473 +
1474 +
1475 +
1476 +
1477 +
1478 +
1479 +
1480 +
1481 +
1482 +
1483 +
1484 +
1485 +
1486 +
1487 +
1488 +
1489 +
1490 +
1491 +
1492 +
1493 +
1494 +
1495 +
1496 +
1497 +
1498 +
1499 +
1500 +
1501 +
1502 +
1503 +
1504 +
1505 +
1506 +
1507 +
1508 +
1509 +
1510 +
1511 +
1512 +
1513 +
1514 +
1515 +
1516 +
1517 +
1518 +
1519 +
1520 +
1521 +
1522 +
1523 +
1524 +
1525 +
1526 +
1527 +
1528 +
1529 +
1530 +
1531 +
1532 +
1533 +
1534 +
1535 +
1536 +
1537 +
1538 +
1539 +
1540 +
1541 +
1542 +
1543 +
1544 +
1545 +
1546 +
1547 +
1548 +
1549 +
1550 +
1551 +
1552 +
1553 +
1554 +
1555 +
1556 +
1557 +
1558 +
1559 +
1560 +
1561 +
1562 +
1563 +
1564 +
1565 +
1566 +
1567 +
1568 +
1569 +
1570 +
1571 +
1572 +
1573 +
1574 +
1575 +
1576 +
1577 +
1578 +
1579 +
1580 +
1581 +
1582 +
1583 +
1584 +
1585 +
1586 +
1587 +
1588 +
1589 +
1590 +
1591 +
1592 +
1593 +
1594 +
1595 +
1596 +
1597 +
1598 +
1599 +
1600 +
1601 +
1602 +
1603 +
1604 +
1605 +
1606 +
1607 +
1608 +
1609 +
1610 +
1611 +
1612 +
1613 +
1614 +
1615 +
1616 +
1617 +
1618 +
1619 +
1620 +
1621 +
1622 +
1623 +
1624 +
1625 +
1626 +
1627 +
1628 +
1629 +
1630 +
1631 +
1632 +
1633 +
1634 +
1635 +
1636 +
1637 +
1638 +
1639 +
1640 +
1641 +
1642 +
1643 +
1644 +
1645 +
1646 +
1647 +
1648 +
1649 +
1650 +
1651 +
1652 +
1653 +
1654 +
1655 +
1656 +
1657 +
1658 +
1659 +
1660 +
1661 +
1662 +
1663 +
1664 +
1665 +
1666 +
1667 +
1668 +
1669 +
1670 +
1671 +
1672 +
1673 +
1674 +
1675 +
1676 +
1677 +
1678 +
1679 +
1680 +
1681 +
1682 +
1683 +
1684 +
1685 +
1686 +
1687 +
1688 +
1689 +
1690 +
1691 +
1692 +
1693 +
1694 +
1695 +
1696 +
1697 +
1698 +
1699 +
1700 +
1701 +
1702 +
1703 +
1704 +
1705 +
1706 +
1707 +
1708 +
1709 +
1710 +
1711 +
1712 +
1713 +
1714 +
1715 +
1716 +
1717 +
1718 +
1719 +
1720 +
1721 +
1722 +
1723 +
1724 +
1725 +
1726 +
1727 +
1728 +
1729 +
1730 +
1731 +
1732 +
1733 +
1734 +
1735 +
1736 +
1737 +
1738 +
1739 +
1740 +
1741 +
1742 +
1743 +
1744 +
1745 +
1746 +
1747 +
1748 +
1749 +
1750 +
1751 +
1752 +
1753 +
1754 +
1755 +
1756 +
1757 +
1758 +
1759 +
1760 +
1761 +
1762 +
1763 +
1764 +
1765 +
1766 +
1767 +
1768 +
1769 +
1770 +
1771 +
1772 +
1773 +
1774 +
1775 +
1776 +
1777 +
1778 +
1779 +
1780 +
1781 +
1782 +
1783 +
1784 +
1785 +
1786 +
1787 +
1788 +
1789 +
1790 +
1791 +
1792 +
1793 +
1794 +
1795 +
1796 +
1797 +
1798 +
1799 +
1800 +
1801 +
1802 +
1803 +
1804 +
1805 +
1806 +
1807 +
1808 +
1809 +
1810 +
1811 +
1812 +
1813 +
1814 +
1815 +
1816 +
1817 +
1818 +
1819 +
1820 +
1821 +
1822 +
1823 +
1824 +
1825 +
1826 +
1827 +
1828 +
1829 +
1830 +
1831 +
1832 +
1833 +
1834 +
1835 +
1836 +
1837 +
1838 +
1839 +
1840 +
1841 +
1842 +
1843 +
1844 +
1845 +
1846 +
1847 +
1848 +
1849 +
1850 +
1851 +
1852 +
1853 +
1854 +
1855 +
1856 +
1857 +
1858 +
1859 +
1860 +
1861 +
1862 +
1863 +
1864 +
1865 +
1866 +
1867 +
1868 +
1869 +
1870 +
1871 +
1872 +
1873 +
1874 +
1875 +
1876 +
1877 +
1878 +
1879 +
1880 +
1881 +
1882 +
1883 +
1884 +
1885 +
1886 +
1887 +
1888 +
1889 +
1890 +
1891 +
1892 +
1893 +
1894 +
1895 +
1896 +
1897 +
1898 +
1899 +
1900 +
1901 +
1902 +
1903 +
1904 +
1905 +
1906 +
1907 +
1908 +
1909 +
1910 +
1911 +
1912 +
1913 +
1914 +
1915 +
1916 +
1917 +
1918 +
1919 +
1920 +
1921 +
1922 +
1923 +
1924 +
1925 +
1926 +
1927 +
1928 +
1929 +
1930 +
1931 +
1932 +
1933 +
1934 +
1935 +
1936 +
1937 +
1938 +
1939 +
1940 +
1941 +
1942 +
1943 +
1944 +
1945 +
1946 +
1947 +
1948 +
1949 +
1950 +
1951 +
1952 +
1953 +
1954 +
1955 +
1956 +
1957 +
1958 +
1959 +
1960 +
1961 +
1962 +
1963 +
1964 +
1965 +
1966 +
1967 +
1968 +
1969 +
1970 +
1971 +
1972 +
1973 +
1974 +
1975 +
1976 +
1977 +
1978 +
1979 +
1980 +
1981 +
1982 +
1983 +
1984 +
1985 +
1986 +
1987 +
1988 +
1989 +
1990 +
1991 +
1992 +
1993 +
1994 +
1995 +
1996 +
1997 +
1998 +
1999 +
2000 +
2001 +
2002 +
2003 +
2004 +
2005 +
2006 +
2007 +
2008 +
2009 +
2010 +
2011 +
2012 +
2013 +
2014 +
2015 +
2016 +
2017 +
2018 +
2019 +
2020 +
2021 +
2022 +
2023 +
2024 +
2025 +
2026 +
2027 +
2028 +
2029 +
2030 +
2031 +
2032 +
2033 +
2034 +
2035 +
2036 +
2037 +
2038 +
2039 +
2040 +
2041 +
2042 +
2043 +
2044 +
2045 +
2046 +
2047 +
2048 +
2049 +
2050 +
2051 +
2052 +
2053 +
2054 +
2055 +
2056 +
2057 +
2058 +
2059 +
2060 +
2061 +
2062 +
2063 +
2064 +
2065 +
2066 +
2067 +
2068 +
2069 +
2070 +
2071 +
2072 +
2073 +
2074 +
2075 +
2076 +
2077 +
2078 +
2079 +
2080 +
2081 +
2082 +
2083 +
2084 +
2085 +
2086 +
2087 +
2088 +
2089 +
2090 +
2091 +
2092 +
2093 +
2094 +
2095 +
2096 +
2097 +
2098 +
2099 +
2100 +
2101 +
2102 +
2103 +
2104 +
2105 +
2106 +
2107 +
2108 +
2109 +
2110 +
2111 +
2112 +
2113 +
2114 +
2115 +
2116 +
2117 +
2118 +
2119 +
2120 +
2121 +
2122 +
2123 +
2124 +
2125 +
2126 +
2127 +
2128 +
2129 +
2130 +
2131 +
2132 +
2133 +
2134 +
2135 +
2136 +
2137 +
2138 +
2139 +
2140 +
2141 +
2142 +
2143 +
2144 +
2145 +
2146 +
2147 +
2148 +
2149 +
2150 +
2151 +
2152 +
2153 +
2154 +
2155 +
2156 +
2157 +
2158 +
2159 +
2160 +

```

```

28 +         """
29 +         for key in self.list_keys():
30 +             if key['fingerprint'] == fingerprint:
31 +                 rs = self.r.search(key['uids'][1])
32 +                 key['name'] = rs.group(1)
33 +                 key['email'] = rs.group(2)
34 +                 return key
35 +         return 0
36 +
37 +     def key_is_expired(self, fingerprint):
38 +         """ Checks if a key with a specific fingerprint is expired
39 +         Raises ValueError if there is no key with that fingerprint
40 +         """
41 +         key = self.get_key(fingerprint)
42 +         if key == 0:
43 +             raise ValueError("There is no key with fingerprint: %s" % fingerprint)
44 +
45 +         if (key['expires'] == "") or (float(key['expires']) > time()):
46 +             return False
47 +         else:
48 +             return True
49 +
50 +     def gen_key(self, input):
51 +         """Generate a key; you might use gen_key_input() to create the
52 +         control input.
53 +         @@ -682,6 +711,12 @@
54 +             import_res = value.split()
55 +             for i in range(len(self.counts)):
56 +                 setattr(self, self.counts[i], int(import_res[i]))
57 +         elif key == "KEYEXPIRED":
58 +             self.results.append({'fingerprint': None,
59 +                                 'problem': '0', 'text': 'Key or subkey has expired'})
60 +         elif key == "SIGEXPIRED":
61 +             self.results.append({'fingerprint': None,
62 +                                 'problem': '0', 'text': 'Key or subkey has expired'})
63 +         else:
64 +             raise ValueError("Unknown status message: %r" % key)

```

Βιβλιογραφία

- [1] Steven Furnel, Sokratis Katsikas, Javier Lopez, Ahmed Patel, “Securing Information and Communications Systems: Principles, Technologies, and Applications”, Chapter 15: Electronic Voting Systems, Artech House, 2008
- [2] Andreas Mitrakas, Pim Hengeveld, Despina Polemi & Johann Gamper, “Secure E-government web services”, Idea Group Publishing, 2007
- [3] “Legal, operational and technical standards for e-voting”, Recommendation Rec(2004)11 and explanatory memorandum, Council of Europe Publishing, April 2005
- [4] Gritzalis, Dimitris A., “Secure Electronic Voting”, Kluwer Academic Publishers Group, 2003
- [5] Mitrou L., Gritzalis D., Katsikas S., “Revisiting legal and regulatory requirements for secure e-voting”, May 2002
[http://www.instore.gr/evote/evote_end/htm/3public/doc3/public/evote_paper_SEC_2002_2.doc]
- [6] Safevote, Inc. and THE BELL, “Voting System Requirements”, February 2001
[<http://www.thebell.net/papers/vote-req.pdf>]
- [7] J. Callas, L. Donnerhacke, H. Finnelly, D. Shaw, R. Thayer, “OpenPGP Message Format”, Network Working Group, Request for Comments: 4880, November 2007
[<http://www.ietf.org/rfc/rfc4880.txt>]
- [8] “Introduction to Cryptography”, Chapter 1
[<http://www.pgpi.org/doc/pgpintro/>]
- [9] “The GNU Privacy Guard Manual”, version 2.0.15, March 2010
[<http://www.gnupg.org/documentation/manuals/gnupg/>]