ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΚΡΥΠΤΟΓΡΑΦΙΑ

(2021 – 2022)

Project: Machine Learning Classification over Encrypted Data

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 031 17 176

Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr
- chris99ts@gmail.com

1. Introduction

Οι ταξινομητές είναι ένα ιδιαίτερα χρήσιμο εργαλείο για πολλές διεργασίες, όπως ιατρικές προβλέψεις, οι οποίες απαιτούν χειρισμό ευαίσθητης πληροφορίας (sensitive data) και για αυτό το λόγο είναι σημαντικό τα δεδομένα και ο ταξινομητής να παραμένουν ιδιωτικά.

Οι αλγόριθμοι Επιβλεπόμενης Μάθησης αποτελούνται από την φάση την εκπαίδευσης (training) κατά την διάρκεια της οποίας μαθαίνει ένα μοντέλο w από ένα data set με παραδείγματα με ένδειξη (labeled examples) και από την φάση της ταξινόμησης κατά την οποία ο ταξινομητής C εκτελείται πάνω σε ένα διάνυσμα χαρακτηριστικών (feature vector) x, που δεν έχει ξαναδεί οπότε το ταξινομεί με βάση το μοντέλο w και επιστρέφει μια πρόβλεψη (prediction) C(x, w). Στις εφαρμογές που χειρίζονται ευαίσθητα δεδομένα, είναι σημαντικό το διάνυσμα x και το μοντέλο w να παραμένουν μυστικά.

Εν προκειμένω, ο στόχος αυτός ονομάζεται privacy-preserving classification. Ειδικότερα, ο χρήστης έχει μια ιδιωτική είσοδο, δηλαδή ένα διάνυσμα χαρακτηριστικών x και ο server έχει επίσης, μια ιδιωτική είσοδο, δηλαδή ένα ιδιωτικό μοντέλο w. Ο τρόπος που αποκτά κανείς το μοντέλο w είναι ανεξάρτητος από τα Πρωτόκολλα που θα αναλυθούν. Μόνο η ταξινόμηση πρέπει να διατηρεί την ιδιωτικότητα (privacy-preserving), δηλαδή ο χρήστης να μάθει την πρόβλεψη C(x, w) χωρίς να μάθει τίποτα για το μοντέλο w, ενώ ταυτόχρονα, ο server δεν πρέπει να μάθει τίποτα για την είσοδο που έδωσε ο χρήστης ή το αποτέλεσμα της ταξινόμησης.

Αναλυτικότερα, κατασκευάζονται αποδοτικά privacy-preserving Πρωτόκολλα για τους ταξινομητές hyperplane decision, Naive Bayes και decision trees, καθώς επίσης και έναν πιο γενικό ταξινομητή που τους συνδυάζει χρησιμοποιώντας AdaBoost. Σημειώνεται ότι, αν και εξετάζονται Πρωτόκολλα για τους τρεις ταξινομητές που αναφέρθηκαν, χρησιμοποιήθηκαν και άλλοι ή συνδυασμοί αυτών για την ρύθμισή τους.

Παρόλο που ένας generic secure multi-party υπολογισμός μπορεί να υλοποιήσει οποιονδήποτε ταξινομητή, λόγω της γενικότητάς τους, τέτοια συστήματα δεν είναι αποτελεσματικά για τους κοινούς ταξινομητές καθώς είτε ξεμένουν από μνήμη είτε εκτελούνται πιο αργά. Έτσι, τα Πρωτόκολλα που εστιάζουν στο πρόβλημα ταξινόμησης, υπόσχονται καλύτερη επίδοση. Ωστόσο, η ήδη υπάρχουσα έρευνα πάνω στον τομέα της Μηχανικής Μάθησης και της Ιδιωτικότητας εστιάζει στην διατήρηση της ιδιωτικότητας κατά την διάρκεια της φάσης της εκπαίδευσης και δεν αναφέρεται στην ταξινόμηση ενώ η έρευνα που έχει γίνει πάνω σε αυτό είτε χρησιμοποιεί ασθενέστερη ασφάλεια κατά την οποία ο χρήστης μαθαίνει το μοντέλο, είτε εστιάζει σε συγκεκριμένους ταξινομητές που χρησιμοποιείται σε ειδικές περιπτώσεις.

Ο σχεδιασμός αποδοτικών privacy-preserving ταξινομητών αντιμετωπίζει τις εξής προκλήσεις. Πρώτον, ο υπολογισμός των ευαίσθητων δεδομένων πάνω σε κάποιον ταξινομητή είναι ιδιαίτερα σύνθετος, γεγονός που το καθιστά δύσκολο για υποστήριξη. Δεύτερον, η παροχή μιας λύσης που να είναι γενικότερη και από τους τρεις ταξινομητές.

Η αντιμετώπιση των παραπάνω προκλήσεων έγινε χρησιμοποιώντας τις εξής τεχνικές. Η πρώτη είναι η αναγνώριση ενός set από κύριους τελεστές (set of core operations) πάνω σε κρυπτογραφημένα δεδομένα που υπόκεινται σε πολλά Πρωτόκολλα Ταξινόμησης. Αυτοί οι τελεστές είναι το argmax και το dot product.

Η δεύτερη είναι ο σχεδιασμός των building blocks με έναν συνθετικό (composable) τρόπο, λαμβάνοντας υπόψιν τόσο την λειτουργικότητα αλλά και την ασφάλεια. Για την επίτευξη αυτού του στόχου, χρησιμοποιούνται οι εξής τεχνικές. Αρχικά, η είσοδος και η έξοδος για όλα τα building blocks είναι κρυπτογραφημένη με ομομορφική (homomorphic) κρυπτογράφηση. Έτσι, παρέχεται ένας μηχανισμός για την εναλλαγή από ένα σύστημα σε ένα άλλο. Έπειτα, το API αυτών των blocks είναι εύκαμπτο (flexible) ώστε να επιλέγεται ποιος θα επεξεργάζεται το κάθε τι. Τέλος, η ασφάλεια αυτών των πρωτοκόλλων έγκειται στην χρήση αρθρωτής διαδοχικής σύνθεσης (modular sequential composition).

Επισημαίνεται ότι η συνεισφορά της βιβλιοθήκης των building blocks μπορεί να επεκταθεί και εκτός των ταξινομητών που χρησιμοποιούνται στο paper. Όλα τα Πρωτόκολλα που χρησιμοποιούνται είναι ασφαλή σε επιθέσεις παθητικών αντιπάλων (passive adversaries). Μάλιστα, παρέχεται και μια υλοποίηση και μια εκτίμηση για ένα μοντέλο ενώ παράλληλα, το paper αναλύει όλα τα βήματα που αναφέρθηκαν, σε βάθος.

2. Related work

Αυτό το paper αποτελεί το πρώτο που παρουσιάζει αποτελεσματικά privacy-preserving Πρωτόκολλα για μια μεγάλη γκάμα από ταξινομητές. Αν και υπάρχουν ήδη ασφαλή two-party υπολογιστικά Πρωτόκολλα για generic functions στην θεωρία και στην πράξη, απαιτούν βαρύ κρυπτογραφικό υπολογισμό οπότε η εφαρμογή τους στο εν λόγω πρόβλημα δεν θα ήταν αποδοτική. Η έρευνα που έχει ήδη γίνει επί του θέματος χωρίζεται σε δυο κατηγορίες: (i) τεχνικές για privacy-preserving εκπαίδευση και (ii) τεχνικές για privacy-preserving ταξινόμηση. Το παρόν paper εξετάζει στην δεύτερη κατηγορία και επεκτείνεται και για building blocks. Αξίζει να σημειωθεί ότι η δουλειά που έχει γίνει (απόκρυψη της εισόδου κάθε χρήστη κατά την φάση ταξινόμησης) είναι συμπληρωματική με την διαφορική ασφάλεια (differential privacy) (δημιουργία ταξινομητών/μοντέλων από ευαίσθητα δεδομένα εκπαίδευσης των χρηστών που όμως διαρρέουν ένα μέρος της πληροφορίας στο data set εκπαίδευσης).

2.1 Privacy-preserving training

Ένα σύνολο από τεχνικές έχουν αναπτυχθεί για privacy-preserving αλγόριθμους εκπαίδευσης όπως Naive Bayes, decision trees, linear discriminant ταξινομητές αλλά και πιο γενικές kernel μέθοδοι. Στο paper του Grapel εξηγείται πως μπορεί να γίνει η εκπαίδευση διάφορων ταξινομητών μηχανικής μάθησης χρησιμοποιώντας ένα ομομορφικό σύστημα κρυπτογράφησης. Εστιάζει σε μερικούς απλούς ταξινομητές και δεν επεκτείνεται σε πιο σύνθετους αλγορίθμους όπως τα SVMs. Επίσης, αναφέρεται στην ιδιωτική ταξινόμηση αλλά με ένα μοντέλο με ασθενέστερη ασφάλεια, καθώς ο χρήστης μαθαίνει περισσότερα για το μοντέλο και όχι απλώς και μόνο το τελικό αποτέλεσμα. Πράγματι, η τελευταία σύγκριση με πλήρως ομομορφική κρυπτογράφηση (FHE) από μόνη της είναι αναποτελεσματική αλλά αυτό το ζήτημα ξεπερνιέται με μια διαδραστική ρύθμιση που θα αναλυθεί στην συνέχεια.

2.2 Privacy-preserving classification

Η έρευνα που έχει διεξαχθεί γύρω από το γενικότερο πρόβλημα της privacy-preserving ταξινόμησης στην πράξη είναι περιορισμένη καθώς προηγούμενες εργασίες έχουν στοχεύσει σε ασθενέστερη ασφάλεια ή/και σε συγκεκριμένους ταξινομητές.

Στον Bos, ένα third party μπορεί να κάνει ιατρικές προβλέψεις πάνω σε κρυπτογραφημένα δεδομένα ενός ασθενή χρησιμοποιώντας πλήρως ομομορφική κρυπτογράφηση, όμως όλοι μπορεί να γνωρίζουν το μοντέλο και ο αλγόριθμος κρύβει μόνο την είσοδο από το cloud. Αντιθέτως, στο paper κρύβεται και το μοντέλο. Έτσι, οι αλγόριθμοί τους δεν μπορούν να εφαρμοστούν στην περίπτωση που εξετάζεται εδώ επειδή διαρρέουν πολλές πληροφορίες αλλά και γιατί οι τεχνικές που χρησιμοποιούνται εδώ διαφέρουν.

Στον Barni, δημιουργείται ασφαλής εκτίμηση πάνω σε linear branching προγράμματα τα οποία χρησιμοποιούν για να εφαρμόσουν έναν ασφαλή ταξινομητή για ECG σήματα. Η τεχνική τους βασίζεται σε fine-tuned garbled κυκλώματα. Αντιθέτως, στο paper δεν υπάρχει περιορισμός μόνο σε branching προγράμματα και η εκτίμηση δείχνει ότι είναι πολύ ταχύτερη πάνω σε αυτά. Επιπλέον, σε μια άλλη εργασία του Barni εξετάζονται ασφαλείς ταξινομητές πάνω σε Νευρωνικά Δίκτυα αλλά και αυτή η περίπτωση καλύπτεται από το paper.

Άλλες εργασίες δημιουργούν συγκεκριμένους ταξινομητές για αναγνώριση προσώπων ή εντοπισμό ενώ το paper παρέχει ένα γενικό σύνολο από ταξινομητές και building blocks για την δημιουργία πιο σύνθετων ταξινομητών.

2.3 Work related to our building blocks

Δύο από τα βασικά στοιχεία που χρησιμοποιούνται είναι η ιδιωτική σύγκριση (private comparison) και ο ιδιωτικός υπολογισμός του dot product και η ανάλυση για το πως σχεδιάστηκαν φαίνεται παρακάτω.

3. Background and preliminaries

3.1 Classification in machine learning algorithms

Ο χρήστης δίνει μια είσοδο x το οποίο είναι ένα διάνυσμα d στοιχείων $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ και ονομάζεται διάνυσμα χαρακτηριστικών. Για την ταξινόμηση του x θα πρέπει να γίνει εκτίμηση με την συνάρτηση ταξινόμησης $C_w: \mathbb{R}^d \mapsto \{c_1, \dots, c_k\}$. Η έξοδος θα είναι $c_{k^*} = C_w(x)$, όπου $k^* \in \{1, \dots k\}$ και c_{k^*} είναι η κλάση που αντιστοιχεί το x, με βάση το μοντέλο w. Για ευκολία στον συμβολισμό, αντί για c_{k^*} θα χρησιμοποιείται το k^*, δηλαδή $k^* = C_w(x)$.

Στην συνέχεια, περιγράφεται ο τρόπος λειτουργίας των τριών δημοφιλέστερων ταξινομητών πάνω σε κανονικά, μη κρυπτογραφημένα δεδομένα. Διαφέρουν στο μοντέλο w και στην συνάρτηση C_w.

Hyperplane decision-based ταξινομητές: Το μοντέλο w αποτελείται από k διανύσματα στο $\mathbb{R}^d$ $\left(w = \{w_i\}_{i=1}^k\right)$. Ο ταξινομητής είναι:

$$k^* = \underset{i \in [k]}{argmax} \langle w_i, x \rangle$$

Όπου, $\langle w_i, x \rangle$ συμβολίζει το εσωτερικό γινόμενο.

Αυτός ο ταξινομητής τυπικά δουλεύει πάνω σε έναν υποθετικό χώρο $\mathcal{H}$ στον οποίο ισχύει το εσωτερικό γινόμενο. Συχνά επιλύει ένα πρόβλημα δυαδικής ταξινόμησης (k = 2): δίνεται είσοδος x, το x ταξινομείται στην κλάση c_2 αν $\langle w, \varphi(x) \rangle \geq 0$, αλλιώς ταξινομείται στην c_1. Εδώ, το $\varphi: \mathbb{R}^d \mapsto \mathcal{H}$ συμβολίζει την αντιστοίχιση των χαρακτηριστικών από τον $\mathbb{R}^d$ στον $\mathcal{H}$. Συγκεκριμένα, εδώ εξετάζεται η περίπτωση που $\mathcal{H} = \mathbb{R}^d$ και σημειώνεται ότι μια μεγάλη κλάση από απειροδιάστατους χώρους μπορεί να προσεγγιστεί με έναν πεπερασμένων διαστάσεων χώρο, συμπεριλαμβανομένου του γνωστού Gaussian πυρήνα (RBF). Τότε, $\varphi(x) = x$ ή $\varphi(x) = Px$ για έναν randomized projection matrix P που επιλέγεται κατά την εκπαίδευση. Σημειώνεται ότι το Px αποτελείται εξ ολοκλήρου από εσωτερικά γινόμενα. Για την γενίκευση σε k κλάσεις, χρησιμοποιείται η προσέγγιση one-versus-all, κατά την οποία k διαφορετικά μοντέλα $\{w_i\}_{i=1}^k$ εκπαιδεύονται για τον διαχωρισμό κάθε κλάση από τις υπόλοιπες. Αυτή η δομή είναι αρκετά γενική ώστε να καλύψει πολλούς αλγορίθμους, όπως SVMs, Logistic Regression & Least Squares.

Naive Bayes ταξινομητές: Το μοντέλο w αποτελείται από πολλές πιθανότητες: Την πιθανότητα ότι κάθε κλάση c_i προκύπτει ως $\{p(C = c_i)\}_{i=1}^{k}$ και τις πιθανότητες ότι ένα στοιχείο x_j του x προκύπτει μέσα σε μια συγκεκριμένη κλάση c_i. Η τελευταία, είναι η πιθανότητα το j-οστό στοιχείο του x_j του x να είναι v όταν το x ανήκει στην κατηγορία c_i και συμβολίζεται ως $\left\{\left\{\{p(X_j = v|C = c_i)\}_{v\in D_j}\right\}_{j=1}^{d}\right\}_{i=1}^{k}$ όπου D_j είναι το domain του X_j. Η συνάρτηση ταξινόμησης χρησιμοποιεί τον maximum a posteriori κανόνα απόφασης και επιστρέφει την κλάση με την μεγαλύτερη posterior πιθανότητα:

$$k^* = \underset{i\in[k]}{\operatorname{arg\,max}}\, p(C = c_i|X = x) = \underset{i\in[k]}{\operatorname{arg\,max}}\, p(C = c_i, X = x)$$

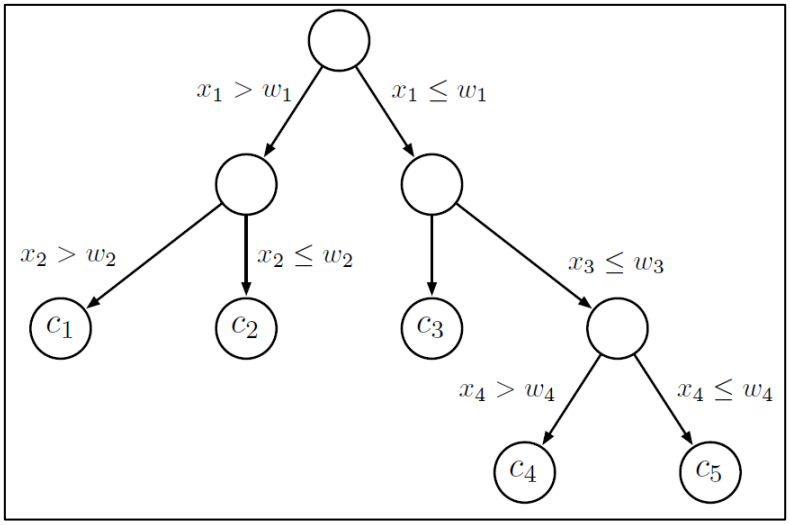
$$= \underset{i\in[k]}{\operatorname{arg\,max}}\, p(C = c_i, X_1 = x_1, \dots, X_d = x_d)$$

Αυτό το μοντέλο υποθέτει ότι το $p(C = c_i, X = x)$ έχει την εξής παραγοντοποίηση:

$$p(C = c_i, X_1 = x_1, \dots, X_d = x_d) = p(C = c_i)\prod_{j=1}^{d} p(X_j = x_j|C = c_i)$$

Δηλαδή, κάθε ένα από τα d χαρακτηριστικά είναι ανεξάρτητα υπό συνθήκη για δεδομένα κλάση. Για απλότητα, γίνεται η υπόθεση ότι το domain των τιμών των χαρακτηριστικών, τα x_i, είναι διακριτά και πεπερασμένα, οπότε οι $p(X_j = x_j|C = c_i)$ είναι μάζες πιθανοτήτων (probability masses).

Decision trees: Αποτελούν μη παραμετρικούς ταξινομητές και λειτουργούν με το να διαμερίζουν τον χώρο του διανύσματος χαρακτηριστικών με ένα χαρακτηριστικό την φορά (οι εσωτερικοί κόμβοι αποτελούν τους κανόνες διαμέρισης και τα φύλλα αποτελούν τις ετικέτες κάθε τάξης). Ένα διάνυσμα x ταξινομείται με βάση αυτό το δέντρο, χρησιμοποιώντας τον κανόνα διαμέρισης σε κάθε κόμβο ώστε να αποφασίσει ποιο κλαδί θα επιλέξει μέχρι να φτάσει στα φύλλα.



3.2 Cryptographic preliminaries

3.2.1 Cryptosystems

Στο paper χρησιμοποιούνται τρία προσθετικά ομομορφικά κρυπτοσυστήματα. Ένα σύστημα κρυπτογράφησης δημοσίου κλειδιού HE είναι προσθετικά ομομορφικό αν, για δοσμένα κρυπτογραφημένα μηνύματα HE.Enc(a) και HE.Enc(b), υπάρχει ένας τελεστής δημοσίου κλειδιού $\oplus$ έτσι ώστε το HE.Enc(a) $\oplus$ HE.Enc(b) να είναι η κρυπτογράφηση του a + b. Αυτά είναι ομομορφικά μόνο για την πρόσθεση και για αυτό το λόγο είναι αποτελεσματικά, σε αντίθεση με την πλήρως ομομορφική κρυπτογράφηση που υποστηρίζει όλες τις συναρτήσεις. Έτσι, τα συστήματα θα είναι:

1. Quadratic Residuosity (QR) κρυπτοσύστημα των Goldwasser-Micali:

Το κρυπτοσύστημα των Goldwasser-Micali στηρίζεται στο πρόβλημα των τετραγωνικών υπολοίπων, σύμφωνα με το οποίο δίνονται δύο ακέραιοι a και N, όπου ο N είναι το γινόμενο δύο πρώτων αριθμών και πρέπει να αποφασιστεί αν ο a είναι τετραγωνικό υπόλοιπο του N ή αλλιώς, αν υπάρχει αριθμός b ώστε $a \equiv b^2 (mod\ N)$.

Για την περιγραφή του τετραγωνικού υπολοίπου μπορούν να χρησιμοποιηθούν και τα σύμβολα Legendre και Jacobi, τα οποία θα χρησιμοποιήσουμε και παρακάτω. Συγκεκριμένα, το σύμβολο Legendre, με p περιττό και a ακέραιο, συμβολίζεται ως εξής:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, \exists x: x^2 \equiv a (mod\ p) \\ -1, \nexists x: x^2 \equiv a (mod\ p) \\ 0, p|a \end{cases}$$

Το σύμβολο Jacobi αποτελεί γενίκευση του συμβόλου Legendre. Ορίζεται για ακέραιο a και n, όπου αν ο n είναι γινόμενο των πρώτων παραγόντων έτσι ώστε $n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$, τότε:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{a_1} \left(\frac{a}{p_2}\right)^{a_2} \dots \left(\frac{a}{p_k}\right)^{a_k}$$

- Παραγωγή Κλειδιού: Στην αρχή, η Alice παράγει το δημόσιο και το ιδιωτικό κλειδί. Αυτό γίνεται παράγοντας δύο τυχαίους ανεξάρτητους πρώτους αριθμούς p και q, μεγάλους σε μέγεθος. Μετά, υπολογίζει τον N = p·q. Τέλος υπολογίζεται ο αριθμός x, ως το μη τετραγωνικό υπόλοιπο ως προς p και q. Έτσι, $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$. Οπότε, $\left(\frac{x}{N}\right) = \left(\frac{x}{p}\right)\left(\frac{x}{q}\right) = 1$. Οι αριθμοί (x, N) αποτελούν το δημόσιο κλειδί και οι (p, q) το ιδιωτικό κλειδί.
- Κρυπτογράφηση: Έστω ότι ο Bob θέλει να στείλει ένα μήνυμα $m = (m_1, \dots, m_n)$ στην Alice. Αρχικά, το μήνυμα τροποποιείται ώστε να είναι σε binary bit format. Έπειτα, για κάθε bit m_i, παράγει μια τιμή y_i, με $y_i \in U(\mathbb{Z}_m)$ (δηλ. $\gcd(y_i, N) = 1$. Έτσι, για κάθε bit παράγει $c_i = y_i^2 x^{m_i}$. Το ciphertext θα είναι το $c = (c_1, \dots, c_n)$.
- Αποκρυπτογράφηση: Εφόσον η Alice, που έχει το ιδιωτικό κλειδί, λάβει το ciphertext c, προχωράει στην αποκρυπτογράφησή του. Για κάθε c_i, υπολογίζει αν αποτελεί τετραγωνικό υπόλοιπο του N χρησιμοποιώντας την παραγοντοποίησή του σε πρώτους παράγοντες, οι οποίοι αποτελούν το ιδιωτικό κλειδί. Αν είναι τετραγωνικό υπόλοιπο, τότε το αντίστοιχο m_i είναι μηδέν, αλλιώς είναι ίσο με ένα. Έτσι, ανακατασκευάζεται το μήνυμα $m = (m_1, \dots, m_n)$.
- Ασφάλεια: Έγκειται στην δυσκολία να αποφανθεί κανείς αν μία τιμή mod N με σύμβολο Jacobi+1 είναι τετραγωνικό υπόλοιπο. Επιπλέον, λόγω της τυχαιότητας, κάθε φορά μπορεί να προκύψει διαφορετικό κρυπτοκείμενο για ίδια κείμενα.

2. Paillier κρυπτοσύστημα:

Το κρυπτοσύστημα Paillier χρησιμοποιεί το πρόβλημα του n-οστού υπολοίπου. Ένας αριθμός α ονομάζεται n-οστό υπόλοιπο $mod\ n^2$ αν $\exists b \in \mathbb{Z}_{n^2}^*$ έτσι ώστε $a \equiv b^n (mod\ n^2)$. Επίσης, έστω η συνάρτηση $L(x,n) = \frac{x-1}{n}$ η οποία θα χρησιμοποιηθεί στην συνέχεια.

- Παραγωγή κλειδιού: Στην αρχή, επιλέγονται δύο τυχαίοι και ανεξάρτητοι μεταξύ τους πρώτοι αριθμοί p και q. Μετά, υπολογίζεται ο αριθμός n = p·q ενώ ορίζεται και ο $\lambda = \lambda(n) = lcm(p-1, q-1)$. Ακόμη, ορίζεται το σύνολο $B_a = \{g \in \mathbb{Z}_{n^2}^* | ord(g) = n \cdot a\}$ και με B η διακεκριμένη ένωσή τους (disjoint union) για $a = 1, \dots, \lambda$. Επίσης, τίθεται $L(x) = L(x,n)$ για ευκολία. Παράλληλα, επιλέγεται τυχαία μια βάση g, όπου $g \in B$. Αποδεικνύεται πως όταν ορίζεται και ο αριθμός $\left(L\left(g^\lambda\ mod\ n^2\right)\right)^{-1}\ mod\ n$, τότε το δημόσιο κλειδί θα είναι οι αριθμοί (n, g) και λ θα είναι το ιδιωτικό.
- Κρυπτογράφηση: Έστω κείμενο m. Επιλέγεται ένας τυχαίος αριθμός r < n όπου $r \in \mathbb{Z}_n^*$ και παράγεται το κρυπτογραφημένο κείμενο ως εξής: $c = g^m r^n\ mod\ n^2$.
- Αποκρυπτογράφηση: Για να υπολογιστεί το αρχικό κείμενο ισχύει η παρακάτω σχέση:

$$m = L\left(c^\lambda\ mod\ n^2\right)\left(L\left(g^\lambda\ mod\ n^2\right)\right)^{-1}\ mod\ n$$

3. Leveled Fully Homomorphic Encryption (FHE) σύστημα, HELib:

Αυτό το κρυπτογραφικό σύστημα είναι μια υλοποίηση ενός leveled fully homomorphic encryption (FHE) scheme από την IBM που λέγεται HELib. Τα FHE ανακαλύφθηκαν το 2009 και είναι κρυπτογραφικά συστήματα τα οποία επιτρέπουν την εφαρμογή όλων των δυνατών πράξεων στο plaintext χωρίς την ανάγκη να αποκρυπτογραφηθούν. Η λέξη leveled υποδεικνύει ότι εμπεριέχονται κάποιοι περιορισμοί στο σύστημα για χάρη όμως της υπολογιστικής βελτίωσης του συστήματος.

3.2.2 Cryptographic assumptions

Αποδεικνύεται ότι τα πρωτόκολλα είναι ασφαλή για σημασιολογική ασφάλεια των παραπάνω κρυπτοσυστημάτων. Αυτά τα κρυπτοσυστήματα βασίζονται σε standard και πολυμελετημένες υποθέσεις: την Quadratic Residuosity υπόθεση, την Decisional Composite Residuosity υπόθεση και την Ring Learning With Error (RLWE) υπόθεση.

3.2.3 Adversarial model

Αποδεικνύεται ότι τα πρωτόκολλα που χρησιμοποιούν two-party computation framework για παθητικούς αντιπάλους είναι ασφαλή. Ένας παθητικός αντίπαλος, σε high level, είναι ένα party, έστω party A το οποίο είναι εκτεθειμένο σε ένα τέτοιο αντίπαλο ο οποίο προσπαθεί να μάθει ιδιωτικές πληροφορίες για την είσοδο στο άλλο party παρατηρώντας τις πληροφορίες που λαμβάνει το party A αλλά δεν μπορεί να αποτρέψει το party A από το να ακολουθήσει το πρωτόκολλο πιστά. Για την σύνθεση ποικίλων πρωτοκόλλων σε ένα μεγαλύτερο πρωτόκολλο με ασφάλεια, χρησιμοποιείται modular sequential composition.

3.3 Notation

Όλα τα πρωτόκολλα είναι ανάμεσα σε 2 parties, τα A και B για τα building blocks και στα C (client) και S (server) για όλους τους ταξινομητές.

Οι είσοδοι και οι έξοδοι των building blocks είναι κρυπτογραφημένες με προσθετικά ομομορφικό κρυπτοσύστημα. Ο χώρος του plaintext του QR είναι $\mathbb{F}_2$ (bits) και με [b] συμβολίζεται ένα bit b κρυπτογραφημένο με QR. Ο χώρος του plaintext του Paillier είναι $\mathbb{Z}_N$, όπου N είναι το public modulus του Paillier και με $[\![m]\!]$ συμβολίζεται ένα ακέραιος m κρυπτογραφημένος με Paillier. Ο χώρος του plaintext του FHE είναι $\mathbb{F}_2$ και με SK_P, PK_P συμβολίζονται το μυστικό και το δημόσιο κλειδί για Paillier, αντίστοιχα. Τέλος, με SK_{QR}, PK_{QR} συμβολίζονται το μυστικό και το δημόσιο κλειδί για QR, αντίστοιχα. Για μια σταθερά b, το a ← b σημαίνει ότι στο α ανατίθεται η τιμή του b. Για μια κατανομή D, a ← D σημαίνει ότι το α παίρνει δείγμα από το D.

4. Building Blocks

Σε αυτό το στάδιο, αναπτύσσεται μια βιβλιοθήκη από building blocks τα οποία χρησιμοποιούνται για την δημιουργία των ταξινομητών παρακάτω. Αυτή η βιβλιοθήκη υποστηρίζει και άλλους ταξινομητές πέρα από αυτούς που εξετάζονται εδώ.

Type	Input A	Input B	Output A	Output B	Implementation
1	PK_P, PK_{QR}, a	SK_P,SK_{QR}, b	$[a < b]$	–	Sec. 4.1.1
2	PK_P, SK_{QR}, $[\![a]\!]$, $[\![b]\!]$	SK_P,PK_{QR}	–	$[a \leq b]$	Sec. 4.1.2
3	PK_P, SK_{QR}, $[\![a]\!]$, $[\![b]\!]$	SK_P,PK_{QR}	$a \leq b$	$[a \leq b]$	Sec. 4.1.2
4	PK_P, PK_{QR}, $[\![a]\!]$, $[\![b]\!]$	SK_P,SK_{QR}	$[a \leq b]$	–	Sec. 4.1.3
5	PK_P, PK_{QR},$[\![a]\!]$, $[\![b]\!]$	SK_P,SK_{QR}	$[a \leq b]$	$a \leq b$	Sec. 4.1.3

4.1 Comparison

Σε αυτό το μέρος αναλύεται το πρωτόκολλο σύγκρισης που χρησιμοποιείται. Για να μπορέσει αυτό το πρωτόκολλο να χρησιμοποιηθεί σε ένα εύρος ταξινομητών θα πρέπει να είναι flexible, δηλαδή να υποστηρίζει μια πληθώρα επιλογών που αφορούν το ποιο party παίρνει την είσοδο, ποιο party παίρνει την έξοδο και αν η είσοδος και η έξοδος θα είναι κρυπτογραφημένη ή όχι. Σε κάθε περίπτωση, το κάθε party δεν θα πρέπει να μαθαίνει τίποτα άλλο για τις εισόδους των υπόλοιπων party εκτός από αυτά που φαίνονται στον παραπάνω πίνακα ως έξοδος. Κάθε σειρά του πίνακα υλοποιείται τροποποιώντας ήδη υπάρχοντα πρωτόκολλα. Υπάρχουν τουλάχιστον δυο προσεγγίσεις για αποτελεσματική σύγκριση: η χρήση ειδικής ομομορφικής κρυπτογράφησης και η χρήση garbled κυκλωμάτων. Από την εμπειρική σύγκριση της απόδοσης αυτών των προσεγγίσεων προκύπτει ότι η πρώτη είναι πιο αποτελεσματική για σύγκριση κρυπτογραφημένων τιμών ενώ η δεύτερη είναι καλύτερη για σύγκριση μη κρυπτογραφημένων τιμών.

4.1.1 Comparison with unencrypted inputs

Για την σύγκριση μη κρυπτογραφημένων εισόδων χρησιμοποιούνται garbled κυκλώματα με state-of-the-art garbling σύστημα του Bellare, με μικρό κύκλωμα για σύγκριση του Kolesnikov και με τον oblivious transfer (OT) σύστημα των Naor & Pinkas. Εφόσον πολλά από τα υπόλοιπα building blocks αναμένουν εισόδους κρυπτογραφημένες με ομομορφική κρυπτογράφηση, θα πρέπει επιπλέον να γίνει μετατροπή από μια garbled έξοδο σε ομομορφική κρυπτογράφηση για την ενεργοποίηση της σύνθεσης. Αυτό υλοποιείται εύκολα χρησιμοποιώντας την τεχνική random shares. Οι τεχνικές αυτές δίνουν το επιθυμητό πρωτόκολλο σύγκρισης. Μάλιστα, με κατάλληλο συνδυασμό μπορεί να υλοποιηθεί ένα ακόμα πιο αποδοτικό πρωτόκολλο: χρήση ενισχυμένου κυκλώματος σύγκρισης το οποίο επίσης παίρνει ως είσοδο ένα masking bit. Μετά, με garbled κύκλωμα και OT, το A θα υπολογίσει (a < b) $\oplus$ c, όπου c είναι ένα bit που επιλέγεται τυχαία από το B. Το B παρέχει κρυπτογράφηση [c] του c, ώστε ο A να υπολογίσει το [a < b] χρησιμοποιώντας τις ομομορφικές ιδιότητες του QR. Η ανάλυση αυτή αφορά την Γραμμή 1.

Oblivious Transfer: Το oblivious transfer είναι ένα πρωτόκολλο στο οποίο ο αποστολέας μεταφέρει ένα από κάποια πιθανά σημεία πληροφορίας στον παραλήπτη, με τρόπο ώστε να προστατεύει και τους δύο εμπλεκόμενους, ενώ ο αποστολέας δεν γνωρίζει ποιο κομμάτι πληροφορίας έστειλε, και ο παραλήπτης δεν λαμβάνει περισσότερη πληροφορία από όση δικαιούται. Εδώ μελετάται η 1-out-of-2 Oblivious Transfer, όπου ο αποστολέας έχει είσοδο που αποτελείται από 2 ακολουθίες (M_1, M_2), ενώ ο παραλήπτης έχει παρέχει ένα bit σ. Ο παραλήπτης θα πρέπει να μάθει την M_σ και ταυτόχρονα καμία πληροφορία για την $M_{1-\sigma}$, ενώ ο αποστολέας δεν πρέπει να μάθει τίποτα για το bit σ.

Garbling Circuit: Τα garbling circuits είναι ένα πρωτόκολλο το οποίο επιτρέπει σε δύο μέλη τον υπολογισμό μιας συνάρτησης (Boolean Circuit) με εισόδους ιδιωτικά (κρυπτογραφημένα) δεδομένα, χωρίς την ύπαρξη ενός τρίτου μέλους και χωρίς να μάθει ο ένας την είσοδο του άλλου. Αποτελούν ουσιαστικά circuits, όπου κάθε καλώδιο "μεταφέρει" κάποια ακολουθία αντί ενός bit. Στην παρούσα δημοσίευση χρησιμοποιήθηκε η βελτιστοποίηση που προτάθηκε από τον Bellarke, με την χρήση fixed-key-blockcipher.

4.1.2 Comparison with encrypted inputs

Οι ταξινομητές επιπλέον απαιτούν την ικανότητα σύγκρισης μεταξύ δυο κρυπτογραφημένων εισόδων. Συγκεκριμένα, έστω ότι το party A θέλει να συγκρίνει δυο κρυπτογραφημένους ακέραιους a και b, αλλά το party B έχει το κλειδί αποκρυπτογράφησης. Η υλοποίηση γίνεται τροποποιώντας το Πρωτόκολλο του Veugen ώστε να χρησιμοποιεί το πρωτόκολλο σύγκρισης που περιγράφηκε παραπάνω (Γραμμή 2). Το B θα πρέπει να στείλει το κρυπτογραφημένο αποτέλεσμα στο A για να το αποκρυπτογραφήσει, ώστε το A λαμβάνει το plaintext εξόδου σωστά (Γραμμή 3).

4.1.3 Reversed comparison over encrypted data

Σε μερικές περιπτώσεις, είναι θεμιτό το αποτέλεσμα της σύγκρισης να παραμένει στο party που δεν έχει τα κρυπτογραφημένα δεδομένα. Για αυτό το λόγο τροποποιείται το πρωτόκολλο του Veugen ώστε να αντιστρέψει τις εξόδους του party A και party B. Αυτό επιτυγχάνεται από την ανταλλαγή των ρόλων των A και B στα τελευταία βήματα του πρωτοκόλλου μετά την σύγκριση με τις μη κρυπτογραφημένες είσόδους. Αυτό οδηγεί σε ένα πρωτόκολλο το οποίο φαίνεται στην Γραμμή 4 του πίνακα. Η Γραμμή 5 προκύπτει εάν ο A στείλει το κρυπτογραφημένο αποτέλεσμα στο B που θα το αποκρυπτογραφήσει.

4.1.4 Negative integers comparison and sign determination

Οι αρνητικοί ακέραιοι αντιμετωπίζονται από τα πρωτόκολλα χωρίς κάποια αλλαγή. Αν και το μέγεθος του Paillier plaintext είναι θετικό, ένας αρνητικός αριθμός απλώς γίνεται ένας μεγάλος αριθμός στον χώρο του plaintext λόγω της cyclicity του χώρου. Εφόσον οι τιμές που κρυπτογραφούνται είναι εντός ενός preset interval $(-2^l, 2^l)$, για κάποιο fixed l, το πρωτόκολλο του Veugen και τα παραπάνω πρωτόκολλα δουλεύουν σωστά. Σε μερικές περιπτώσεις, απαιτείται ο υπολογισμός του προσήμου ενός κρυπτογραφημένου ακεραίου $[\![b]\!]$, οπότε γίνεται σύγκριση με το 0.

4.2 argmax over encrypted data

Σε αυτό το σενάριο, το party A έχει k τιμές $a_1, \dots, a_k$ κρυπτογραφημένες με το μυστικό κλειδί του party B και το party B θέλει να μάθει το argmax πάνω σε αυτές τις τιμές (το index της μεγαλύτερης τιμής) αλλά κανένα party δεν πρέπει να μάθει τίποτα άλλο. Για παράδειγμα, αν A έχει τιμές $[\![1]\!]$, $[\![100]\!]$ και $[\![2]\!]$, το B πρέπει να μάθει ότι το δεύτερο έχει την μεγαλύτερη τιμή, αλλά τίποτα άλλο. Συγκεκριμένα, το B δεν θα πρέπει να ξέρει την σειρά των a_i. Το πρωτόκολλο 1 φαίνεται στην συνέχεια. Παρακάτω θα δοθεί μια διαίσθηση για το πρωτόκολλο και την ασφάλειά του.

Intuition: Έστω ένας strawman. Για να μην μάθει ο B την σειρά των k τιμών $\{a_i\}_{i=1}^k$, ο A εφαρμόζει μια τυχαία μετάθεση π. Το i-οστό στοιχείο γίνεται $[\![a'_i]\!] = [\![a_{\pi(i)}]\!]$ αντί για $[\![a_i]\!]$. Ύστερα, τα A και B συγκρίνουν τις πρώτες δυο τιμές $[\![a'_1]\!]$ και $[\![a'_2]\!]$ χρησιμοποιώντας το πρωτόκολλο σύγκρισης από την Γραμμή 4. Το B μαθαίνει το index, m, της μεγαλύτερης τιμής και λέει στο A να συγκρίνει τα $[\![a'_m]\!]$, $[\![a'_3]\!]$. Μετά από την επανάληψη της διαδικασίας αυτής για όλες τις k τιμές, ο B βρίσκει το index m της μεγαλύτερης τιμής. Ο A μπορεί τότε να υπολογίσει το $\pi^{-1}(m)$ το οποίο αποτελεί το argmax της αρχικής σειράς. Αφού το A εφάρμοσε μια τυχαία μετάθεση π, το B δεν μαθαίνει την σειρά των τιμών.

Το πρόβλημα είναι όμως ότι το A μαθαίνει την σειρά επειδή σε κάθε επανάληψη, ο A ξέρει την τιμή του m έως αυτό το βήμα και το π. Ένας τρόπος να διορθωθεί αυτό είναι το B να συγκρίνει κάθε ζευγάρι εισόδου του A αλλά αυτό θα οδηγούσε σε quadratic αριθμό συγκρίσεων, το οποίο είναι αργό. Ωστόσο, το πρωτόκολλο διατηρεί το γραμμικό αριθμό συγκρίσεων με τον εξής τρόπο. Σε κάθε σύγκριση, μόλις το B καθορίσει ποια είναι η μέγιστη τιμή εκ των δυο, μετά πρέπει να τυχαιοποιήσει (randomize) την κρυπτογράφηση αυτής της τιμής έτσι ώστε ο A να μην μπορεί να βρει ποια ήταν. Για αυτό ο B χρησιμοποιεί την Refresh διαδικασία για την τυχαιοποίηση των Paillier ciphertexts. Η περίπτωση όπου ο "refresher" γνωρίζει το μυστικό κλειδί, μπορεί να θεωρηθεί ως αποκρυπτογράφηση που ακολουθείται από επανακρυπτογράφηση αλλιώς μπορεί να θεωρηθεί ως πολλαπλασιασμός με κρυπτογράφηση του 0. Μια δυσκολία είναι ότι για την τυχαιοποίηση της κρυπτογράφησης της μέγιστης $[\![a'_m]\!]$, ο B χρειάζεται να πάρει αυτή την κρυπτογράφηση. Όμως, ο B δεν πρέπει να λάβει την κρυπτογράφηση διότι ο B έχει το κλειδί SK_P για την αποκρυπτογράφηση, γεγονός που παραβιάζει την ιδιωτικότητα. Αντιθέτως, η ιδέα είναι ο A να προσθέσει θόρυβο r_i και s_i στο $[\![a'_m]\!]$ ώστε η αποκρυπτογράφηση από το B να παράξει τυχαίες τιμές, κι έτσι το B κάνει refresh το ciphertext και το A αφαιρεί την τυχαιότητα r_i και s_i που είχε προσθέσει. Στο τέλος, το πρωτόκολλο θα κάνει k – 1 συγκρίσεις για l bits και 7(k – 1) ομομορφικές πράξεις. Σε round trips, προστίθενται κ – 1 roundtrips στο πρωτόκολλο σύγκρισης και 1 roundtrip σε κάθε loop.

Proposition: Το πρωτόκολλο 1 είναι σωστό και ασφαλές σε ένα honest-but-curious μοντέλο.

Proof intuition: Η ορθότητα είναι προφανής. Όσον αφορά την ασφάλεια, ο A δεν μαθαίνει ενδιάμεσα αποτελέσματα λόγω της ασφάλειας του πρωτόκολλου σύγκρισης και λόγω του γεγονότος ότι λαμβάνει ένα refreshed cyphertext από το B, το οποίο ο A δεν μπορεί να συνδυάσει με κάποιο προηγούμενο. Ο B μαθαίνει το αποτέλεσμα κάθε σύγκρισης όμως, επειδή ο A έχει εφαρμόσει τυχαία μετάθεση πριν την σύγκριση, ο B δεν μαθαίνει κάποια χρήσιμη πληροφορία.

Protocol 1 argmax over encrypted data

Input A: k encrypted integers $([\![a_1]\!], \ldots, [\![a_k]\!])$, the bit length l of the a_i, and public keys PK_{QR} and PK_P
Input B: Secret keys SK_P and SK_{QR}, the bit length l
Output A: $\text{argmax}_i\, a_i$

1: A: chooses a random permutation π over $\{1, \ldots, k\}$
2: A: $[\![\mathsf{max}]\!] \leftarrow [\![a_{\pi(1)}]\!]$
3: B: $m \leftarrow 1$
4: **for** $i = 2$ **to** k **do**
5: Using the comparison protocol (Sec. 4.1.3), B gets the bit $b_i = (\mathsf{max} \leq a_{\pi(i)})$
6: A picks two random integers $r_i, s_i \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$
7: A: $[\![m'_i]\!] \leftarrow [\![\mathsf{max}]\!] \cdot [\![r_i]\!]$ ▷ $m'_i = \mathsf{max} + r_i$
8: A: $[\![a'_i]\!] \leftarrow [\![a_{\pi(i)}]\!] \cdot [\![s_i]\!]$ ▷ $a'_i = a_{\pi(i)} + s_i$
9: A sends $[\![m'_i]\!]$ and $[\![a'_i]\!]$ to B
10: **if** b_i is true **then**
11: B: $m \leftarrow i$
12: B: $[\![v_i]\!] \leftarrow \mathsf{Refresh}[\![a'_i]\!]$ ▷ $v_i = a'_i$
13: **else**
14: B: $[\![v_i]\!] \leftarrow \mathsf{Refresh}[\![m'_i]\!]$ ▷ $v_i = m'_i$
15: **end if**
16: B sends to A $[\![v_i]\!]$
17: B sends to A $[\![b_i]\!]$
18: A: $[\![\mathsf{max}]\!] \leftarrow [\![v_i]\!] \cdot (g^{-1} \cdot [\![b_i]\!])^{r_i} \cdot [\![b_i]\!]^{-s_i}$
19: ▷ $\mathsf{max} = v_i + (b_i - 1) \cdot r_i - b_i \cdot t_i$
20: **end for**
21: B sends m to A
22: A outputs $\pi^{-1}(m)$

4.3 Changing the encryption scheme

Για την ανάπτυξη των building blocks αναπτύχθηκε ένα πρωτόκολλο για μετατροπή των ciphertexts από ένα σύστημα κρυπτογράφησης σε ένα άλλο ενώ διατηρούνται τα βασικά plaintexts. Πρώτα, παρουσιάζεται ένα πρωτόκολλο που εναλλάσσεται μεταξύ δυο συστημάτων κρυπτογράφησης με το ίδιο μέγεθος plaintext (QR & FHE) και μετά παρουσιάζεται ένα διαφορετικό πρωτόκολλο για εναλλαγή από QR σε Paillier.

Έστω δυο προσθετικά ομομορφικά συστήματα κρυπτογράφησης E_1 και E_2, και τα δυο σημασιολογικά ασφαλή με το ίδιο plaintext χώρο M. Έστω $[\![\cdot]\!]_1$ μια κρυπτογράφηση με το E_1 και $[\![\cdot]\!]_2$ μια κρυπτογράφηση με το E_2. Έστω ότι το B έχει τα μυστικά κλειδιά SK_1 και SK_2 και για τα δυο συστήματα και το A έχει τα αντίστοιχα δημόσια κλειδιά PK_1 και PK_2. Επίσης, το A έχει μια τιμή κρυπτογραφημένη με PK_1, $[\![c]\!]_1$. Το πρωτόκολλο 2 ενεργοποιεί το A ώστε να αποκτήσει μια κρυπτογράφηση του c με το E_2, $[\![c]\!]_2$ χωρίς να αποκαλύψει τίποτα το B για το c.

Protocol intuition: Η ιδέα είναι ο A να προσθέσει ένα τυχαίο θόρυβο r στο ciphertext χρησιμοποιώντας την ομομορφική ιδιότητα του E_1. Έπειτα, ο B αποκρυπτογραφεί την τιμή αυτή με το E_1 και την κρυπτογραφεί με E_2 και στέλνει το αποτέλεσμα στο A το οποίο αφαιρεί την τυχαιότητα r χρησιμοποιώντας την ομομορφική ιδιότητα του E_2. Παρόλο που ο B μπορούσε να αποκρυπτογραφήσει $[\![c']\!]_1$, ο B αποκτά το x + r ∈ M, το οποίο κρύβει το x με ένα information - theoretic way (One Time Pad).

Σημειώνεται ότι, για κάποια συστήματα, ο χώρος M του plaintext βασίζεται στα μυστικά κλειδιά. Σε αυτή την περίπτωση, πρέπει ο A να μπορεί να επιλέξει ομοιόμορφα στοιχεία του M χωρίς να το ξέρει. Για παράδειγμα, για Paillier, $M = \mathbb{Z}_N^* \approx \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ όπου p, q είναι ιδιωτικοί πρώτοι αριθμοί. Όμως, τότε ο A μπορεί να δοκιμάσει θόρυβο στο $\mathbb{Z}_N$ που δεν θα ανήκει στο $\mathbb{Z}_N^*$ με αμελητέα πιθανότητα $\left(1-\frac{1}{p}\right)\left(1-\frac{1}{p}\right) \approx 1-\frac{2}{\sqrt{N}}$

Proposition: Το πρωτόκολλο 2 είναι ασφαλές σε ένα honest-but-curious μοντέλο.

Στους ταξινομητές χρησιμοποιείται αυτό το πρωτόκολλο για M = {0, 1} και τα συστήματα κρυπτογράφησης είναι QR (για E_1) και FHE συστήματα σε bites (για E_2). Σε ορισμένες περιπτώσεις, ίσως να γίνει εναλλαγή από QR σε Paillier. Σημειώνεται ότι μπορεί να προσομοιωθεί ο ομομορφικός τελεστής XOR και ο χώρος μηνυμάτων M = {0, 1} με Paillier: εύκολα υπολογίζεται η κρυπτογράφηση Paillier των $b_1 \oplus b_2$ όταν το πολύ ένα από τα b_i είναι κρυπτογραφημένο. Αυτή είναι η περίπτωση που ισχύει σε αυτό το paper επειδή το party A έχει τυχαιότητα r.

Protocol 2 Changing the encryption scheme

Input A: $[\![c]\!]_1$ and public keys PK_1 and PK_2
Input B: Secret keys SK_1 and SK_2
Output A: $[\![c]\!]_2$

1: A uniformly picks $r \leftarrow M$
2: A sends $[\![c']\!]_1 \leftarrow [\![c]\!]_1 \cdot [\![r]\!]_1$ to B
3: B decrypts c' and re-encrypts with E_2
4: B sends $[\![c']\!]_2$ to A
5: A: $[\![c]\!]_2 = [\![c']\!]_2 \cdot [\![r]\!]_2^{-1}$
6: A outputs $[\![c]\!]_2$

4.3.1 XOR with Paillier

Έστω ότι ένα party παίρνει ένα bit b_1 κρυπτογραφημένο με το σύστημα Paillier και ότι μόνο αυτό το party έχει το δημόσιο κλειδί. Αυτό το party γνωρίζει το bit b_2 και θέλει να υπολογίσει την κρυπτογράφηση του $[\![b_1 \oplus b_2]\!]$. Ισχύει ότι: $b_1 \oplus b_2 = \begin{cases} b_1, if\ b_2 = 0 \\ 1 - b_1, if\ b_2 = 1 \end{cases}$

Έτσι, είναι εύκολος ο υπολογισμός αν είναι γνωστό ότι το modulus N και η γεννήτρια g. Οπότε,

$$[\![b_1 \oplus b_2]\!] = \begin{cases} [\![b_1]\!], if\ b_2 = 0 \\ g[\![b_1]\!]^{-1}\ mod\ N^2, if\ b_2 = 1 \end{cases}$$

Αν θέλει κάποιος να αποκαλύψει το αποτέλεσμα σε έναν αντίπαλο που ξέρει την αρχική κρυπτογράφηση του b_1 αλλά όχι το μυστικό κλειδί, τότε θα πρέπει να γίνει refresh στο αποτέλεσμα της προηγούμενης συνάρτησης για να είναι σίγουρη η σημασιολογική ασφάλεια.

4.4 Computing dot products

Για πληρότητα, συμπεριλαμβάνεται ένας straightforward αλγόριθμος (Πρωτόκολλο 3) για τον υπολογισμό του dot product δυο διανυσμάτων, ο οποίος βασίζεται στην ομομορφική ιδιότητα του Paillier.

Protocol 3 Private dot product

Input A: $x = (x_1, \ldots, x_d) \in \mathbb{Z}^d$, public key PK_P
Input B: $y = (y_1, \ldots, y_d) \in \mathbb{Z}^d$, secret key SK_P
Output A: $[\![\langle x, y \rangle]\!]$

1: B encrypts $y_1, \ldots, y_d$ and sends the encryptions $[\![y_i]\!]$ to A
2: A computes $[\![v]\!] = \prod_i [\![y_i]\!]^{x_i} \mod N^2$ ▷ $v = \sum y_i x_i$
3: A re-randomizes and outputs $[\![v]\!]$

Proposition: Το πρωτόκολλο 3 είναι ασφαλές σε ένα honest-but-curious μοντέλο.

Re-randomization ονομάζεται η διαδικασία όπου, από μια κρυπτογράφηση Paillier, μπορεί κανείς, χωρίς γνώση του ιδιωτικό κλειδί, από μία κωδικοποίηση P να παράξει μια νέα πιθανοτική κωδικοποίηση P_2. Ακόμη, στο δεύτερο βήμα ισχύει η ομομορφική ιδιότητα παρόλο που ισχύει Partial Homomorphic Encryption αφού τα x_i αποτελούν το δημόσιο κλειδί.

4.5 Dealing with floating point numbers

Αν και όλα τα πρωτόκολλα χειρίζονται ακέραιους, οι ταξινομητές συνήθως χρησιμοποιούν floating point αριθμούς. Έτσι, όταν αναπτύσσεται ένας ταξινομητής με αυτά τα πρωτόκολλα, θα πρέπει να γίνει ανάλογη προσαρμογή. Ευτυχώς, οι περισσότερες λειτουργίες είναι είτε προσθέσεις είτε πολλαπλασιασμοί. Μια απλή λύση είναι ο πολλαπλασιασμός κάθε floating point τιμής με μια σταθερά K (π.χ. $K = 2^{52}$ για IEEE 754 doubles) κι έτσι, υποστηρίζεται πεπερασμένη ακρίβεια. Επίσης, θα πρέπει να ληφθεί υπόψιν το μήκος των bit για τις συγκρίσεις.

5. Private hyperplane decision

Αυτός ο ταξινομητής υπολογίζει το εξής:

$$k^* = \underset{i\in[k]}{argmax}\langle w_i, x\rangle$$

Τώρα που έχει δημιουργηθεί η βιβλιοθήκη για building blocks, είναι straightforward η υλοποίηση του ταξινομητή (Πρωτόκολλο 4) με ασφάλεια: ο χρήστης υπολογίζει την κρυπτογράφηση $[\![\langle w_i, x\rangle]\!]$ για όλα τα $i \in [k]$ με χρήση του dot product και έτσι εφαρμόζει το argmax (Πρωτόκολλο 1) για το κρυπτογραφημένο dot product.

Protocol 4 Private hyperplane decision

Client's (C) Input: $x = (x_1, \ldots, x_d) \in \mathbb{Z}^d$, public keys PK_P and PK_{QR}
Server's (S) Input: $\{w_i\}_{i=1}^k$ where $\forall i \in [k], w_i \in \mathbb{Z}^n$, secret keys SK_P and SK_{QR}
Client's Output: $\underset{i\in[k]}{\text{argmax}}\langle w_i, x\rangle$

1: **for** $i = 1$ **to** k **do**
2: C and S run Protocol 3 for private dot product where C is party A with input x and S is party B with input w_i.
3: C gets $[\![v_i]\!]$ the result of the protocol.
▷ $v_i \leftarrow \langle x, w_i\rangle$
4: **end for**
5: C and S run Protocol 1 for argmax where C is the A, and S the B, and $[\![v_1]\!], \ldots, [\![v_k]\!]$ the input ciphertexts. C gets the result i_0 of the protocol.
▷ $i_0 \leftarrow \underset{i\in[k]}{\text{argmax}}\, v_i$
6: C outputs i_0

Proposition: Το πρωτόκολλο 4 είναι ασφαλές σε ένα honest-but-curious μοντέλο.

6. Secure Naive Bayes Classifier

Αυτός ο ταξινομητής υπολογίζει το εξής:

$$k^* = \underset{i\in[k]}{\text{argmax}}\, p(C = c_i | X = x) =$$

$$= \underset{i\in[k]}{\text{argmax}} \left\{ \log p(C = c_i) + \sum_{j=1}^{d} \log p\big(X_j = x_j | C = c_i\big) \right\}$$

Στόχος είναι ο χρήστης να μάθει το k^* χωρίς να μάθει τίποτα για τις πιθανότητες που συνθέτουν το μοντέλο και ο server πρέπει να μην μάθει τίποτα για το x.

6.1 Preparing the model

Αφού το σύστημα κρυπτογράφησης Paillier δουλεύει για ακέραιους, κάθε log μετατρέπεται σε έναν ακέραιο πολλαπλασιάζοντάς τον με ένα μεγάλο αριθμό K ώστε να διατηρείται μεγάλη ακρίβεια. Εφόσον οι μόνες πράξεις που χρησιμοποιούνται για την ταξινόμηση είναι προσθέσεις και συγκρίσεις, πολλαπλασιάζοντας την $p(x_j|c_i)$ με K προκύπτουν παντού ακέραιοι και δεν επηρεάζεται το αποτέλεσμα. Για παράδειγμα, αν μπορεί να γίνει ο υπολογισμός χρησιμοποιώντας τους IEEE 754 διπλής ακρίβειας floating point αριθμούς με 52 bits ακρίβειας, τότε κάθε πιθανότητα μπορεί να αναπαρασταθεί ως: $p = m \cdot 2^e$, όπου m σε δυαδική αναπαράσταση είναι $(m)_2 = 1.d$ και d είναι ένα ακέραιος 52 bits. Έτσι, για $1 \leq m < 2$, το m ξαναγράφεται ως:

$$m = \frac{m'}{2^{52}} \; with \; m' \in \mathbb{N} \cap [2^{52}, 2^{53})$$

Με αυτή την αναπαράσταση υπολογίζεται η σταθερά K έτσι ώστε $K \cdot v_i \in \mathbb{N}$, για όλα τα i. Όπως και πριν, μπορούν να γραφτούν τα v_i ως:

$$v_i = m_i \cdot 2^{e_i - 52}$$

Έστω $e^* = \min_i e_i$ και $\delta_i = e_i - e^* \geq 0$. Τότε,

$$v_i = m_i' \cdot 2^{\delta_i} \cdot 2^{e^* - 52}$$

Και για $K = 2^{52 - e^*}$ θα ισχύει $K \cdot v_i = m_i' \cdot 2^{\delta_i} \in \mathbb{N}$.

Σημειώνεται ότι τα v_i μπορεί να είναι μεγάλοι ακέραιοι και ίσως να προκληθούν overflow errors. Μάλιστα, όλα αυτά γίνονται για την αποθήκευση λογαρίθμων από πιθανότητες σε Paillier ciphertexts, όπου ο Paillier plaintext χώρος είναι πολύ μεγάλος και τα δ_i παραμένουν μικρά. Επίσης, αυτή η διαδικασία του shifting γίνεται χωρίς απώλεια ακρίβειας καθώς μπορεί να εργαστεί κανείς απευθείας με την bit αναπαράσταση των floating point αριθμών.

Τέλος, θα πρέπει να εξεταστεί ότι δεν γίνεται overflow στο χώρο μηνυμάτων του Paillier όταν εκτελούνται οι πράξεις. Αν το d είναι ο αριθμός των χαρακτηριστικών, ο μέγιστος αριθμός από bits όταν γίνονται οι υπολογισμοί θα είναι $l_{max} = d + 1 + (52 - \delta^*)$ όπου $\delta^* = \max \delta_i$. Οπότε, θα πρέπει να προστεθούν οι πιθανότητες για τα d χαρακτηριστικά και η πιθανότητα της ετικέτας της κλάσης (όρος d + 1) και κάθε πιθανότητα είναι κωδικοποιημένη χρησιμοποιώντας $(52 + \delta^*)$ bits. Έτσι, πρέπει να ισχύει ότι $\log_2 N \geq 1024$ και συνήθως ισχύει $\lambda \approx 100$.

Έστω D_j το domain με τις πιθανές τιμές των x_j. Ο server προετοιμάζει k·d + 1 πίνακες ως μέρος του μοντέλου, όπου το K υπολογίζεται όπως περιγράφηκε παραπάνω:

- Ένας πίνακας για τα priors στις κλάσεις P: $P(i) = \lceil K \log p(C = c_i) \rceil$
- Ένας πίνακας για κάθε χαρακτηριστικό j και για κάθε κλάση i, $T_{i,j}$: $T_{i,j}(v) \approx \lceil K \log p(X_j = v | C = c_i) \rceil$, για όλα τα $v \in D_j$

Ο πίνακας παραμένει μικρός καθώς P έχει μια είσοδο ανά κατηγορία, οπότε k εισόδους συνολικά και T έχει μια είσοδο ανά κατηγορία και χαρακτηριστικό οπότε k·D εισόδους όπου $D = \sum |D_j|$. Στα παραδείγματα που εξετάζονται, αυτό αποτελεί περίπου 3600 εισόδους. Επιπλέον, αυτό το βήμα προετοιμασίας μπορεί να γίνει μια φορά και για όλα κατά το startup του server κι έτσι, είναι amortized.

6.2 Protocol

Intuition: Ο server κρυπτογραφεί κάθε είσοδο σε πίνακες με Paillier και δίνει την κρυπτογράφηση στον χρήστη. Για κάθε κλάση c_i, ο χρήστης χρησιμοποιεί τον προσθετικό ομομορφισμό του Paillier για να υπολογίσει το $[\![p_i]\!] = [\![P(i)]\!] \prod_{j=1}^{d} [\![T_{i,j}(x_j)]\!]$. Τέλος, ο χρήστης τρέχει το argmax πρωτόκολλο (Πρωτόκολλο 1) για να πάρει το $argmax\ p_i$. Για πληρότητα, παρακάτω φαίνεται το Πρωτόκολλο 5.

Protocol 5 Naïve Bayes Classifier

Client's (C) Input: $x = (x_1, \ldots, x_d) \in \mathbb{Z}^d$, public key PK_P, secret key SK_{QR}
Server's (S) Input: The secret key SK_P, public key PK_{QR} and probability tables $\{\log p(C = c_i)\}_{1 \le i \le k}$ and $\left\{\{\log p(X_j = v | C = c_i)\}_{v \in D_j}\right\}_{1 \le j \le d, 1 \le i \le k}$
Client's Output: i_0 such that $p(x, c_{i_0})$ is maximum

1: The server prepares the tables P and $\{T_{i,j}\}_{1 \le i \le k, 1 \le j \le d}$ and encrypts their entries using Paillier.
2: The server sends $[\![P]\!]$ and $\{[\![T_{i,j}]\!]\}_{i,j}$ to the client.
3: For all $1 \le i \le k$, the client computes $[\![p_i]\!] = [\![P(i)]\!] \prod_{j=1}^{d} [\![T_{i,j}(x_j)]\!]$.
4: The client runs the argmax protocol (Protocol 1) with the server and gets $i_0 = \text{argmax}_i\, p_i$
5: C outputs i_0

Proposition: Το πρωτόκολλο 5 είναι ασφαλές σε ένα honest-but-curious μοντέλο.

Proof intuition: Δεδομένης της ιδιότητας ασφάλειας του argmax πρωτόκολλου, και της σημασιολογικής ασφάλειας του Paillier κρυπτοσυστήματος, η ασφάλεια αυτού του ταξινομητή είναι trivial με την χρήση ενός modular composition θεωρήματος.

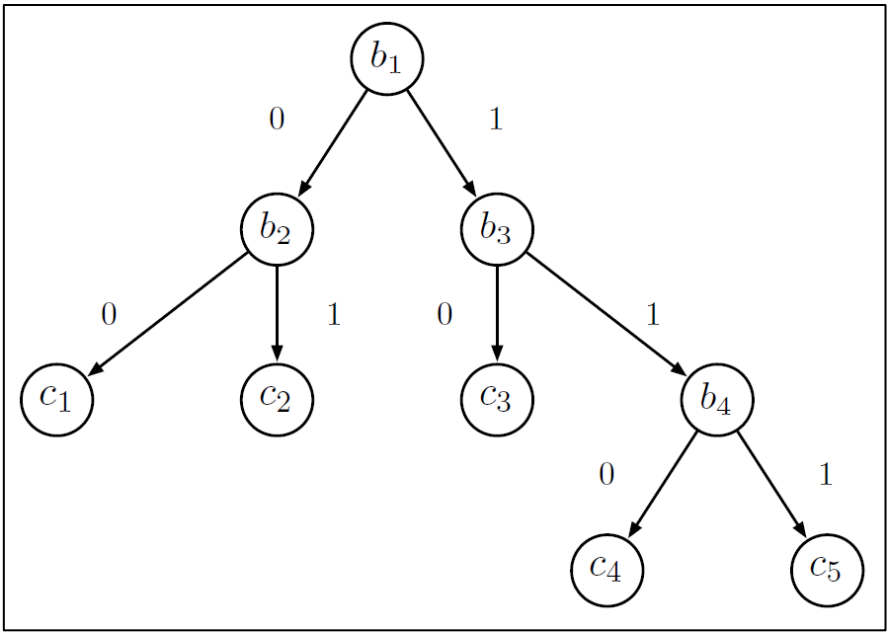
Efficiency: Οι πίνακες P και $\{T_{i,j}\}_{1 \le i \le k, 1 \le j \le d}$ μπορούν να είναι ήδη υπολογισμένοι από πριν. Έτσι, το κόστος κατασκευής των πινάκων μπορεί να είναι amortized μετά από πολλές χρήσεις. Για τον υπολογισμό των κρυπτογραφημένων πιθανοτήτων p_i, ο χρήστης τρέχει d ομομορφικές πράξεις για κάθε i, οπότε κάνει k·d modular πολλαπλασιασμούς. Τότε, τα parties τρέχουν ένα argmax πρωτόκολλο οπότε γίνονται k – 1 συγκρίσεις και O(k) ομομορφικές πράξεις. Συνεπώς, σε σύγκριση με τον μη κρυπτογραφημένο υπολογισμό, το overhead προκύπτει μόνο από την κρυπτογράφηση με ομομορφικές πράξεις αντί για τις πράξεις στο plaintext. Όσον αφορά τα round trips, οφείλονται στο argmax πρωτόκολλο και θα είναι k – 1 τρεξίματα για συγκρίσεις και k – 1 προσθετικά roundtrips.

7. Private decision trees

Ένας private decision tree ταξινομητής επιτρέπει στον server να διασχίσει το decision tree χρησιμοποιώντας την είσοδο x του χρήστη έτσι ώστε ο server να μην μάθει το x και ο χρήστης να μην μάθει την δομή του δέντρου και τα thresholds σε κάθε κόμβο. Συγκεκριμένα, η πρόκληση είναι ότι ο χρήστης δεν πρέπει να μάθει το μονοπάτι του δέντρου που αντιστοιχεί στο x, δηλαδή την θέση του μονοπατιού στο δέντρο και το μήκος του θα διαρρεύσουν πληροφορίες για το μοντέλο. Το αποτέλεσμα της ταξινόμησης δεν είναι σίγουρο ότι θα διαρρεύσει πληροφορίες για το μονοπάτι του δέντρου. Η ιδέα είναι να αποδοθεί το decision tree ως ένα πολυώνυμο P του οποίου η έξοδος θα είναι το αποτέλεσμα της ταξινόμησης. Τότε, ο server και ο χρήστης θα υπολογίσουν ιδιωτικά τις εισόδους με βάση το x και τα thresholds w_i. Τέλος, ο server θα εκτιμήσει το πολυώνυμο P ιδιωτικά.

7.1 Polynomial form of a decision tree

Έστω ότι κάθε κόμβος του δέντρου είναι μια boolean μεταβλητή που σχετίζεται με αυτόν. Η τιμή της θα είναι ίση με 1 για τον κόμβο αν, για είσοδο x, ακολουθήσει κανείς το δεξί κλαδί, αλλιώς 0. Για παράδειγμα, έστω ότι η μεταβλητή στην ρίζα του δέντρου είναι b_1. Το b_1 θα είναι 1 αν $x_1 \leq w_1$ αλλιώς 0. Έτσι, μπορεί να προκύψει ένα πολυώνυμο P το οποίο, για είσοδο όλες αυτές τις μεταβλητές και την τιμή της κάθε κλάσης σε ένα φύλλο, δίνει έξοδο την κλάση που προβλέφθηκε για το x. Η ιδέα είναι ότι το P είναι ένα άθροισμα από όρους, όπου κάθε όρος (έστω t) αντιστοιχεί σε ένα μονοπάτι του δέντρου από την ρίζα προς ένα φύλλο (έστω c). Ένας όρος t κάνει εκτίμηση c, αν και μόνο αν, το x είναι ταξινομημένα μέσα στο μονοπάτι T, αλλιώς κάνει εκτίμηση ίση με 0. Έτσι, ο όρος που αντιστοιχεί σε μονοπάτι του δέντρου είναι ουσιαστικά ο πολλαπλασιασμός των boolean μεταβλητών πάνω στο μονοπάτι και η κλάση θα είναι το φύλλο. Για παράδειγμα, για το δέντρο, το P θα είναι $P(b_1, \dots, b_4, c_1, \dots, c_5) = b_1(b_3(b_4c_5 + (1-b_4)c_4) + (1-b_3)c_3) + (1-b_1)(b_2c_2 + (1-b_2)c_1)$:



Έστω τώρα ότι με $\mathcal{F}$ αναπαρίσταται μια αναδρομική διαδικασία για την κατασκευή του P δεδομένου ενός δυαδικού decision tree T:

- Αν το T αποτελείται μόνο από τον κόμβο φύλλο, με index κατηγορίας c_i, τότε $\mathcal{F}(T) = c_i$.
- Αν το T είναι άδειο, τότε $\mathcal{F}(T) = 0$.
- Αλλιώς, αν το T έχει έναν εσωτερικό κόμβο χρησιμοποιώντας την boolean b και T_0 και T_1 είναι το αριστερό και το δεξί υποδέντρο, τότε $\mathcal{F}(T) = b\mathcal{F}(T_1) + (1-b)\mathcal{F}(T_0)$.

7.2 Private evaluation of a polynomial

Αρχικά, θα γίνει αναφορά για τον ασφαλή υπολογισμό των τιμών των boolean μεταβλητών. Έστω n ο αριθμός των κόμβων στο δέντρο και nleaves ο αριθμός των φύλλων στο δέντρο. Αυτές οι τιμές πρέπει να παραμείνουν άγνωστες στον server επειδή διαρρέουν πληροφορία για το x, αφού είναι αποτέλεσμα των ενδιάμεσων υπολογισμών για το κριτήριο ταξινόμησης. Για κάθε boolean μεταβλητή b_i, ο server και ο χρήστης συμμετέχουν στο πρωτόκολλο σύγκρισης για να συγκρίνουν τα w_i και τα αντίστοιχα χαρακτηριστικά (attribute) του x. Συνεπώς, ο server αποκτά $[b_i]$ για $i \in 1, \dots, n$ και μετά αλλάζει την κρυπτογράφηση αυτών των τιμών σε FHE χρησιμοποιώντας το Πρωτόκολλο 2 ώστε να αποκτήσει το $[\![[b_i]]\!]$.

Ο server κάνει εκτίμηση για το P πάνω σε $([\![[b_1]]\!], \dots, [\![[b_n]]\!])$ χρησιμοποιώντας τις ομομορφικές ιδιότητες του FHE. Στις περισσότερες περιπτώσεις, η εκτίμηση του FHE είναι πολύ αργή, αλλά επιτυγχάνονται καλύτερα αποτελέσματα μέσω του συνδυασμού τεχνικών που θα αναλυθούν στην συνέχεια. Για την κατανόηση αυτών των τεχνικών υπενθυμίζεται ότι μια τυπική FHE εκτίμηση υπολογίζεται πάνω σε ένα κύκλωμα του οποίου οι πύλες είναι modular πρόσθεση και πολλαπλασιασμός. Η επίδοση του FHE βασίζεται πολύ στο βάθος των πολλαπλασιασμών στο κύκλωμα. Πρώτα, χρησιμοποιείται leveled FHE σύστημα, δηλαδή ένα σύστημα που υποστηρίζει μόνο ένα a priori fixed multiplicative βάθος αντί για ένα αυθαίρετο βάθος. Όσο αυτό το βάθος είναι μικρό, το σύστημα είναι ταχύτερο από ένα full FHE σύστημα. Έπειτα, πρέπει το multiplicative βάθος να είναι πολύ μικρό χρησιμοποιώντας μια tree-based εκτίμηση. Αν h_{max} είναι το μέγιστο ύψος του decision tree, τότε το P έχει έναν όρο $a_1 \cdot \ldots \cdot a_{h_{max}}$. Αν εκτιμηθεί αυτός ο όρος naively με FHE, τότε αυτές οι τιμές πολλαπλασιάζονται σειριακά. Αυτό προκαλεί ένα multiplicative βάθος h_{max}, το οποίο καθιστά το FHE αργό για κοινές h_{max} τιμές. Αντιθέτως, κατασκευάζεται ένα δυαδικό δέντρο πάνω σε αυτές τις τιμές, οι οποίες πολλαπλασιάζονται σε ζεύγη με βάση την δομή του δέντρου. Αυτό οδηγεί σε multiplicative βάθος $\log h_{max}$, γεγονός που κάνει την εκτίμηση του FHE σημαντικά ταχύτερη. Τέλος, χρησιμοποιείται $\mathbb{F}_2$ ως χώρος για το plaintext και SIMD slots για παραλληλοποίηση. Τα FHE συστήματα είναι πολύ γρήγορα όταν οι τιμές είναι κρυπτογραφημένες σε bits. Όμως, το P περιέχει κλάσεις οι οποίες είναι συνήθως παραπάνω από ένα bit σε μήκος. Για να γίνει ο υπολογισμός του P πάνω στο $\mathbb{F}_2$ θα πρέπει κάθε κλάση να αναπαρασταθεί σε δυαδική μορφή. Έστω $l = \lceil \log_2 k \rceil$ (k είναι ένας αριθμός από κλάσεις) ο αριθμός από τα bits που απαιτούνται για την αναπαράσταση της κλάσης. Τότε, το P θα εκτιμηθεί l φορές, μια για κάθε ένα από τα l bits της κλάσης. Έτσι, η j-οστή εκτίμηση του P παίρνει ως είσοδο $b_1, \dots, b_n$ και για κάθε φύλλο c_i, το j-οστό bit c_{ij}. Το αποτέλεσμα θα είναι $P(b_1, \dots, b_n, c_{1j}, c_{2j}, \dots, c_{nleavesj})$, και αναπαριστά το j-οστό bit της κλάσης. Συνεπώς, απαιτείται να τρέξει η FHE εκτίμηση l φορές. Για την αποφυγή του παράγοντα l, η ιδέα είναι να χρησιμοποιηθούν τα SIMD slots. Αυτά, επιτρέπουν την κρυπτογράφηση πολλαπλών bits σε ένα ciphertext έτσι ώστε κάθε πράξη που εφαρμόζεται σε αυτό να εφαρμόζεται παράλληλα σε κάθε bit. Έτσι, για κάθε κλάση c_j, ο server δημιουργεί ένα FHE ciphertext $[\![[c_{j0}, \dots, c_{jl-1}]]\!]$. Για κάθε κόμβο b_i, δημιουργείται ένα ciphertext $[\![[b_1, \dots, b_i]]\!]$ απλώς από την επανάληψη της b_i τιμής σε κάθε slot. Τότε, ο server τρέχει μια FHE εκτίμηση στο P πάνω σε όλα αυτά τα ciphertexts και προκύπτουν $[\![[c_{o0}, \dots, c_{ol-1}]]\!]$, όπου c_o είναι η κλάση που προέκυψε ως αποτέλεσμα. Έτσι, αντί για l FHE εκτιμήσεις, ο server τρέχει μια εκτίμηση μόνο μια φορά. Αυτό βελτιώνει την επίδοση κατά $\log k$. Η εφαρμογή του παραλληλισμού με SIMD slots ήταν εφικτή χάρη στο γεγονός ότι το ίδιο πολυώνυμο P έπρεπε να υπολογιστεί για κάθε slot. Τέλος, η εκτίμηση του decision tree γίνεται χρησιμοποιώντας 2n FHE πολλαπλασιασμούς και 2n FHE προσθέσεις, όπου n είναι ο αριθμός των κριτηρίων. Η εκτίμηση του κυκλώματος έχει multiplicative βάθος $\lceil \log_2 n + 1 \rceil$.

7.3 Formal description

Το Πρωτόκολλο 6 περιγράφει το resulting πρωτόκολλο.

Protocol 6 Decision Tree Classifier

Client's (C) Input: $x = (x_1, \ldots, x_n) \in \mathbb{Z}^n$, secret keys $\mathsf{SK}_{QR}, \mathsf{SK}_{FHE}$

Server's (S) Input: The public keys $\mathsf{PK}_{QR}, \mathsf{PK}_{FHE}$, the model as a decision tree, including the n thresholds $\{w_i\}_{i=1}^n$.

Client's Output: The value of the leaf of the decision tree associated with the inputs $b_1, \ldots, b_n$.

1: S produces an n-variate polynomial P as described in section 7.1.
2: S and C interact in the comparison protocol, so that S obtains $[b_i]$ for $i \in [1 \ldots n]$ by comparing w_i to the corresponding attribute of x.
3: Using Protocol 2, S changes the encryption from QR to FHE and obtains $[\![b_1]\!], \ldots, [\![b_n]\!]$.
4: To evaluate P, S encrypts the bits of each category c_i using FHE and SIMD slots, obtaining $[\![c_{i1}, \ldots, c_{il}]\!]$. S uses SIMD slots to compute homomorphically $[\![P(b_1, \ldots, b_n, c_{10}, \ldots, c_{\mathsf{nleaves}0}), \ \ldots, \ P(b_1, \ldots, b_n, c_{1l-1}, \ldots, c_{\mathsf{nleaves}l-1})]\!]$. It rerandomizes the resulting ciphertext using FHE's rerandomization function, and sends the result to the client.
5: C decrypts the result as the bit vector $(v_0, \ldots, v_{l-1})$ and outputs $\sum_{i=0}^{l-1} v_i \cdot 2^i$.

Proposition: Το πρωτόκολλο 6 είναι ασφαλές σε ένα honest-but-curious μοντέλο.

Proof intuition: Κατά το πρωτόκολλο σύγκρισης, ο server μαθαίνει μόνο κρυπτογραφημένα bits, οπότε δεν μαθαίνει τίποτα για το x. Κατά την FHE εκτίμηση, παρομοίως, δεν μαθαίνει τίποτα για την είσοδο λόγω της ασφάλειας του FHE. Ο χρήστης δεν μαθαίνει τίποτα για την δομή του δέντρου επειδή ο server πραγματοποιεί την εκτίμηση του πολυωνύμου. Ομοίως, ο χρήστης δεν μαθαίνει τα bits στους κόμβους του δέντρου λόγω της ασφάλειας του πρωτοκόλλου σύγκρισης. Οι αλληλεπιδράσεις μεταξύ server και χρήστη οφείλονται στις συγκρίσεις σχεδόν εξ ολοκλήρου, καθώς η εκτίμηση του decision tree δεν απαιτεί καμία αλληλεπίδραση και απλώς στέλνει το κρυπτογραφημένο αποτέλεσμα.

8. Combining classifiers with AdaBoost

Το AdaBoost είναι μια τεχνική και η βασική ιδέα είναι ότι συνδυάζει ένα σύνολο από αδύναμους ταξινομητές $h_i(x)$: $\mathbb{R}^d \longmapsto \{-1, +1\}$ και διατηρεί τον καλύτερο εξ αυτών. Ο αλγόριθμος επιλέγει t scalars $\{a_i\}_{i=1}^t$ και δημιουργεί ένα δυνατό ταξινομητή:

$$H(x) = sign\left(\sum_{i=1}^{t} a_i h_i(x)\right)$$

Αν κάθε ένα από τα $h_i(\cdot)$ είναι ένα instance ενός ταξινομητή που υποστηρίζεται από τα παραπάνω πρωτόκολλα, τότε για δοσμένα scalars a_i, εύκολα και με ασφάλεια μπορεί να εκτιμηθεί η $H(x)$ απλώς με την σύνθεση των building blocks. Πρώτα, εκτελείται ένα πρωτόκολλο ασφαλείας για κάθε h_i, ο server διατηρεί το ενδιάμεσο αποτέλεσμα, και το αποτέλεσμα του $h_i(x)$ κρυπτογραφείται χρησιμοποιώντας ένα από τα πρωτόκολλα σύγκρισης. Ύστερα, αν χρειάζεται, γίνεται μετατροπή στο σύστημα κρυπτογράφησης του Paillier με το Πρωτόκολλο 2 και συνδυάζονται αυτά τα ενδιάμεσα αποτελέσματα χρησιμοποιώντας την προσθετική ομομορφική ιδιότητα του Paillier όπως στο dot product πρωτόκολλο (Πρωτόκολλο 3). Τέλος, εκτελείται σύγκριση πάνω στα κρυπτογραφημένα δεδομένα συγκρίνοντας το αποτέλεσμα με το μηδέν ώστε ο χρήστης να λάβει το τελικό αποτέλεσμα.

9. Implementation

Η υλοποίηση των πρωτοκόλλων και των ταξινομητών έγινε σε C++ με GMP, Boost, Google Protocol Buffers, και HELib για την υλοποίηση του FHE. Ο κώδικας είναι γραμμένος με modular τρόπο, δηλαδή όλα τα κύρια πρωτόκολλα μπορούν να χρησιμοποιηθούν ως black boxes. Έτσι, η συγγραφή ασφαλών ταξινομητών καταλήγει στην χρήση των σωστών API. Για παράδειγμα, για τον γραμμικό ταξινομητή, ο χρήστης απλώς καλεί ένα Πρωτόκολλο ανταλλαγής κλειδιών για το setup των διάφορων κλειδιών, μετά με το dot product πρωτόκολλο, και τέλος το Πρωτόκολλο σύγκρισης πάνω σε κρυπτογραφημένα δεδομένα ώστε να προκύψει η ζητούμενη έξοδο.

Ένας γραμμικός ταξινομητής:

```
bool Linear_Classifier_Client::run()
{
  exchange_keys();

  // values_ is a vector of integers
  // compute the dot product
  mpz_class v = compute_dot_product(values_);
  mpz_class w = 1; // encryption of 0

  // compare the dot product with 0
  return enc_comparison(v, w, bit_size_, false);
}
```

```
void Linear_Classifier_Server_session::
        run_session()
{
  exchange_keys();

  // enc_model_ is the encrypted model vector
  // compute the dot product
  help_compute_dot_product(enc_model_, true);

  // help the client to get
  // the sign of the dot product
  help_enc_comparison(bit_size_, false);
}
```

10. Evaluation

Η εκτίμηση έγινε με βάση τα εξής: (i) κατά πόσο τα building blocks μπορούν να χρησιμοποιηθούν για την κατασκευή άλλων ταξινομητών με modular τρόπο, (ii) ποιο είναι το overhead της επίδοσης των building blocks και (iii) ποιο είναι το overhead της επίδοσης των ταξινομητών.

10.1 Using our building blocks library

Εδώ εξετάζεται η βιβλιοθήκη που αναπτύχθηκε και δείχνεται ότι μπορεί να χρησιμοποιηθεί για την δημιουργία και άλλων ταξινομητών με modular τρόπο και ότι είναι μια χρήσιμη προσθήκη και από μόνη της. Θα αναλυθούν και θα δημιουργούν ένας πολυπλέκτης (multiplexer) και ένας face detector. Σημειώνεται ότι ήδη υπάρχει ένας αλγόριθμος για face detection πάνω σε κρυπτογραφημένα δεδομένα οπότε εδώ απλώς εξυπηρετεί ως απόδειξη της λειτουργικότητας της βιβλιοθήκης.

10.1.1 Building a multiplexer classifier

Ένας multiplexer είναι επί της ουσίας η ακόλουθη γενικευμένη συνάρτηση σύγκρισης:

$$f_{a,\beta}(a,b) = \begin{cases} a, if\ a > b \\ \beta, otherwise \end{cases}$$

Η $f_{a,\beta}$ εκφράζεται ως ένας γραμμικός συνδυασμός του bit d = (α ≤ b):

$$f_{a,\beta}(d) = d \cdot \beta + (1-d) \cdot a = a + d \cdot (\beta - a)$$

Για την υλοποίηση αυτού του ταξινομητή ιδιωτικά, θα πρέπει να υπολογιστεί το $[\![d]\!]$ συγκρίνοντας τα α και b, διατηρώντας το αποτέλεσμα κρυπτογραφημένο με QR, και τότε αλλάζοντας το σύστημα κρυπτογράφησης σε Paillier. Έτσι, με το ομομορφισμό του Paillier και γνωρίζοντας τα a και β, μπορεί να υπολογιστεί η κρυπτογράφηση του $f_{a,\beta}(d)$:

$$[\![f_{a,\beta}(d)]\!] = [\![a]\!] \cdot [\![d]\!]^{\beta - a}$$

10.1.2 Viola and Jones face detection

Ο αλγόριθμος Viola και Jones για face detection είναι μια ειδική περίπτωση ενός AdaBoost ταξινομητή. Έστω X μια εικόνα που αναπαρίσταται ως ένα διάνυσμα ακεραίων και x ένα συγκεκριμένο παράθυρο εντοπισμού (detection window) (υποσύνολο των συντελεστών του X). Ο δυνατός ταξινομητής H θα είναι:

$$H(x) = sign\left(\sum_{i=1}^{t} a_i h_i(x)\right)$$

Όπου, h_t είναι οι αδύναμοι ταξινομητές για την σχέση $h_i(x) = sign(\langle x, y_i \rangle - \theta_i)$.

Εν προκειμένω, η Alice έχει την εικόνα και ο Bob τον ταξινομητή. Κανείς από τους δυο δεν θέλει να αποκαλύψει την είσοδό του στον άλλο. Χάρη στα building blocks, η Alice μπορεί να εκτελέσει τον ταξινομητή του Bob για την εικόνα της χωρίς να μάθει κάτι εκείνη για τις παραμέτρους και χωρίς ο Bob να μάθει κάποια πληροφορία για την εικόνα.

Οι αδύναμοι ταξινομητές μπορούν να θεωρηθούν πολυπλέκτες. Οπότε, $h_t(x) = f_{1,-1}(\langle x, y_t \rangle - \theta_t)$ και έτσι τελικά μπορεί να υπολογιστεί το H(x).

10.2 Performance evaluation setup

Η εκτιμήσεις επίδοσης έγιναν χρησιμοποιώντας δυο desktop υπολογιστές με ίδια configuration και ήταν στο ίδιο δίκτυο.

10.3 Building blocks performance

Η επίδοση εξετάζεται σε επίπεδο υπολογιστικού χρόνου μεταξύ χρήστη και server, bandwidth επικοινωνίας και round trips. Προκύπτει ότι όλα τα πρωτόκολλα είναι αποδοτικά.

10.3.1 Comparison protocols

Σύγκριση με την εκτίμηση των μη κρυπτογραφημένων πρωτοκόλλων εισόδου:

Bit size	A Computation	B Computation	Total Time	Communication	Interactions
10	14.11 ms	8.39 ms	105.5 ms	4.60 kB	3
20	18.29 ms	14.1 ms	117.5 ms	8.82 kB	3
32	22.9 ms	18.8 ms	122.6 ms	13.89 kB	3
64	34.7 ms	32.6 ms	134.5 ms	27.38 kB	3

Σύγκριση με την εκτίμηση των κρυπτογραφημένων πρωτοκόλλων εισόδου:

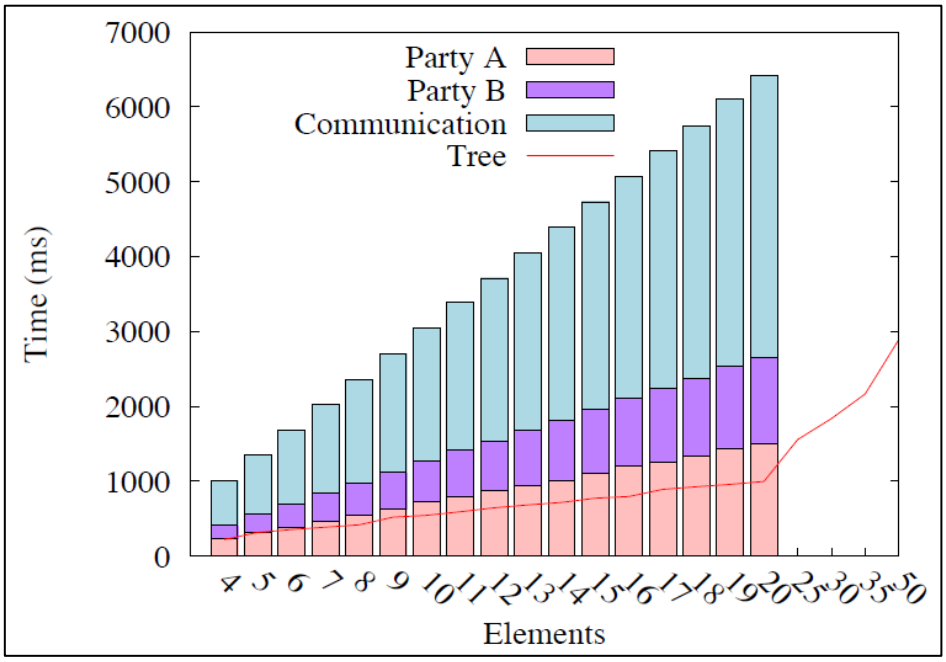
Protocol	Bit size	Computation		Total Time	Communication	Interactions
		Party A	Party B			
Comparison	64	45.34 ms	43.78 ms	190.9 ms	27.91 kB	6
Reversed Comp.	64	48.78 ms	42.49 ms	195.7 ms	27.91 kB	6

Αποτελέσματα από την εκτίμηση του πρωτοκόλλου κρυπτογραφημένων συστημάτων:

Party A Computation	Party B Computation	Total Time	Communication	Interactions
30.80 ms	255.3 ms	360.7 ms	420.1 kB	2

10.3.2 argmax

Η ακόλουθη εικόνα δείχνει τους χρόνους εκτέλεσης και το overhead της επικοινωνίας του argmax σε πρωτόκολλο κρυπτογραφημένης εισόδου. Οι μπάρες παρουσιάζουν την εκτέλεση του πρωτοκόλλου όταν οι συγκρίσεις εκτελούνται η μια μετά την άλλη, γραμμικά. Η γραμμή δείχνει την εκτέλεση όταν οι συγκρίσεις εκτελούνται παράλληλα.



10.3.3 Consequences of the latency on performances

Αξίζει να σημειωθεί ότι για τα πιο πολλά blocks, ο περισσότερος χρόνος εκτέλεσης ξοδεύεται στην επικοινωνία, καθώς το latency του δικτύου έχει μεγάλη επιρροή στις επιδόσεις των πρωτοκόλλων. Για την βελτίωση των επιδόσεων ενός ταξινομητή που έχει υλοποιηθεί με τα blocks αυτά θα πρέπει να εκτελεστούν αρκετά instances μερικών building blocks παράλληλα. Αυτό έγινε με την tree-based υλοποίηση του argmax πρωτόκολλου.

10.4 Classifier performance

Εδώ αναλύεται η εκτίμηση καθενός από τους ταξινομητές που περιεγράφηκαν παραπάνω. Τα μοντέλα εκπαιδεύονται με μη ιδιωτικό τρόπο με την scikit-learn. Χρησιμοποιήθηκαν τα ακόλουθα datasets από το UCI machine learning repository

1. the Wisconsin Diagnostic Breast Cancer data set
2. the Wisconsin Breast Cancer (Original) data set, a simplified version of the previous dataset
3. Credit Approval data set
4. Audiology (Standardized) data set
5. Nursery data set
6. ECG (electrocardiogram) classification data from Barni

Αυτά τα datasets είναι σενάρια τα οποία επαληθεύουν την ιδιωτικότητα του μοντέλου του server και της εισόδου του χρήστη. Με βάση την καταλληλότητα κάθε ταξινομητή, χρησιμοποιούνται τα data sets 2, 3 για το test του hyperplane decision ταξινομητή, τα set 1, 4, 5 για τον Naive Bayes ταξινομητή και τα set 5, 6 για τον decision tree ταξινομητή. Ο ακόλουθος πίνακας δείχνει τα αποτελέσματα που προέκυψαν. Οι ταξινομητές εκτελούνται το πολύ σε μερικά δευτερόλεπτα. Υπογραμμίζεται ότι αν τα data sets μεγαλώσουν, το μέγεθος του μοντέλου παραμένει το ίδιο, καθώς επηρεάζεται μόνο η φάση της εκπαίδευσης που συμβαίνει σε μη κρυπτογραφημένα δεδομένα πριν κάποιος χρησιμοποιήσει τον ταξινομητή. Έτσι, το κόστος των ταξινομητών τα είναι ίδιο ακόμα και για μεγάλα data sets.

Data set	Model size	Computation		Time per protocol		Total running time	Comm.	Interactions
		Client	Server	Compare	Dot product			
Breast cancer (2)	30	46.4 ms	43.8 ms	194 ms	9.67 ms	204 ms	35.84 kB	7
Credit (3)	47	55.5 ms	43.8 ms	194 ms	23.6 ms	217 ms	40.19 kB	7

(a) Linear Classifier. Time per protocol includes communication.

Data set	Specs.		Computation		Time per protocol		Total running time	Comm.	Interactions
	C	F	Client	Server	Prob. Comp.	Argmax			
Breast Cancer (1)	2	9	150 ms	104 ms	82.9 ms	396 ms	479 ms	72.47 kB	14
Nursery (5)	5	9	537 ms	368 ms	82.8 ms	1332 ms	1415 ms	150.7 kB	42
Audiology (4)	24	70	1652 ms	1664 ms	431 ms	3379 ms	3810 ms	1911 kB	166

(b) Naïve Bayes Classifier. C is the number of classes and F is the number of features. The Prob. Comp. column corresponds to the computation of the probabilities $p(c_i|x)$ (*cf.* Section 6). Time per protocol includes communication.

Data set	Tree Specs.		Computation		Time per protocol		FHE		Comm.	Interactions
	N	D	Client	Server	Compare	ES Change	Eval.	Decrypt		
Nursery (5)	4	4	1579 ms	798 ms	446 ms	1639 ms	239 ms	33.51 ms	2639 kB	30
ECG (6)	6	4	2297 ms	1723 ms	1410 ms	7406 ms	899 ms	35.1 ms	3555 kB	44

(c) Decision Tree Classifier. ES change indicates the time to run the protocol for changing encryption schemes. N is the number of nodes of the tree and D is its depth. Time per protocol includes communication.

10.5 Comparison to generic two-party tools

Ένα σύνολο από generic two-party ή multi-party εργαλεία υπολογισμού έχουν αναπτυχθεί (όπως το TASTY και το Fairplay) και υποστηρίζουν γενικές συναρτήσεις, συμπεριλαμβανομένων και των ταξινομητών που αναπτύχθηκαν. Όμως, είναι απαγορευτικά αργά για την συγκεκριμένη ρύθμιση. Η αποτελεσματικότητα εδώ προκύπτει από την εξειδίκευση στην λειτουργικότητα ταξινόμησης (classification functionality). Μετά από πειράματα φάνηκε ότι τα ειδικά Πρωτόκολλα που αναπτύχθηκαν βελτιώνουν την επίδοση κατά τάξεις μεγέθους.

11. Conclusion

Σε αυτό το paper δημιουργήθηκαν τρεις κύριοι privacy-preserving ταξινομητές και αναπτύχθηκε μια βιβλιοθήκη για building blocks που ενδείκνυνται και για την κατασκευή κι άλλων ταξινομητών. Επιπλέον, εξετάστηκε η επίδοση των ταξινομητών και της βιβλιοθήκης σε πραγματικά data sets.

12. Resources

Raphael Bost, Raluca Ada Popa, Stephen Tu, Shafi Goldwasser: Machine Learning Classification over Encrypted Data. NDSS 2015. https://eprint.iacr.org/2014/331