Cobra: Toward Concurrent Ballot Authorization for Internet Voting

Philipp Locher

Seminar, E-Voting Group, BFH

December 19, 2012

Outline

Introduction

Cobra: The first coercion-resistant system to offer concurrent ballot authorization.

- Coercion-resistant, end-to-end verifiable internet voting
- JCJ
- Tallying (ballot/vote authorization) $\rightarrow$ computationally expensive
- Denial of service attacks caused by floods of fake ballots
- Different attempts to improve ballot authorization $\rightarrow$ well known: Spycher, Schläpfer, Koenig...

Introduction

Cobra: The first coercion-resistant system to offer concurrent ballot authorization.

- Coercion-resistant, end-to-end verifiable internet voting
- JCJ
- Tallying (ballot/vote authorization) $\rightarrow$ computationally expensive
- Denial of service attacks caused by floods of fake ballots
- Different attempts to improve ballot authorization $\rightarrow$ well known: Spycher, Schläpfer, Koenig...

Introduction

Cobra: The first coercion-resistant system to offer concurrent ballot authorization.

- Coercion-resistant, end-to-end verifiable internet voting
- JCJ
- Tallying (ballot/vote authorization) $\rightarrow$ computationally expensive
- Denial of service attacks caused by floods of fake ballots
- Different attempts to improve ballot authorization $\rightarrow$ well known: Spycher, Schläpfer, Koenig...

Introduction

Cobra: The first coercion-resistant system to offer concurrent ballot authorization.

- Coercion-resistant, end-to-end verifiable internet voting
- JCJ
- Tallying (ballot/vote authorization) $\rightarrow$ computationally expensive
- Denial of service attacks caused by floods of fake ballots
- Different attempts to improve ballot authorization $\rightarrow$ well known: Spycher, Schläpfer, Koenig...

Introduction

Cobra: The first coercion-resistant system to offer concurrent ballot authorization.

- Coercion-resistant, end-to-end verifiable internet voting
- JCJ
- Tallying (ballot/vote authorization) $\rightarrow$ computationally expensive
- Denial of service attacks caused by floods of fake ballots
- Different attempts to improve ballot authorization $\rightarrow$ well known: Spycher, Schläpfer, Koenig...

Introduction

Cobra: The first coercion-resistant system to offer concurrent ballot authorization.

- Coercion-resistant, end-to-end verifiable internet voting
- JCJ
- Tallying (ballot/vote authorization) $\rightarrow$ computationally expensive
- Denial of service attacks caused by floods of fake ballots
- Different attempts to improve ballot authorization $\rightarrow$ well known: Spycher, Schläpfer, Koenig...

Introduction

- In previous proposals the bulk of computation can not be done until the last ballot has been cast.
- Cobra's new approach: concurrent ballot authorization
- Proof of concept $\rightarrow$ registration is too slow to be viable

Introduction

- In previous proposals the bulk of computation can not be done until the last ballot has been cast.
- Cobra's new approach: concurrent ballot authorization
- Proof of concept $\rightarrow$ registration is too slow to be viable

Introduction

- In previous proposals the bulk of computation can not be done until the last ballot has been cast.
- Cobra's new approach: concurrent ballot authorization
- Proof of concept $\rightarrow$ registration is too slow to be viable

Outline

Concurrent Ballot Authorization

Preliminary 1:

Standard setting of a prime-order subgroup G_q of $\mathbb{Z}_p^*$ where DDH is hard

Preliminary 2:

The encryption of a message m is denoted as $[\![m]\!]$

Preliminary 3:

The encryption scheme is additively homomorphic, rerandomizable an the plaintext space is small (e.g. exponential Elgamal)

Preliminary 4:

Mix & Match protocol (Jakobsson/Juels 2000) for secure function evaluation (SFE)

Example AND:

In		Out
⟦0⟧	⟦0⟧	⟦0⟧
⟦0⟧	⟦1⟧	⟦0⟧
⟦1⟧	⟦0⟧	⟦0⟧
⟦1⟧	⟦1⟧	⟦1⟧

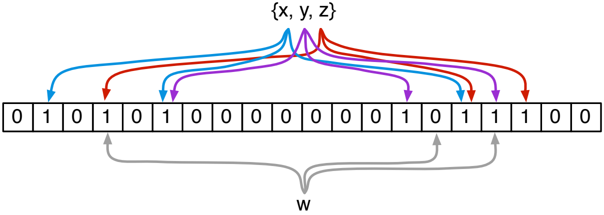
$\rightarrow$

In		Out
⟦0⟧	⟦1⟧	⟦0⟧
⟦1⟧	⟦1⟧	⟦1⟧
⟦1⟧	⟦0⟧	⟦0⟧
⟦0⟧	⟦0⟧	⟦0⟧

Preliminary 5:

Bloom filter (Bloom 1970) for highly efficient membership testing on the cost of potential false positives

- Stores n elements in $O(n)$ space
- Tests for membership in $O(1)$
- $\Pr[\mathrm{Query}(q, B) = \mathrm{TRUE} \mid q \in B] = 1$
- $\Pr[\mathrm{Query}(q, B) = \mathrm{FALSE} \mid q \notin B] < 1$



Concurrent Ballot Authorization

Ballot Authorization Function f **(1/3):**

If $\langle [\![x]\!], [\![v]\!] \rangle$ is a ballot where x is voter's credential and v voter's vote, then a function f

$$f([\![x]\!]) = \begin{cases} [\![1]\!] & x \in \text{Roster} \\ [\![0]\!] & \text{otherwise} \end{cases}$$

is required to realize concurrent ballot authorization.

Concurrent Ballot Authorization

Ballot Authorization Function f **(2/3):**

If such a function f existed, then under encryption

$$[\![\text{v'}]\!] = [\![v \cdot f(x)]\!]$$

could be computed. This would nullify ("zero-out") the vote if the credential was invalid. The result $[\![v']\!]$ is a so called authorised ballot.

Ballot Authorization Function f (3/3):

The homomorphic multiplication in an additively homomorphic encryption scheme can be accomplished indirectly with Mix & Match, evaluation the following truth table on $[\![x]\!]$:

In	Out
$[\![0]\!]$	$[\![0]\!]$
$[\![1]\!]$	$[\![v]\!]$

Concurrent Ballot Authorization

Implementing f with a Bloom Filter:

- Encrypted Bloom filter, (m, k)-EBF, with length m and k cryptographic hash functions
- Insertion (during registration) and querying (during casting) are performed homomorphically under encryption
- Notice: While inserting a value a, a is secret, but a is a public value when later querying value a
- An (m, k)-EBF implements the functions: *Setup*, *Prepare*, *Insert*, *Flatten* and *Query*

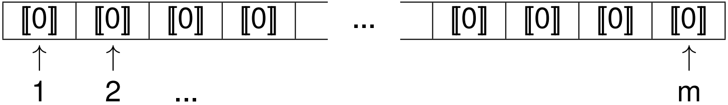
Concurrent Ballot Authorization

Encrypted Bloom Filter: Setup

Input: m, k

Output:

m-array EBF



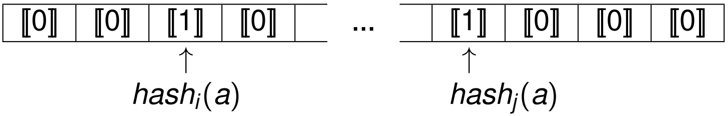
and k-hash functions with output space $[1, m]$

Encrypted Bloom Filter: Prepare

Input: a

Output:

EBF_a



and proof π_a proofing that EBF_a contains only $[\![0]\!]$ and $[\![1]\!]$ and that the sum over all entries is equal $[\![k]\!]$

Concurrent Ballot Authorization

Encrypted Bloom Filter: Insert

Input: EBF, EBF_a

EBF

⟦0⟧	⟦2⟧	⟦0⟧	⟦3⟧		...		⟦1⟧	⟦0⟧	⟦2⟧	⟦0⟧

EBF_a

$+$

⟦0⟧	⟦0⟧	⟦1⟧	⟦0⟧		...		⟦1⟧	⟦0⟧	⟦0⟧	⟦0⟧

Output:

$=$

EBF'

⟦0⟧	⟦2⟧	⟦1⟧	⟦3⟧		...		⟦2⟧	⟦0⟧	⟦2⟧	⟦0⟧

Concurrent Ballot Authorization

Encrypted Bloom Filter: Flatten

Input: EBF

EBF

⟦0⟧	⟦2⟧	⟦1⟧	⟦3⟧		...		⟦2⟧	⟦0⟧	⟦2⟧	⟦0⟧

$$s(\llbracket x \rrbracket) = \begin{cases} \llbracket 0 \rrbracket & x = 0 \\ \llbracket 1 \rrbracket & 1 \leq x \leq v \end{cases}$$

Output:

EBF'

⟦0⟧	⟦1⟧	⟦1⟧	⟦1⟧		...		⟦1⟧	⟦0⟧	⟦1⟧	⟦0⟧

Concurrent Ballot Authorization

Encrypted Bloom Filter: Query

Input: a, EBF

EBF

⟦0⟧	⟦1⟧	⟦1⟧	⟦1⟧		...		⟦1⟧	⟦0⟧	⟦1⟧	⟦0⟧
		$\uparrow$ $hash_i(a)$					$\uparrow$ $hash_j(a)$			

$$[\![x]\!] = \Sigma \quad [\![1]\!] \quad \ldots \quad [\![1]\!]$$

$$f([\![x]\!]) = \begin{cases} [\![0]\!] & 0 \le x \le k-1 \\ [\![1]\!] & x = k \end{cases}$$

Output:

$$[\![t]\!] = f([\![x]\!])$$

Outline

Protocol

Setup:

- The authority runs the distributed key generation DKG and outputs
  - a description of G_q
  - generator $g \in G_q$
  - public key y
- The authority runs *Setup* from (m, k)-EBF
- Most of the Mix & Match tables can be pre-computed

Registration:

- The voter chooses a password $\hat{p}$ and processes it into credential $p = \mathrm{PBKDF}(\hat{p}, \mathit{VoterId})$
- The voter runs *Prepare*(g^p) to generate EBF_a
- The voter repeats this process α times with different passwords, used for a later cut-and-choose protocol (similar to Selections)
- After cut-and-choose the registrar publishes $\langle$VoterId, EBF'_a, $\pi'_a\rangle$ on the Roster
- After all voters have registered, the authority runs *Insert* over all EBFs on the Roster and finally *Flatten* to create the final EBF

Protocol

Casting:

- The voter encodes their password (real or panic) $\hat{p}$ into credential $p = \mathrm{PBKDF}(\hat{p}, \mathit{VoterId})$ and commits to it, g^p
- The voter encrypts their vote $[\![v]\!]$ and cast the ballot $\langle g^p, [\![\mathrm{v}]\!], \pi \rangle$
- The election authority post $\langle g^p, [\![\mathrm{v}]\!], \pi \rangle$ to the public list *AllVotes*

Protocol

Ballot Authorization:

- Upon receiving $\langle g^p, [\![v]\!], \pi\rangle$, the trustees first check $\pi \rightarrow$ on success $\langle g^p, [\![v]\!]\rangle$ is posted to the public list *ProvedVotes*
- The trustees mark previously cast ballots with the same value g^p as duplicates (on all lists)
- The trustees run $Query(g^p, EBF)$ to receive $[\![t]\!]$
- The trustees use Mix & Match to generate the "zeroing" function z:

$$z([\![t]\!],[\![v]\!]) = \begin{cases} [\![0]\!] & t = 0 \\ [\![v]\!] & t = 1 \end{cases}$$

- The trustees evaluate z: $[\![v']\!] = z([\![t]\!],[\![v]\!])$ and post $[\![v']\!]$ on the public list *ValidatedVotes*

Tallying:

- The trustees homomorphically sum all $[\![v']\!]_i$ on *ValidatedVotes* (except the ones marked as duplicates):

$$V = \Pi[\![\text{v'}]\!]_i = [\![\Sigma v'_i]\!]$$

- The trustees distributively decrypt V

Outline

Security Analysis

- Eligibility Verification *(Kremer et al. 2010)*
  (1) Each vote in the final tally was cast by a registered voter
  (2) There is at most one vote per voter
- Integrity
  (1) The final tally is the correct sum of eligible votes in the election
- Coercion-Resistant
  (1) The voter can always realize their voting intent
  (2) An adversary can not distinguish a fake credential from a real credential

Security and Performance

Security Analysis

- Eligibility Verification *(Kremer et al. 2010)*
  (1) Each vote in the final tally was cast by a registered voter
  (2) There is at most one vote per voter
- Integrity
  (1) The final tally is the correct sum of eligible votes in the election
- Coercion-Resistant
  (1) The voter can always realize their voting intent
  (2) An adversary can not distinguish a fake credential from a real credential

Security Analysis

- Eligibility Verification *(Kremer et al. 2010)*
  (1) Each vote in the final tally was cast by a registered voter
  (2) There is at most one vote per voter
- Integrity
  (1) The final tally is the correct sum of eligible votes in the election
- Coercion-Resistant
  (1) The voter can always realize their voting intent
  (2) An adversary can not distinguish a fake credential from a real credential

Security and Performance

Registration *(Before Election)*

	JCJ/Civitas	Selections	SHKS	Cobra
Voter	11	$(4\alpha - 1)$	11	$(4\alpha + 8)116V + 6$
Registrar	8	2α	8	$(4\alpha + 8)116V + 6$

Casting *(During Election)*

	JCJ/Civitas	Selections	SHKS	Cobra
Submit Ballot	$8C + 2$	$8C + 2$	$8C + 2$	$8C + 2$
Submit Cred.	3	$4\beta + 2$	3	2

Processing *(During Election)*

	JCJ/Civitas	Selections	SHKS	Cobra
Check Ballots	$(4C + 4)B$	$(4C + 4)B$	$(4C + 4)B$	$(4C + 4)B$
Ballot Authorization	0	$(4\beta)B$	0	$(280T + 12CT + 19T + 2)B$

Processing & Tallying *(After Election)*

	JCJ/Civitas	Selections	SHKS	Cobra
Ballot Authorization	$(6CT + 7VT + \frac{5}{2}T)B + (\frac{7}{2}T)B^2$	$(12CT + 7T)B$	$(12C\beta T + 7\beta T + 7T)B$	0
Tally Ballots	$3TC$	$3TC$	$3TC$	$3TC$

[Number of modular exponentiations assuming V registered voters, C candidates, B ballots cast, T trustees, α and β are system-specific parameters]

Security and Performance

	JCJ/Civitas	Selections	SHKS	Cobra
Registration *(Before Election)*				
Voter	11	39	11	55‘680‘006
Registrar	8	20	8	37‘120‘006
Casting *(During Election)*				
Submit Ballot	42	42	42	42
Submit Cred.	3	202	3	2
Processing *(During Election)*				
Check Ballots	240‘000	240‘000	240‘000	240‘000
Ballot Authorization	0	2‘000‘000	0	10‘790‘000
Processing & Tallying *(After Election)*				
Ballot Authorization	3‘000‘960‘000	2‘010‘000	100‘710‘000	0
Tally Ballots	45	45	45	45

[Number of modular exponentiations for an election with 5 candidates, 10‘000 registered voters, 20‘000 submitted ballots, 3 trustees, $\alpha = 10$, $\beta = 50$]

Conclusion and Questions