Attacks On Mixnets

Panagiotis Grontas

$\mu\Pi\lambda\forall$ - CoReLab Crypto Group

April 15, 2014

Overview

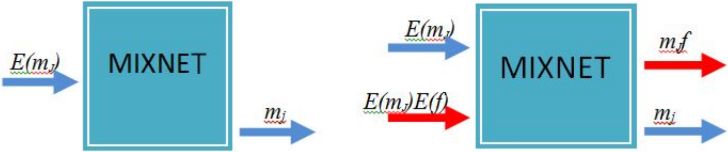
Attack Targets

- **Privacy** Expose the messages of one or more senders
- **Correctness** Alter the output of the mixnet
- **Robustness** Bypass honest mix servers, so they cannot detect/defeat any cheating attempt

Attack Options

- Corrupt One Or More Senders
- Corrupt One Or More Mix Servers
- Corrupt Both
- Exploit Protocol Weaknesses

Related input attack



Chaumian Mixnets I

- Active Attack
- Idea:Trace a message m using RSA multiplicative homomorphism

- Apply randomness of length b to message of length B
- Mixnet input: $\mathcal{E}(m) = (r \cdot 2^B + m)^e mod n$
- Mixnet output: m

Chaumian Mixnets II

The Attack

1. Choose a small factor f
2. For the message m to be tracked input to the mixnet the message $\mathcal{E}(m) \cdot \mathcal{E}(f) = (r \cdot 2^B + m)^e \cdot f^e mod n$
3. Mixnet operates as usual, considering input of the form: $(r^* \cdot 2^B + m^*)^e$
4. As a result:

$$(r^* \cdot 2^B + m^*) = (r \cdot 2^B + m) \cdot f \Rightarrow$$
$$(m^* - f \cdot m) \cdot 2^{-B} = (f \cdot r - r^*)$$

5. Fact 1:$m^*, m \in$ mix output
6. Fact 2:$r, r^* \in \{0, \cdots, 2^b - 1\} \Rightarrow (f \cdot r - r^*) \in \{2^{-b} + 1, \cdots, f \cdot (2^b - 1)\}$
7. For all output message pairs (m, m^*) check for which $(m^* - f \cdot m)$ has a value in $\{2^{-b} + 1, \cdots, f \cdot (2^b - 1)\}$

Reencryption Mixnets I

The Passive Attack

- ElGamal is not semantically secure for any prime p
- Only safe primes $p = 2q + 1$ must be used
- Choose the generator from subgroup $\mathbb{Z}_q$

Reencryption Mixnets II

The Active Attack - Decryption Mixnets

- Track input m_i for participant P_i:
- Initial Encryption $c_{i0} = (g^R, m_i \cdot (y_1, \cdots, y_k)^R)$
- Mix server j input: $c_{ij} = (g^{R'}, m_i \cdot (y_j, \cdots, y_k)^{R'}) = (t, u)$
- For some random x generate $c''_{ij} = (t^x, u^x) = (g^{R'x}, m_i^x \cdot (y_j, \cdots, y_k)^{R'x})$
- Output will contain both m_i^x, m_i
- Raise all output messages to the x and check for duplicates.

Reencryption Mixnets III

Remarks

- If there is a check on number of input items a colluding participant's message must be omitted
- Solution: Redundancy in messages in order to detect the attack
  - Increases Ciphertext Size
  - Does not work if the last mix server is corrupt, since it can replace the m_i^x with a correct looking message after the message correlation
- Detection: DDH Assumption

A single honest mix server can be overriden I

Attack on Privacy

- Break voter privacy even when there is only one honest mix server M_j
- No colluding voters are needed

- Each mix server
  - Proves that can decrypt
  - Reencrypts - Proves Reencryption
  - Shuffles - Proves Shuffle
- The attack happens before M_j performs the shuffle
- The first $j-1$ corrupt mix servers trace the message to be revealed
- The last $k-j$ servers reveal the message

A single honest mix server can be overriden II

Details

- Voter Alice sends to M_1 her input m as

$$(G_1, M_1) = (g^{r_0}, m \cdot (\prod_{t=1}^{k} y_t)^{r_0})$$

- Honest Mix Server receives:

$$(G_j, M_j) = (g^{r_0+R}, m \cdot (\prod_{t=j}^{k} y_t)^{r_0+R}) \text{ where } R = \sum_{t=1}^{j-1} r_t$$

- Honest Mix Server follows the protocols and reveals in order to prove that he can partially decrypt correctly.

$$H_j = G_j^{x_j} = (g^{r_0+R})^{x_j}$$

A single honest mix server can be overriden III

- The rest of the corrupted servers contribute their private keys x_j and compute:

$$\frac{M_j}{H_j \cdot \prod_{t=j+1}^{k} G_j^{x_t}} = \frac{m \cdot (\prod_{t=j}^{k} y_t)^{r_0+R}}{(g^{r_0+R})^{x_j} \cdot (g^{r_0+R})^{\sum_{t=j+1}^{k} x_t}} = m$$

A simple countermeasure

First shuffle then reencrypt

Break privacy when all servers are honest I

Requirements

Attacker can send 2 messages to the mixnet (by corrupting two voters)

- Target user Alice submits message m to mixnet
- $m \rightarrow_{E_1} \rightarrow (\mu_1, \mu_2) \rightarrow_H \rightarrow (\mu_1, \mu_2, w) \rightarrow_{E_2}$
- Randomly select γ, δ
- Calculate

$$\mu_1^\delta, \mu_2^\delta, w_\delta = H(\mu_1^\delta, \mu_2^\delta)$$

  and

$$\mu_1^\gamma, \mu_2^\gamma, w_\gamma = H(\mu_1^\gamma, \mu_2^\gamma)$$

- Calculate and submit

$$E(\mu_1^\delta), E(\mu_2^\delta), E(w_\delta) \text{ and } E(\mu_1^\gamma), E(\mu_2^\gamma), E(w_\gamma)$$

Break privacy when all servers are honest II

- Mixnet operates as usual and since all mix servers are honest removes both layers of encryption!
- Attacker raises outputs to $\frac{\delta}{\gamma}$

$$\textit{output} \rightarrow_{D_1} \begin{bmatrix} (u_1, v_1, w_1) \\ \cdots \\ \mu_1^{\delta}, \mu_2^{\delta}, w_{\delta} \\ \cdots \\ \mu_1, \mu_2, w \\ \cdots \\ \mu_1^{\gamma}, \mu_2^{\gamma}, w_{\gamma} \\ \cdots \\ (u_N, v_N, w_N) \end{bmatrix} \rightarrow_{D_2 \text{ to produce} L_k} \begin{bmatrix} m_1 \\ \cdots \\ m_x = m^{\delta} \\ \cdots \\ m \\ \cdots \\ \\ m_y = m^{\gamma} \\ \cdots \\ m_N \end{bmatrix} \rightarrow_{\text{raise to } \frac{\delta}{\gamma} \text{ to produce } L_k'} \begin{bmatrix} m_1^{\frac{\delta}{\gamma}} \\ \cdots \\ m_x^{\frac{\delta}{\gamma}} \\ \cdots \\ m^{\frac{\delta}{\gamma}} \\ \cdots \\ m_y^{\frac{\delta}{\gamma}} = m_x \\ \cdots \\ m_N^{\frac{\delta}{\gamma}} \end{bmatrix}$$

Break privacy when all servers are honest III

- Calculate $m_y^{\frac{1}{\gamma}} = m$
- Find index l of message m
- Find ciphertext of index l
- Search initial encryption for ciphertext corresponding to l
- Alice is revealed
- Remark: Can be generalised to break the privacy of s senders

Exploits

- Proof of ciphertext knowledge and not of message
- Same keys for both layers of encryption

Different Keys and Corrupt Mix Server I

Requirements

- Inner Encryption (Initial): E_{in}
- Outer Encryption (On triples): E_{out}
- 2 mix sessions with the same keys (unclear protocol assumption)
- First mix server is corrupted and is identified in the first mix session

Different Keys and Corrupt Mix Server II

First Mix Session

- Target user Alice with message m and user Bob with message m'
- Corrupted mix server swaps encryption of hashes

$$m \to E_{in} \to (u, v) \to H \to w \to E_{out} \to [E_{out}(u), E_{out}(v), E_{out}(w')] = \alpha$$

and

$$m' \to E_{in} \to (u', v') \to H \to w' \to E_{out} \to [E_{out}(u'), E_{out}(v'), E_{out}(w)] = \alpha'$$

- Protocol proceeds as usual
- Output contains two invalid tuples (u, v, w') and (u', v', w)
- Back-Tracing reveals first mix server as cheater **BUT**
- Links Alice to (u, v) and Bob to (u', v')

Different Keys and Corrupt Mix Server III

Second Mix Session - Decryption Oracle

- Relation attack on Alice's ciphertext (u, v)
- Randomly select γ, δ and Calculate

$$u^{\delta}, v^{\delta}, w_{\delta} = H(u^{\delta}, v^{\delta})$$

and

$$u^{\gamma}, v^{\gamma}, w_{\gamma} = H(u^{\gamma}, v^{\gamma})$$

- Calculate and submit

$$E_{out}(u^{\delta}), E_{out}(v^{\delta}), E_{out}(w_{\delta}) \text{ and } E_{out}(u^{\gamma}), E_{out}(v^{\gamma}), E_{out}(w_{\gamma})$$

- Mixnet operates as usual and since all mix servers are honest (now) removes both layers of encryption!
- Attacker raises outputs to $\frac{\delta}{\gamma}$ and correlates messages as before

Exploit: Proof of knowledge of ciphertext and not of message!

Attack On Privacy without corrupted users I

Requirements

- Honest Senders
- Last Mix Server M_k is corrupted
- **Objective:** Break privacy of all users

Attack On Privacy without corrupted users II

Actions of M_k

1. Generate the tuple $(\alpha', b', \cdots, f') = (\frac{\alpha_{k-1}}{\alpha_0}, \frac{b_{k-1}}{b_0}, \cdots, \frac{f_{k-1}}{f_0})$ where $\alpha_j = \prod_{i=1}^{N} \alpha_{ji}$ the product of the first components of the N inputs of M_j
2. Generate the tuple $\alpha_1 = (\alpha' \cdot \alpha_{01}, \cdots, f' \cdot f_{01})$
3. Generate the list L'_{k-1} by retrieving L_0, the input list to the mixnet, and replacing the first message with α_1
4. Instead of processing the normal input list L_{k-1} process the list L'_{k-1}.
5. Mixnet output is a permutation and reencryption of L'_{k-1}, essentially of L_0
6. If M_k isn't caught then after decryption, privacy is broken!

Attack On Privacy without corrupted users III

Corrupted Server passes verification

- Proof of correct reencryption
  - The modification to the input list is *invisible*

$$\alpha'_{k-1} = \alpha' \alpha_{01} \cdot \prod_{i=2}^{N} \alpha_{0i} = \alpha' \cdot \prod_{i=1}^{N} \alpha_{0i} = \alpha' \cdot \alpha_0 = \frac{\alpha_{k-1}}{\alpha_0} \cdot \alpha_0 = \alpha_{k-1}$$

  - Subsequently M_k behaves honestly
- Invalid Triples Investigation
- M_k cannot correlate the real L_{k-1} with L_k but it is not necessary as:
  - All users are honest
  - All mix servers except M_k are honest
  - M_k does not corrupt inner triples
  - There are no invalid triples

Two corrupted mix servers I

Requirements

- Honest Senders
- First M_1 and Last M_k Mix Server is corrupted
- **Objective:** Break privacy of a particular user

Main idea

- Let P, Q the primes in the initialisation of the El Gamal Cryptosystem
- All ciphertexts should be in G_Q
- If no check is made, we can use elements in $\mathbb{Z}_P^* - G_Q$ to 'tag' messages and break privacy

Two corrupted mix servers II

The attack

- Corrupt mix server M_1 and select $\gamma \in_R \mathbb{Z}_P^* - G_Q$
- Compute $\alpha'_1 = (\gamma \cdot \alpha_{01}, \cdots, f_{01})$ where α_1 is the message of Alice (target) and $\alpha'_2 = (\gamma^{-1} \cdot \alpha_{02}, \cdots, f_{02})$ where α_2 is the message of another user Bob.
- Replace α_1 with α'_1 and α_2 with α'_2 and proceed as usual
- Corrupt mix server M_{k-1} receives input L_{k-1} and processes it as usual
- Output would be $L_k = \{\beta_i\}_{i=1}^N = \{(\cdot \alpha_{ki}, \cdots, f_{ki})\}_{i=1}^N$
- Search for l, l' st: $\alpha_{kl}^Q = \gamma^Q$, $\alpha_{kl'}^Q = \gamma^{-Q}$
- Replace β_l with $(\gamma^{-1}\alpha_{kl}, \cdots, f_{kl})$ and $\beta_{l'}$ with $(\gamma_{kl'}^{\alpha}, \cdots, f_{kl'})$
- Output the result. Decryption takes place. Alice's message is m_l

Two corrupted mix servers III

Justification

- Attack works because: $\forall b \in G_Q : b^Q = 1$. As a result $\alpha_{kl}^Q = \gamma^Q (g^r \alpha_{01})^Q = \gamma^Q$
- Attack passes validations since only the a-component of the tuple is changed and
- Products are left unchanged as:

$$\alpha_1 = \gamma \alpha_{01} \gamma^{-1} \alpha_{02} \prod_{i=3}^{N} \alpha_{0i} = \prod_{i=1}^{N} \alpha_{0i} = \ \alpha_1$$

and

$$\alpha_k = \gamma \alpha_{kl'} \gamma^{-1} \alpha_{kl} \prod_{i \neq l, l'}^{N} \alpha_{0i} = \prod_{i=1}^{N} \alpha_{ki} = \ \alpha_k$$

Delayed Effect Attack I

Requirements

- Corrupted First Mix Server
- Even Number Of Corrupted Senders

Main idea

The adversary can cancel out the cheating based on an event after the vote casting phase. **Example:** Double elections. The adversary might inject votes in the second election, but depending on the first outcome he might cancel them out.

- Casted votes are $x_i = (\alpha_i, b_i, c_i, d_i, e_i, f_i) = (E(u_i), E(v_i), E(w_i)), i = 1, \cdots, N$

Delayed Effect Attack II

- Select $z \in G_Q$ and two voters k, l and compute $x'_k = (\alpha_i, b_i, c_i, zd_i, e_i, f_i)$ and $x'_l = (\alpha_i, b_i, c_i, z^{-1}d_i, e_i, f_i)$
- Product is maintained
- Replace x_k, x_l
- M_1 Might choose to correct them or not

Pfitzmann Attacks work with probability $\frac{1}{2}$ I

Requirements

- Corrupted Mix Servers: First
- Voters: One / None

Trace a single message

- Submitted ciphertext is c
- Corrupted mix server chooses an integer δ
- Replace an output by c^{δ}
- This is not detected during verification with probability $\frac{1}{2}$
- Search the output for messages $m, m*$ st $m* = m^{\delta}$
- Ciphertext c was encryption of m

Pfitzmann Attacks work with probability $\frac{1}{2}$ II

Trace multiple messages

- Submitted ciphertexts is $\{c_i\}_{i=1}^{s}$
- Corrupted mix server chooses integers$\{\delta_i\}_{i=1}^{s}$
- Replace an output by $\prod_{i=1}^{s} c_i^{\delta_i}$
- This is not detected during verification with probability $\frac{1}{2}$
- Search the output for messages $\{m_i\}_{i=1}^{s}, m*$ st $m* = \prod_{i=1}^{s} m_i^{\delta_i}$
- Correspondence for m_i is revealed

On duplicate removal I

Requirements

- Chaumian Mixnet
- Corrupted Mix Servers: First and Last
- Corrupted Voters: $\frac{s(s+1)}{2}$

Actions of first mix server

- Target ciphertexts $\{c_i\}_{i=1}^{s}$
- First mix server removes first layer of encryption
- For c_{i1} make i independent encryptions using his public key
- Replace $\frac{s(s+1)}{2}$ ciphertexts with the duplicate encryptions
- These encryptions replace the original messages

On duplicate removal II

Actions of last mix server

- Input contains duplicate items ($i+1$ for ciphetext i)
- Decrypt using public key as always
- Identify correspondence based on the number of duplicates
- $s + \frac{s(s+1)}{2} \leq N$
- The privacy of $O(\sqrt{N})$ voters can be broken

A simple countermeasure

Remove duplicates everywhere

Commitments must be valid permutations I

Requirements

- Reencryption Mixnet
- Corrupted Mix Servers: First and Second (operated by a single entity)
- Corrupted Voters: 2

Target: Break Privacy Undetected

Initialisation

- Submitted ciphertexts is $\{c_i\}_{i=1}^{s}$
- Attacker chooses integers $\{\delta_i\}_{i=1}^{s}$
- Calculate $c = \prod_{i=1}^{s} c_i^{\delta_i}$
- Corrupted senders posts 2 messages that are reencryption of each other $c_{i_1,0}, c_{i_2,0}$

Commitments must be valid permutations II

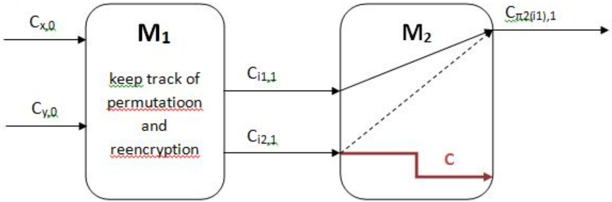
M_1

- Keep track of corrupted messages permutations
- Keep track of reencryption randomness

M_2

- Commit c_{i_1}, c_{i_2} to same output $\pi_2(i_1)$.
- Replace output $\pi_2(i_2)$ with c

Commitments must be valid permutations III



After mixing

- Search the output for messages $\{m_i\}_{i=1}^{s}, m*$ st $m* = \prod_{i=1}^{s} m_i^{\delta_i}$
- Correspondence for m_i is revealed without detection

Commitments must be valid permutations IV

Attack goes undetected

- Both commitments verify since c_{i_1}, c_{i_2} are reencryptions of the same message
- M_2 can provide the randomness to prove it

Generalisation

- $r + 1$ reeencryptions of a message
- $r + 1$ false commitments
- $r \cdot s$ messages

Commitments must be valid permutations V

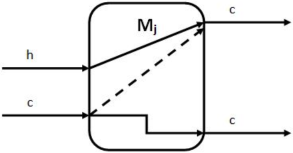
Rig an election undetected

- Corrupt the first mix server and the first sender
- First sender sends m as plaintext $c_{10} = Enc(m)$
- Replace all the ciphertexts with c_{10}
- Corrupted mix server commits all outputs to 1
- Corrupted Mix Server can pass verification
- **Realistic scenario:** Replace the actual outputs with the distribution of votes that he likes

Commitments must be valid permutations VI

Unopened Commitments

- Unopened commitments might not be consistent with a permutation
- Replace a message sent by an honest sender with a cheating message
- Commit to the same output
- Detection: Open both commitments
- Detection probability $\frac{1}{4}$
- Repeat t times with success probability $(\frac{3}{4})^t$



The Scytl Mixnet [AC10] I

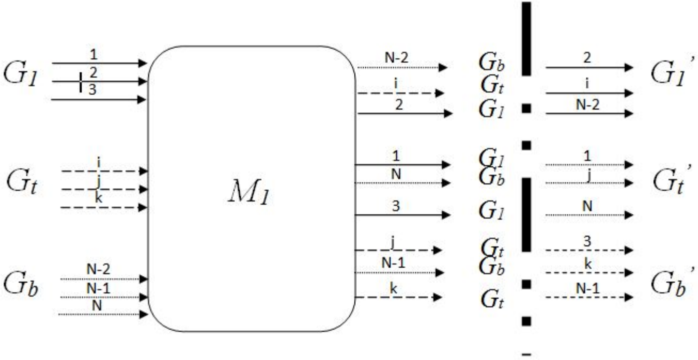
- Proposed by Puiggali and Castello in 2010
- Commercial Use from Scytl
- Part of the Norwegian 2011 municipal elections
- Operation based on
    - Randomised Partial Checking (RPC)
    - Almost Entirely Correct Mixing
    - Optimised Mixing
- Reencryption based mixnet
- Main Idea
    - Verification occurs after operation and before decryption
    - Inputs and outputs are split into groups
    - Group membership is revealed for all votes
    - Verification by comparing $\prod \textit{inputs}$ with $\prod \textit{outputs}$ for each group
    - Verification is done in Zero-Knowledge

The Scytl Mixnet [AC10] II

Verification

- Verifier groups encrypted input
- For each output the prover reveals the input group
- Verifier calculates:
    - **Input Integrity Proof:** Products per group of input votes
    - **Output Integrity Proof:** Products per input group of output proofs
- ZK Proof that Output Integrity Proof is reencryption of Input Integrity Proof
- Remarks
    - Group size l is constant
    - First grouping is random
    - Regrouping for M_j
        - Sort outputs of M_{j-1} by group index
        - Input group $\#1$ Receive *first* item of each output group
        - Input group $\#2$ Receive *second* item of each output group ...

The Scytl Mixnet [AC10] III



Basic Pfitzmann Attack

- **Target:** privacy of s voters with inputs $c_1, ..., c_s$
- **Requirements**: Corrupt 2 voters with inputs c_{01}, c_{02} and the first mix server
- Select s random exponents $\delta_1, ..., \delta_s$
- Calculate $u_1 = \prod_{i=1}^{s} c_i^{\delta_i}$ and $u_2 = \frac{c_{01} \cdot c_{02}}{u_1}$
- Replace c_{01}, c_{02} with u_1, u_2
- **Remark:** Product is preserved: $c_{01} \cdot c_{02} = u_1 \cdot u_2$
- In order to pass verifcation u_1, u_2 must belong to the same block
- **Success Probability:** $\frac{1}{b}$
- After decryption, search $s + 1$ messages such that $m = \prod_{i=1}^{s} m_i^{\delta_i}$

Undetected Pfitzmann Attack I

- Target privacy of s voters
- 1 corrupted mix server
- B corrupted voters
- The corrupted messages are reencryptions of each other
- The corrupted mix server can handle them without affecting the verification process

The birthday paradox

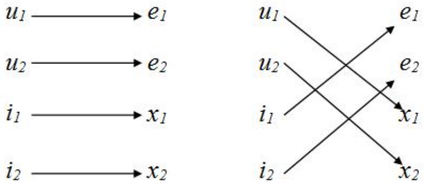
Pick s elements of a multiset with b elements.
Collision Probability $P(s, b) \geq 1 - e^{\frac{-s^2}{2 \cdot b}}$

- Very high probability of success with few corrupted voters
- For $B = 3 \cdot \sqrt{b}$ corrupted voters 98% chance of success.

Undetected Pfitzmann Attack II

- B corrupted messages
- u_1, u_2 the 'malicious messages', $u_1 = \prod_{i=1}^{s} c_i^{\delta_i}$ and $u_2 = \frac{c_{01} \cdot c_{02}}{u_1}$
- u_1, u_2 must end up in the same input block
- Say that u_1, u_2 exit the mix server at positions e_1, e_2
- Let i_1, i_2 two corrupted messages that are indeed in the same input group
- Say that i_1, i_2 exit at positions x_1, x_2
- The corrupted mix server can swap u's with the i's to put them in the same block

Undetected Pfitzmann Attack III



- Swapping can go undetected wvhp
- Explanation
  - The corrupted mix server must prove that $x_1 \cdot x_2 = ReEnc(i_1 \cdot i_2)$
  - But $ReEnc(i_1 \cdot i_2) = ReEnc(c_{01} \cdot c_{02}) = ReEnc(u_1 \cdot u_2)$
  - And of course by hushing up the replacement the other 'processing' can be proved as well $e_1 = ReEnc(u_1) = ReEnc(i_1)$ and $e_2 = ReEnc(u_2) = ReEnc(i_2)$
- *Probability of success = Probability 2 corrupted messages end up in the same block*

Attack On Correctness I

Overview

- 1 corrupted mix server
- $l \geq b$
- **Target**: Replace $R = \frac{1}{3}\sqrt{b} - 1$ votes without detection

- Corrupted mix server M_j replaces first R+1 votes
- The first corrupted votes $c_{j-1,1} \cdots c_{j-1,R}$ get replaced by 'malicious' messages $u_1, \cdots u_R$
- In order to maintain the product the vote $c_{j-1,R+1}$ gets replaced by $\frac{\prod_{i=1}^{R+1} c_{j-1,i}}{\prod_{i=1}^{R} u_j}$

Attack On Correctness II

- The replaced list L'_{j-1} gets reencrypted and shuffled.
- Probability that 2 integers in $\{1, R+1\}$are in the same block is 0.05
- With very high probability all such integers are in different blocks
- By way of output partitioning all end up in the same output, with very highy probability.
- The proof is valid both for L_{j-1} and L'_{j-1}.

References I

Jordi Puiggalí Allepuz and Sandra Guasch Castelló.
Universally verifiable efficient re-encryption mixnet.
In *Electronic Voting*, pages 241–254, 2010.

B. Adida.
Advances in cryptographic voting systems.
PhD thesis, Massachusetts Institute of Technology, 2006.

Shahram Khazaei, Björn Terelius, and Douglas Wikström.
Cryptanalysis of a universally verifiable efficient re-encryption mixnet.
In *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'12, pages 7–7, Berkeley, CA, USA, 2012. USENIX Association.

References II

Shahram Khazaei and Douglas Wikström.

Randomized partial checking revisited.

Technical report, Cryptology ePrint Archive, Report 2012/063, 2 012. http://eprint. iacr. org, 2012.

M. Michels and P. Horster.

Some remarks on a receipt-free and universally verifiable mix-type voting scheme.

In *Advances in Cryptology–ASIACRYPT'96*, pages 125–132. Springer, 1996.

B. Pfitzmann.

Breaking an efficient anonymous channel.

In *Advances in Cryptology–EUROCRYPT'94*, pages 332–340. Springer, 1995.

References III

B. Pfitzmann and A. Pfitzmann.

How to break the direct rsa-implementation of mixes.

In *Advances in Cryptology—EUROCRYPT'89*, pages 373–381. Springer, 1990.

Douglas Wikström.

Five practical attacks for “optimistic mixing for exit-polls”.

In *Selected Areas in Cryptography*, pages 160–174. Springer, 2004.