Bern University of Applied Sciences

Pseudo-Code Algorithms for Verifiable Re-Encryption Mix-Nets

Rolf Haenni (P. Locher, R. E. Koenig, E. Dubuis)

Voting'17 (FC'17), Sliema, Malta, April 7, 2017

Outline

- Introduction
- Context of CHVote Project
- CHVote Voting Protocol
- Pseudo-Code Algorithms for Verifiable Mix-Nets
- Conclusion

Outline

- Introduction
- Context of CHVote Project
- CHVote Voting Protocol
- Pseudo-Code Algorithms for Verifiable Mix-Nets
- Conclusion

Verifiable Re-Encryption Mix-Nets

- Variant of Chaum's onion mix-net by Park et al. (1993)

Verifiable Re-Encryption Mix-Nets

- Variant of Chaum's onion mix-net by Park et al. (1993)
- Important building blocks in electronic voting protocols
  - Shuffling of encrypted votes
  - Proving validity of complex ballots

Verifiable Re-Encryption Mix-Nets

- Variant of Chaum's onion mix-net by Park et al. (1993)
- Important building blocks in electronic voting protocols
  - Shuffling of encrypted votes
  - Proving validity of complex ballots
- NIZK shuffle proofs
  - Wikström and Terelius (2009, 2010)
  - Groth and Bayer (2010, 2012)
  - Lipmaa, Zhang, Fauzi, Zajac (2012, 2015, 2016)

Verifiable Re-Encryption Mix-Nets

- Variant of Chaum's onion mix-net by Park et al. (1993)
- Important building blocks in electronic voting protocols
  - Shuffling of encrypted votes
  - Proving validity of complex ballots
- NIZK shuffle proofs
  - Wikström and Terelius (2009, 2010)
  - Groth and Bayer (2010, 2012)
  - Lipmaa, Zhang, Fauzi, Zajac (2012, 2015, 2016)
- Implementations
  - Verificatum Mix-Net (since 2008)
  - UniCrypt (since 2014)
  - RPC-based Ximix (vVote, 2014)
  - PANORAMIX (?)

Obstacles in Practice

- “Do-it-yourself”
  - Complexity of theory
  - Subtleties and pitfalls of writing cryptographic code
  - Limited cryptographic background of software engineers (and auditors)
  - Limited software engineering background of cryptographers

Obstacles in Practice

- "Do-it-yourself"
  - Complexity of theory
  - Subtleties and pitfalls of writing cryptographic code
  - Limited cryptographic background of software engineers (and auditors)
  - Limited software engineering background of cryptographers
- Using a library
  - Licensing and ownership restrictions
  - Dependency to third-party code
  - System certification

Outline

- Introduction
- Context of CHVote Project
- CHVote Voting Protocol
- Pseudo-Code Algorithms for Verifiable Mix-Nets
- Conclusion

Direct Democracy in Switzerland

- Up to four election days per year
  - Elections
  - Mandatory referendums
  - Optional referendums ($>$50k signatures)
  - Popular initiatives ($>$100k signatures)

Direct Democracy in Switzerland

- Up to four election days per year
  - Elections
  - Mandatory referendums
  - Optional referendums (>50k signatures)
  - Popular initiatives (>100k signatures)
- On four different political levels
  - Federal
  - Cantonal
  - Municipal
  - Pastoral

Direct Democracy in Switzerland

- Up to four election days per year
  - Elections
  - Mandatory referendums
  - Optional referendums ($>$50k signatures)
  - Popular initiatives ($>$100k signatures)
- On four different political levels
  - Federal
  - Cantonal
  - Municipal
  - Pastoral
- Up to 10 different election topics per election day

E-Voting Tradition in Switzerland

- Classical voting channels
  - Polling station
  - Postal voting (approx. 90%)

E-Voting Tradition in Switzerland

- Classical voting channels
  - Polling station
  - Postal voting (approx. 90%)
- Non-verifiable e-voting systems (1st generation)
  - Canton of Geneva (since 2003)
  - Canton of Zürich (2004–2015)
  - Canton of Neuchâtel (Scytl, 2005–2015)

E-Voting Tradition in Switzerland

- Classical voting channels
  - Polling station
  - Postal voting (approx. 90%)
- Non-verifiable e-voting systems (1st generation)
  - Canton of Geneva (since 2003)
  - Canton of Zürich (2004–2015)
  - Canton of Neuchâtel (Scytl, 2005–2015)
- Collaborations with 10 other cantons (since 2009)

E-Voting Tradition in Switzerland

- Classical voting channels
  - Polling station
  - Postal voting (approx. 90%)
- Non-verifiable e-voting systems (1st generation)
  - Canton of Geneva (since 2003)
  - Canton of Zürich (2004–2015)
  - Canton of Neuchâtel (Scytl, 2005–2015)
- Collaborations with 10 other cantons (since 2009)
- Target audience: Swiss citizens living abroad

Legal Ordinance on Electronic Voting

- Effective since December 2013

Legal Ordinance on Electronic Voting

- Effective since December 2013
- Enhanced security requirements
  - End-to-end encryption
  - Individual verifiability (recorded-as-intended)
  - Universal verifiability (counted-as-recorded)
  - Distribution of trust (shared encryption key, mix-net)

Legal Ordinance on Electronic Voting

- Effective since December 2013
- Enhanced security requirements
  - End-to-end encryption
  - Individual verifiability (recorded-as-intended)
  - Universal verifiability (counted-as-recorded)
  - Distribution of trust (shared encryption key, mix-net)
- Two-step expansion
  - Current systems: max. 10/30% of federal/cantonal electorate
  - Step 1: max. 30/50% of federal/cantonal electorate
  - Step 2: 100% electorate

Legal Ordinance on Electronic Voting

- Effective since December 2013
- Enhanced security requirements
  - End-to-end encryption
  - Individual verifiability (recorded-as-intended)
  - Universal verifiability (counted-as-recorded)
  - Distribution of trust (shared encryption key, mix-net)
- Two-step expansion
  - Current systems: max. 10/30% of federal/cantonal electorate
  - Step 1: max. 30/50% of federal/cantonal electorate
  - Step 2: 100% electorate
- Two competing 2nd generation projects
  - Canton of Geneva (CHVote)
  - Swiss Post (Scytl)

CHVote Project

- Project goals
  - New implementation from scratch
  - Reach second expansion stage in one step (100% electorate)
  - Developed, hosted and operated entirely by the Geneva Canton

CHVote Project

- Project goals
  - New implementation from scratch
  - Reach second expansion stage in one step (100% electorate)
  - Developed, hosted and operated entirely by the Geneva Canton
- Strategy
  - Collaboration with academia
  - State-of-the-art technologies
  - Full transparency
  - High-quality open documentation
  - Open-source license (Affero GPL)
  - Invitation to public code reviewing

Outline

CHVote Voting Protocol

- Collaboration with Bern University of Applied Sciences

CHVote Voting Protocol

- Collaboration with Bern University of Applied Sciences
- Cast-as-intended verifiability à la Norway
  - Code sheets are sent to voters by postal mail
  - Codes for the selected candidates are displayed after voting
  - If codes match, voter submits confirmation

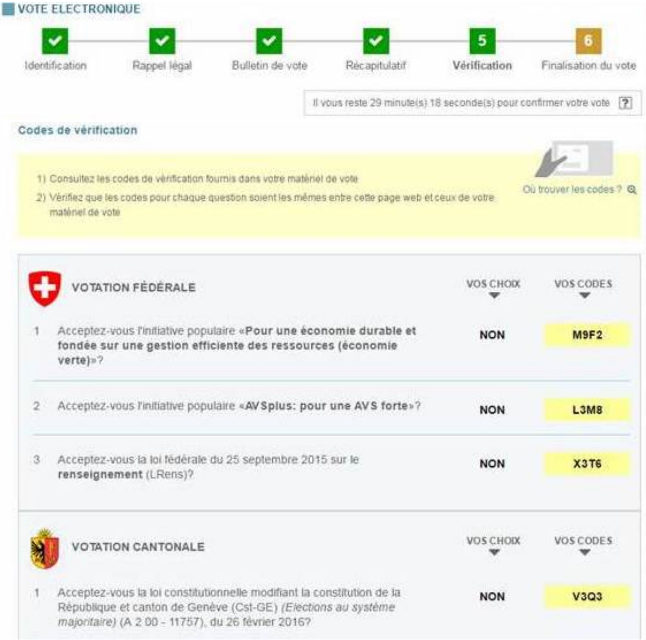
CHVote Voting Protocol

- Collaboration with Bern University of Applied Sciences
- Cast-as-intended verifiability à la Norway
  - Code sheets are sent to voters by postal mail
  - Codes for the selected candidates are displayed after voting
  - If codes match, voter submits confirmation
- Key cryptographic ingredients
  - Distributed generation of codes
  - Oblivious transfer of selected codes
  - Verifiable mix-net
  - Distributed decryption with shared ElGamal private key

CHVote Voting Protocol

- Collaboration with Bern University of Applied Sciences
- Cast-as-intended verifiability à la Norway
  - Code sheets are sent to voters by postal mail
  - Codes for the selected candidates are displayed after voting
  - If codes match, voter submits confirmation
- Key cryptographic ingredients
  - Distributed generation of codes
  - Oblivious transfer of selected codes
  - Verifiable mix-net
  - Distributed decryption with shared ElGamal private key
- Paper presented at E-Vote-ID 2016

Liste de codes pour la carte n° 5874-8863-1400-8743			
Votation fédérale			
Question 1 Acceptez-vous l'arrêté fédéral du 20 juin 2013 portant règlement du financement et de l'aménagement de l'infrastructure ferroviaire (Contre-projet direct à l'initiative populaire "Pour les transports publics", qui a été retirée) ?	Oui A2B4	Non J5B9	Blanc Z8H5
Question 2 Acceptez-vous l'initiative populaire "Financer l'avortement est une affaire privée - Alléger l'assurance-maladie en radiant les coûts de l'interruption de grossesse de l'assurance de base" ?	Oui P8H3	Non X2A7	Blanc Q3L7
Votation cantonale			
Question 1 Acceptez-vous l'initiative 143 «Pour une véritable politique d'accueil de la Petite enfance» ?	Oui U6T4	Non P3D6	Blanc S6C2
Question 2 Acceptez-vous la loi constitutionnelle modifiant la constitution de la République et canton de Genève (Contreprojet à l'IN 143) (A 2 00 – 10895), du 15 décembre 2011 ?	Oui N4F2	Non M2A3	Blanc Q9L5
Question 3 **Question subsidiaire**: Si l'initiative (IN 143 «Pour une véritable politique d'accueil de la Petite enfance») et le contreprojet sont acceptés, lequel des deux a-t-il votre préférence ? Initiative 143 ? Contreprojet ?	IN K9W9	CP T3S6	Blanc Y2V4


VOTE ELECTRONIQUE
Identification
Rappel légal
Bulletin de vote
Récapitulatif
5
Vérification
6
Finalisation du vote
Il vous reste 29 minute(s) 18 seconde(s) pour confirmer votre vote ?
Codes de vérification
1) Consultez les codes de vérification fournis dans votre matériel de vote
2) Vérifiez que les codes pour chaque question soient les mêmes entre cette page web et ceux de votre matériel de vote
Où trouver les codes ?
VOTATION FÉDÉRALE
VOS CHOIX
VOS CODES
1 Acceptez-vous l'initiative populaire «Pour une économie durable et fondée sur une gestion efficiente des ressources (économie verte)»?
NON
M9F2
2 Acceptez-vous l'initiative populaire «AVSplus: pour une AVS forte»?
NON
L3M8
3 Acceptez-vous la loi fédérale du 25 septembre 2015 sur le renseignement (LRens)?
NON
X3T6
VOTATION CANTONALE
VOS CHOIX
VOS CODES
1 Acceptez-vous la loi constitutionnelle modifiant la constitution de la République et canton de Genève (Cst-GE) (Elections au système majoritaire) (A 2 00 - 11757), du 26 février 2016?
NON
V3Q3

CHVote Protocol Specification

- Publication on April 20 (together with prototype source code)

CHVote Protocol Specification

- Publication on April 20 (together with prototype source code)
- Self-contained and comprehensive document ($\sim$120 pages)
  - Description of election use cases
  - Mathematical and cryptographic background
  - Details of encoding and hashing algorithms
  - Adversary and trust assumptions
  - Cryptographic and election parameters
  - Recommendations for group sizes, key lengths, code lengths

CHVote Protocol Specification

- Publication on April 20 (together with prototype source code)
- Self-contained and comprehensive document (∼120 pages)
  - Description of election use cases
  - Mathematical and cryptographic background
  - Details of encoding and hashing algorithms
  - Adversary and trust assumptions
  - Cryptographic and election parameters
  - Recommendations for group sizes, key lengths, code lengths
- Three main protocols (three sub-protocols each)
  - Pre-election
  - Election
  - Post-election

CHVote Protocol Specification

- Publication on April 20 (together with prototype source code)
- Self-contained and comprehensive document ($\sim$120 pages)
  - Description of election use cases
  - Mathematical and cryptographic background
  - Details of encoding and hashing algorithms
  - Adversary and trust assumptions
  - Cryptographic and election parameters
  - Recommendations for group sizes, key lengths, code lengths
- Three main protocols (three sub-protocols each)
  - Pre-election
  - Election
  - Post-election
- About 60 pseudo-code algorithms

Phase	Election Admin.	Election Authority	Printing Authority	Voter	Voting Client	Bulletin Board	Protocol Nr.
1. Pre-Election	•	•	•	•		•	
1.1 Election Preparation	•	•				•	6.1
1.2 Printing of Code Sheets		•	•	•		•	6.2
1.3 Key Generation		•				•	6.3
2. Election		•		•	•	•	
2.1 Candidate Selection				•	•	•	6.4
2.2 Vote Casting		•			•	•	6.5
2.3 Vote Confirmation		•		•	•	•	6.6
3. Post-Election	•	•				•	
3.1 Mixing		•				•	6.7
3.2 Decryption		•				•	6.8
3.3 Tallying	•					•	6.9

Voting Client		Bulletin Board		Election Authority $j \in \{1,\dots,s\}$
knows $i, X_i, \mathbf{k}_i, \mathbf{s}$		knows $\mathbf{pk}$		knows $pk, \mathbf{n}, \mathbf{K}, \mathbf{P}_j, \hat{\mathbf{x}}, B_j \leftarrow \langle\rangle$
	$\xleftarrow{\mathbf{pk}}$			
$pk \leftarrow \mathsf{GetPublicKey}(\mathbf{pk})$ $(\alpha, \mathbf{r}) \leftarrow \mathsf{GenBallot}(X_i, \mathbf{s}, pk)$				
	$\xrightarrow{i, \alpha}$			
			$\xrightarrow{i, \alpha}$	
				if $\neg\mathsf{CheckBallot}(i, \alpha, pk, \mathbf{K}, \hat{\mathbf{x}}, B)$ **abort** $(\beta_j, \mathbf{r}) \leftarrow \mathsf{GenResponse}(i, \mathbf{a}, pk, \mathbf{n}, \mathbf{K}, \mathbf{P}_j)$ $B_j \leftarrow B_j \,\Vert\, \langle(i, \alpha, \mathbf{r})\rangle$
			$\xleftarrow{i, \beta_j}$	
		$\boldsymbol{\beta}_i = (\beta_1, \dots, \beta_s)$		
	$\xleftarrow{\boldsymbol{\beta}_i}$			
$\mathbf{P}_{\mathbf{s}} \leftarrow \mathsf{GetPointMatrix}(\boldsymbol{\beta}_i, \mathbf{k}_i, \mathbf{s}, \mathbf{r})$ $\mathbf{rc}_{\mathbf{s}} \leftarrow \mathsf{GetReturnCodes}(\mathbf{s}, \mathbf{P}_{\mathbf{s}})$				

Protocol 6.5: Vote Casting

Algorithm: $\mathsf{GenBallot}(X, \mathbf{s}, pk)$

Input: Voting code $X \in A_X^{\ell_X}$
Selection $\mathbf{s} = (s_1, \ldots, s_k)$, $1 \leqslant s_1 < \cdots < s_k$
Encryption key $pk \in \mathbb{G}_q \backslash \{1\}$

$x \leftarrow \mathsf{ToInteger}(X)$ // see Alg. 4.7
$\hat{x} \leftarrow \hat{g}^x \bmod \hat{p}$
$\mathbf{q} \leftarrow \mathsf{GetSelectedPrimes}(\mathbf{s})$ // $\mathbf{q} = (q_1, \ldots, q_k)$, see Alg. 7.19
$m \leftarrow \prod_{i=1}^k q_i$
if $m \geqslant p$ **then**
  return $\perp$ // (k, n) is incompatible with p
$(\mathbf{a}, \mathbf{r}) \leftarrow \mathsf{GenQuery}(\mathbf{q}, pk)$ // $\mathbf{a} = (a_1, \ldots, a_k)$, $\mathbf{r} = (r_1, \ldots, r_k)$, see Alg. 7.20
$a \leftarrow \prod_{i=1}^k a_i \bmod p$
$r \leftarrow \sum_{i=1}^k r_i \bmod q$
$b \leftarrow g^r \bmod p$
$\pi \leftarrow \mathsf{GenBallotProof}(x, m, r, \hat{x}, a, b, pk)$ // $\pi = (t, s)$, see Alg. 7.21
$\alpha \leftarrow (\hat{x}, \mathbf{a}, b, \pi)$
return $(\alpha, \mathbf{r})$ // $\alpha \in \mathbb{Z}_{\hat{q}} \times \mathbb{G}_q^k \times \mathbb{G}_q \times ((\mathbb{G}_{\hat{q}} \times \mathbb{G}_q^2) \times (\mathbb{Z}_{\hat{q}} \times \mathbb{G}_q \times \mathbb{Z}_q))$, $\mathbf{r} \in \mathbb{Z}_q^k$

Algorithm 7.18: Generates a ballot based on the selection $\mathbf{s}$ and the voting code X. The ballot includes an OT query $\mathbf{a}$ and a NIZKP π. The algorithm also returns the randomizations $\mathbf{r}$ of the OT query, which are required in Alg. 7.27 to derive the transferred messages from the OT response.

```java
/**
 * Algorithm 7.18: GenBallot
 *
 * @param upper_x the voting code
 * @param bold_s  voters selection (indices)
 * @param pk      the public encryption key
 * @return the combined ballot, OT query and random elements used
 */
public BallotQueryAndRand genBallot(String upper_x, List<Integer> bold_s, EncryptionPublicKey pk) {
    BigInteger x = conversion.toInteger(upper_x, publicParameters.getUpper_a_x());
    BigInteger x_circ = modExp(g_circ, x, p_circ);
    List<BigInteger> bold_q = computeBoldQ(bold_s);
    BigInteger m = computeM(bold_q, p);
    ObliviousTransferQuery query = genQuery(bold_q, pk);
    BigInteger a = computeA(query, p);
    BigInteger r = computeR(query, q);
    BigInteger b = modExp(g, r, p);
    NonInteractiveZKP pi = genBallotProof(x, m, r, x_circ, a, b, pk);
    BallotAndQuery alpha = new BallotAndQuery(x_circ, query.getBold_a(), b, pi);

    return new BallotQueryAndRand(alpha, query.getBold_r());
}
```

Outline

- Introduction
- Context of CHVote Project
- CHVote Voting Protocol
- Pseudo-Code Algorithms for Verifiable Mix-Nets
- Conclusion

Crypto-Algorithms in Pseudo-Code

- Ideal interface between cryptographers, developers, auditors
  - Cryptographers can write, read, and check pseudo-code
  - Developers can derive real code from pseudo-code
  - Auditors can check if pseudo-code and real code match
  - Useful for security proofs

Crypto-Algorithms in Pseudo-Code

- Ideal interface between cryptographers, developers, auditors
  - Cryptographers can write, read, and check pseudo-code
  - Developers can derive real code from pseudo-code
  - Auditors can check if pseudo-code and real code match
  - Useful for security proofs
- Rarely used in ...
  - cryptographic literature
  - electronic voting protocols

Crypto-Algorithms in Pseudo-Code

- Ideal interface between cryptographers, developers, auditors
  - Cryptographers can write, read, and check pseudo-code
  - Developers can derive real code from pseudo-code
  - Auditors can check if pseudo-code and real code match
  - Useful for security proofs
- Rarely used in ...
  - cryptographic literature
  - electronic voting protocols
- Often used in standards (FIPS, RFC, PKCS, ...)

FIPS PUB 186-4: Digital Signature Standard (DSS)

A.2.3 Verifiable Canonical Generation of the Generator *g*

Input:

1.	*p*, *q*	The primes.
2.	*domain_parameter_seed*	The seed used during the generation of *p* and *q*.
3.	*index*	The index to be used for generating *g*. *index* is a bit string of length 8 that represents an unsigned integer.

Process:

1. If (*index* is incorrect), then return **INVALID.**
2. $N = \mathbf{len}(q)$.
3. $e = (p - 1)/q$.
4. $count = 0$.
5. $count = count + 1$.
6. If ($count = 0$), then return **INVALID**.
7. *U* = *domain_parameter_seed* || "ggen" || *index* || *count*.
8. $W = \mathbf{Hash}(U)$.
9. $g = W^e \bmod p$.
10. If ($g < 2$), then go to step 5. Comment: If a generator has not been found.
11. Return **VALID** and the value of *g*.

Pseudo-Code for Verifiable Mix-Nets

- Two inputs:

$$\begin{aligned} \mathbf{e} &= \text{input list of ElGamal encryptions} \\ pk &= \text{encryption public key} \end{aligned}$$

Pseudo-Code for Verifiable Mix-Nets

- Two inputs:

$$\begin{aligned} \mathbf{e} &= \text{input list of ElGamal encryptions} \\ pk &= \text{encryption public key} \end{aligned}$$

- Three main algorithms:

$$\begin{aligned} (\mathbf{e}', \mathbf{r}', \psi) &\leftarrow \mathsf{GenShuffle}(\mathbf{e}, pk) \\ \pi &\leftarrow \mathsf{GenProof}(\mathbf{e}, \mathbf{e}', \mathbf{r}', \psi, pk) \\ \mathit{true}/\mathit{false} &\leftarrow \mathsf{CheckProof}(\pi, \mathbf{e}, \mathbf{e}', pk) \end{aligned}$$

Algorithm: $\mathsf{GenProof}(\mathbf{e}, \mathbf{e}', \mathbf{r}', \psi, pk)$

Input: ElGamal encryptions $\mathbf{e} = (e_1, \dots, e_N)$, $e_i = (a_i, b_i) \in \mathbb{G}_q^2$
Shuffled ElGamal encryptions $\mathbf{e}' = (e'_1, \dots, e'_N)$, $e'_i = (a'_i, b'_i) \in \mathbb{G}_q^2$
Re-encryption randomizations $\mathbf{r}' = (r'_1, \dots, r'_N)$, $r'_i \in \mathbb{Z}_q$
Permutation $\psi = (j_1, \dots, j_N) \in \Psi_N$
Encryption key $pk \in \mathbb{G}_q$

$(\mathbf{c}, \mathbf{r}) \leftarrow \mathsf{GenCommitment}(\psi)$ // $\mathbf{c} = (c_1, \dots, c_N)$, $\mathbf{r} = (r_1, \dots, r_N)$
for $i = 1, \dots, N$ **do**
  $u_i \leftarrow \mathsf{Hash}((\mathbf{e}, \mathbf{e}', \mathbf{c}), i)$
  $u'_i \leftarrow u_{j_i}$
$\mathbf{u} \leftarrow (u_1, \dots, u_N)$, $\mathbf{u}' \leftarrow (u'_1, \dots, u'_N)$
$(\hat{\mathbf{c}}, \hat{\mathbf{r}}) \leftarrow \mathsf{GenCommitmentChain}(h, \mathbf{u}')$ // $\hat{\mathbf{c}} = (\hat{c}_1, \dots, \hat{c}_N)$, $\hat{\mathbf{r}} = (\hat{r}_1, \dots, \hat{r}_N)$
$\bar{r} \leftarrow \sum_{i=1}^N r_i \bmod q$
$v_N \leftarrow 1$
for $i = N-1, \dots, 1$ **do**
  $v_i \leftarrow u'_{i+1} v_{i+1} \bmod q$
$\hat{r} \leftarrow \sum_{i=1}^N \hat{r}_i v_i \bmod q$
$\tilde{r} \leftarrow \sum_{i=1}^N r_i u_i \bmod q$
$r' \leftarrow \sum_{i=1}^N r'_i u_i \bmod q$
for $i = 1, \dots, 4$ **do**
  $\omega_i \in_R \mathbb{Z}_q$

for $i = 1, \dots, N$ **do**

$\quad \hat{\omega}_i \in_R \mathbb{Z}_q,\ \omega'_i \in_R \mathbb{Z}_q$

$t_1 \leftarrow g^{\omega_1} \bmod p$

$t_2 \leftarrow g^{\omega_2} \bmod p$

$t_3 \leftarrow g^{\omega_3} \prod_{i=1}^{N} h_i^{\omega'_i} \bmod p$

$(t_{4,1}, t_{4,2}) \leftarrow (pk^{-\omega_4} \prod_{i=1}^{N} (a'_i)^{\omega'_i} \bmod p, g^{-\omega_4} \prod_{i=1}^{N} (b'_i)^{\omega'_i} \bmod p)$

$\hat{c}_0 \leftarrow h$

for $i = 1, \dots, N$ **do**

$\quad \hat{t}_i \leftarrow g^{\hat{\omega}_i} \hat{c}_{i-1}^{\omega'_i} \bmod p$

$y \leftarrow (\mathbf{e}, \mathbf{e}', \mathbf{c}, \hat{\mathbf{c}}, pk),\ t \leftarrow (t_1, t_2, t_3, (t_{4,1}, t_{4,2}), (\hat{t}_1, \dots, \hat{t}_N))$

$c \leftarrow \mathsf{Hash}(y, t)$

$s_1 \leftarrow \omega_1 + c \cdot \bar{r} \bmod q$

$s_2 \leftarrow \omega_2 + c \cdot \hat{r} \bmod q$

$s_3 \leftarrow \omega_3 + c \cdot \tilde{r} \bmod q$

$s_4 \leftarrow \omega_4 + c \cdot r' \bmod q$

for $i = 1, \dots, N$ **do**

$\quad \hat{s}_i \leftarrow \hat{\omega}_i + c \cdot \hat{r}_i \bmod q,\ s'_i \leftarrow \omega'_i + c \cdot u'_i \bmod q$

$s \leftarrow (s_1, s_2, s_3, s_4, (\hat{s}_1, \dots, \hat{s}_N), (s'_1, \dots, s'_N))$

$\pi \leftarrow (t, s, \mathbf{c}, \hat{\mathbf{c}})$

return π $\quad // \pi \in (\mathbb{G}_q^3 \times \mathbb{G}_q^2 \times \mathbb{G}_q^N) \times (\mathbb{Z}_q^4 \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N) \times \mathbb{G}_q^N \times \mathbb{G}_q^N$

Implementation within CHVote

- By a single developer of the CHVote project
  - Basic cryptographic background knowledge
  - Little experience in implementing cryptographic protocols
  - Familiar with formal/mathematical notations
  - Strong background in software development
  - Many years of practical experience (e-voting context)

Implementation within CHVote

- By a single developer of the CHVote project
  - Basic cryptographic background knowledge
  - Little experience in implementing cryptographic protocols
  - Familiar with formal/mathematical notations
  - Strong background in software development
  - Many years of practical experience (e-voting context)
- Finished within 3–4 weeks, including...
  - Optimized performance
  - Full test coverage
  - Documentation
  - Ready to go open-source

"It's taken some blood and sweat, but I've implemented Algorithm 5.44 (GenShuffleProof), along with a unit test for it, with all computations done by hand ..."

"It's taken some blood and sweat, but I've implemented Algorithm 5.44 (GenShuffleProof), along with a unit test for it, with all computations done by hand ..."

"Thanks for writing this [pseudo-code] implementation of Wikström's proof of a shuffle, it would have been MUCH harder starting from his papers."

"It's taken some blood and sweat, but I've implemented Algorithm 5.44 (GenShuffleProof), along with a unit test for it, with all computations done by hand ..."

"Thanks for writing this [pseudo-code] implementation of Wikström's proof of a shuffle, it would have been MUCH harder starting from his papers."

"Hope it gets accepted, since it's the kind of work cryptography needs more of."

Outline

Conclusion

- Pseudo-code algorithms facilitate the implementation of verifiable mix-net

Conclusion

- Pseudo-code algorithms facilitate the implementation of verifiable mix-net
- Implementation possible by qualified software developers

Conclusion

- Pseudo-code algorithms facilitate the implementation of verifiable mix-net
- Implementation possible by qualified software developers
- Positive feedback from e-voting practitioners

Conclusion

- Pseudo-code algorithms facilitate the implementation of verifiable mix-net
- Implementation possible by qualified software developers
- Positive feedback from e-voting practitioners
- Recommended interface between cryptographers and software developers (e.g. in the context of voting protocols)