Introduction to e-Voting Bulletin Boards

Subodh Sharma

Subhashis Banerjee

IIT Delhi, Computer Science Department

Motivations

Motivations

- Requirement to publish information concerning all the votes cast for verifying inclusion and tallying

Motivations

Motivations

- Requirement to publish information concerning all the votes cast for verifying inclusion and tallying
- Voters and other external parties are able to check the published information and challenge the election

Motivations

Motivations

- Requirement to publish information concerning all the votes cast for verifying inclusion and tallying
- Voters and other external parties are able to check the published information and challenge the election
- Other motivations:
  - Auctions: Bidder places a bid at a particular time based on current bid and receive proof of receipt. The auctioneer must publish a proof that the sequence of bids has not been manipulated.

Motivations

Motivations

- Requirement to publish information concerning all the votes cast for verifying inclusion and tallying
- Voters and other external parties are able to check the published information and challenge the election
- Other motivations:
  - Auctions: Bidder places a bid at a particular time based on current bid and receive proof of receipt. The auctioneer must publish a proof that the sequence of bids has not been manipulated.
  - System logs: It might be useful to have a verifiable online log of the activities of a distrobuted system. Eg: IP infringement cases requiring verifiability of git server logs.

Key Security Requirements

- Once items are the on Bulletin Board (BB), they cannot be removed. That is, the board is *append-only*.

Key Security Requirements

- Once items are the on Bulletin Board (BB), they cannot be removed. That is, the board is *append-only*.
- The BB provides a consistent view to all the viewers

Key Security Requirements

- Once items are the on Bulletin Board (BB), they cannot be removed. That is, the board is *append-only*.
- The BB provides a consistent view to all the viewers
- The BB must accept submissions only from accredited writers
- Others should be able to verify the origin of the message when *Writers* write to the BB.

Key Security Requirements

- Once items are the on Bulletin Board (BB), they cannot be removed. That is, the board is *append-only*.
- The BB provides a consistent view to all the viewers
- The BB must accept submissions only from accredited writers
- Others should be able to verify the origin of the message when *Writers* write to the BB.
- Readers should also be able to determine that the writer intended the message to be published with the stated timestamp and at this particular point in the BB's sequence of messages

Key Security Requirements

- Once items are the on Bulletin Board (BB), they cannot be removed. That is, the board is *append-only*.
- The BB provides a consistent view to all the viewers
- The BB must accept submissions only from accredited writers
- Others should be able to verify the origin of the message when *Writers* write to the BB.
- Readers should also be able to determine that the writer intended the message to be published with the stated timestamp and at this particular point in the BB's sequence of messages
- Even if there is a collusion among the writer and the BB, we should still have protection against insertion, deletion or alteration of messages

A Note on Timeliness of Acceptance

Problem: Networks are asynchronous

A reader checks the BB at time T. Subsequently, she discovers a new message published with a timestamp $T' < T$. It is legitimately possible that the writer sent the message at $T - \delta$ and the message was in flight when the reader checked.

Solution:

A Note on Timeliness of Acceptance

Problem: Networks are asynchronous

A reader checks the BB at time T. Subsequently, she discovers a new message published with a timestamp $T' < T$. It is legitimately possible that the writer sent the message at $T - \delta$ and the message was in flight when the reader checked.

Solution:

- Introduce a security parameter ϵ.

A Note on Timeliness of Acceptance

Problem: Networks are asynchronous

A reader checks the BB at time T. Subsequently, she discovers a new message published with a timestamp $T' < T$. It is legitimately possible that the writer sent the message at $T - \delta$ and the message was in flight when the reader checked.

Solution:

- Introduce a security parameter ϵ.
- If whenever a reader views the BB at time T and a new message is subsequently published with a claimed publication timestamp earlier thatn $T - \epsilon$, then the reader can prove that the board has been corrupted.

A Note on Timeliness of Acceptance

Problem: Networks are asynchronous

A reader checks the BB at time T. Subsequently, she discovers a new message published with a timestamp $T' < T$. It is legitimately possible that the writer sent the message at $T - \delta$ and the message was in flight when the reader checked.

Solution:

- Introduce a security parameter ϵ.
- If whenever a reader views the BB at time T and a new message is subsequently published with a claimed publication timestamp earlier thatn $T - \epsilon$, then the reader can prove that the board has been corrupted.

Choice of ϵ is important

If ϵ is large, then the BB will make fewer message rejections. However, this increases the extent to which the BB can delay the decision on whether to publish a particular message.

Competing Message? Early Rejection

- Suppose writer W at time T attempts to write a message m as message $\lambda + 1$.

Competing Message? Early Rejection

- Suppose writer W at time T attempts to write a message m as message $\lambda + 1$.
- Suppose writer W' at time T' attempts to write a message m' as message $\lambda + 1$.

Competing Message? Early Rejection

- Suppose writer W at time T attempts to write a message m as message $\lambda + 1$.
- Suppose writer W' at time T' attempts to write a message m' as message $\lambda + 1$.
- Then the later message wins. That is, whenever $T + \epsilon < T'$ and m is accepted in the BB, then W' can prove that the board is corrupted.

Competing Message? Early Rejection

- Suppose writer W at time T attempts to write a message m as message $\lambda + 1$.
- Suppose writer W' at time T' attempts to write a message m' as message $\lambda + 1$.
- Then the later message wins. That is, whenever $T + \epsilon < T'$ and m is accepted in the BB, then W' can prove that the board is corrupted.

Practical reason: Early rejection prevents the BB from collecting a pool of next messages from writers and delay its decision over which message to publish until it has received a favorable one.

Structure of the BB

- $\langle BB_n \rangle$ is a sequence where each $\langle BB_i \rangle = \langle m_i, T_i, W_i, H_i, WSg_i, BSg_i \rangle$
  - m_i is a message
  - T_i is a timestamp
  - W_i is the id of the writer
  - H_i is the hash
  - WSg_i, BSg_i are the writer's and the board's commitment, respectively.

Structure of the BB

- $\langle BB_n \rangle$ is a sequence where each $\langle BB_i \rangle = \langle m_i, T_i, W_i, H_i, WSg_i, BSg_i \rangle$
  - m_i is a message
  - T_i is a timestamp
  - W_i is the id of the writer
  - H_i is the hash
  - WSg_i, BSg_i are the writer's and the board's commitment, respectively.

Consistent History

- $H_i = H(m_i.T_i.W_i.H_{i-1})$ where $H_0 = 0$ [HashChains]

Structure of the BB

- $\langle BB_n \rangle$ is a sequence where each $\langle BB_i \rangle = \langle m_i, T_i, W_i, H_i, WSg_i, BSg_i \rangle$
  - m_i is a message
  - T_i is a timestamp
  - W_i is the id of the writer
  - H_i is the hash
  - WSg_i, BSg_i are the writer's and the board's commitment, respectively.

Consistent History

- $H_i = H(m_i.T_i.W_i.H_{i-1})$ where $H_0 = 0$ [HashChains]
- $WSg_i = S_{W_i}(H_i)$ and $BSg_i = S_B(WSg_i.T_i')$ where T_i' is the BB's timestamp at the time of signing.

Structure of the BB

- $\langle BB_n \rangle$ is a sequence where each $\langle BB_i \rangle = \langle m_i, T_i, W_i, H_i, WSg_i, BSg_i \rangle$
  - m_i is a message
  - T_i is a timestamp
  - W_i is the id of the writer
  - H_i is the hash
  - WSg_i, BSg_i are the writer's and the board's commitment, respectively.

Consistent History

- $H_i = H(m_i.T_i.W_i.H_{i-1})$ where $H_0 = 0$ [HashChains]
- $WSg_i = S_{W_i}(H_i)$ and $BSg_i = S_B(WSg_i.T_i')$ where T_i' is the BB's timestamp at the time of signing.
- $T_i \leq T_i' < T_i + \epsilon$

Property of Histories

Lemma1

If a history is consistent, then any prefix of history is also consistent.

Implications: Given a hash chain $x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_n$

- Easy to evaluate the hashchain given x_0.
- Tampering of x_i would require tampering of all $x_j, j < i$. Thus, longer the chain higher the security.
- One can produce many one-time keys from a single key (See Leslie Lamport's storage-efficient authentication scheme in an insecure environment).
- In addition to untamperabilty, also provides chronology of data's existence.

Reading Protocol

Whenever anything is read from the BB, the board returns a signed and dated copy of the current state hash:

$$M1 : BB \Rightarrow R : \langle BB_n \rangle . S_B(H_\lambda . T_B)$$

Where H_λ is the state hash of the last state of $\langle BB_n \rangle$.

Reading Protocol

Whenever anything is read from the BB, the board returns a signed and dated copy of the current state hash:

$$M1 : BB \Rightarrow R : \langle BB_n \rangle . S_B(H_\lambda . T_B)$$

Where H_λ is the state hash of the last state of $\langle BB_n \rangle$.

Reason

This makes it impossible for the BB to subsequently change the contents on the BB before this point.

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash
- $M2 : W \Rightarrow BB : m.T.W.H.S_W(H)$, where $H = H(m.T.W.H_\lambda)$ i.e., sending the message to the BB

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash
- $M2 : W \Rightarrow BB : m.T.W.H.S_W(H)$, where $H = H(m.T.W.H_\lambda)$ i.e., sending the message to the BB
- $M3 : BB \Rightarrow W : S_B(S_W(H).T')$, i.e., getting a receipt from the BB.

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash
- $M2 : W \Rightarrow BB : m.T.W.H.S_W(H)$, where $H = H(m.T.W.H_\lambda)$ i.e., sending the message to the BB
- $M3 : BB \Rightarrow W : S_B(S_W(H).T')$, i.e., getting a receipt from the BB.

- In step 1, the writer can reject the message if T_B is more than ϵ old

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash
- $M2 : W \Rightarrow BB : m.T.W.H.S_W(H)$, where $H = H(m.T.W.H_\lambda)$ i.e., sending the message to the BB
- $M3 : BB \Rightarrow W : S_B(S_W(H).T')$, i.e., getting a receipt from the BB.

- In step 1, the writer can reject the message if T_B is more than ϵ old
- In step 2, the BB rejects the message if $T - T_B > \epsilon$. The BB checks and obtains proof that the writer did write the message; the writer also commits that she intended the message appear as the next message in the sequence.

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash
- $M2 : W \Rightarrow BB : m.T.W.H.S_W(H)$, where $H = H(m.T.W.H_\lambda)$ i.e., sending the message to the BB
- $M3 : BB \Rightarrow W : S_B(S_W(H).T')$, i.e., getting a receipt from the BB.

- In step 1, the writer can reject the message if T_B is more than ϵ old
- In step 2, the BB rejects the message if $T - T_B > \epsilon$. The BB checks and obtains proof that the writer did write the message; the writer also commits that she intended the message appear as the next message in the sequence.
- In step 3, the writer will reject if $T' - T > \epsilon$. The writer also gets a proof that the accepted message is appearing next on the BB.

Writing Protocol

Protocol

- $M1 : BB \Rightarrow W : S_B(H_\lambda.T_B)$, ie., getting the current state hash
- $M2 : W \Rightarrow BB : m.T.W.H.S_W(H)$, where $H = H(m.T.W.H_\lambda)$ i.e., sending the message to the BB
- $M3 : BB \Rightarrow W : S_B(S_W(H).T')$, i.e., getting a receipt from the BB.

- In step 1, the writer can reject the message if T_B is more than ϵ old
- In step 2, the BB rejects the message if $T - T_B > \epsilon$. The BB checks and obtains proof that the writer did write the message; the writer also commits that she intended the message appear as the next message in the sequence.
- In step 3, the writer will reject if $T' - T > \epsilon$. The writer also gets a proof that the accepted message is appearing next on the BB.
- Finally, BB publishes: $\langle m.T.W.H.S_W(H).S_B(S_W(H), T') \rangle$

Theorems on Correctness

Consistent History

Any reader has enough information to check that BB history is consistent.

Untamperability

Thm1: The BB has unalterable history.

Theorems on Correctness

Consistent History

Any reader has enough information to check that BB history is consistent.

Untamperability

Thm1: The BB has unalterable history.

Certified Publishing

The BB has certified publishing: provides a proof of who wrote the message and order of the message in the sequence for each message.

Theorems on Correctness

Consistent History

Any reader has enough information to check that BB history is consistent.

Untamperability

Thm1: The BB has unalterable history.

Certified Publishing

The BB has certified publishing: provides a proof of who wrote the message and order of the message in the sequence for each message.

Timeliness of acceptance

The BB has *timely* publication.

Timely Publication

- R at T gets $S_B(H_\lambda, T_B), T_B - T < \epsilon$

Timely Publication

- R at T gets $S_B(H_\lambda, T_B), T_B - T < \epsilon$
- BB subsequently publishes a message at T_0 where $T_0 + \epsilon < T$

Timely Publication

- R at T gets $S_B(H_\lambda, T_B), T_B - T < \epsilon$
- BB subsequently publishes a message at T_0 where $T_0 + \epsilon < T$
- $BSg = S_B(S_W(H(m.T_0.H').T_0'))$ where $T_0' < T_0 + \epsilon$. But now $T_0' < T_0 + \epsilon < T < T_B$.

Timely Publication

- R at T gets $S_B(H_\lambda, T_B), T_B - T < \epsilon$
- BB subsequently publishes a message at T_0 where $T_0 + \epsilon < T$
- $BSg = S_B(S_W(H(m.T_0.H').T_0'))$ where $T_0' < T_0 + \epsilon$. But now $T_0' < T_0 + \epsilon < T < T_B$.
- The above implies that the history at T_B must have included the new entry in it.

What is not guaranteed?

- Livness problem: the bulletin board suffers from a catastrophic failure or compromised losing data and unable to accept read/write requests.

Solution: Distributed and *replicated* bulletin board!
A whole set of new problems arise: *distributed consistent view*, *Consensus and Byzantine adversaries* etc.

Threshold Cryptography Scheme

Requirement for the scheme to work: As long as some threshold set k out of n BB peers survive and function correctly, the integrity of the election(or the application) can be guaranteed by the collective.

Threshold Cryptography Scheme

Requirement for the scheme to work: As long as some threshold set k out of n BB peers survive and function correctly, the integrity of the election(or the application) can be guaranteed by the collective.

- Assume some threshold cryptogrphy scheme (such as ElGamal, Paillier) has been agreed upon
- Each peer has been given a secret share of a threshold signing key.
- Key requirement: Private key can be slit to n parts in such a way that a threshold subset of k key holders can cooperate to perform decryption but $k-1$ holders cannot!

Synchronized Distributed history

- S_B is split among n peers
- W sends a message to a peer of his choice.
- The peer forms a threshold set of peers who can sign the receipt.
- The signed receipt is sent to W and the k peers update their local copy of BB and bcast the update message to $n - k$ peers for publication on their boards too.

Avoiding Diverging Histories

- For peers to be synchronised, concurrent writes must be avoided.

Avoiding Diverging Histories

- For peers to be synchronised, concurrent writes must be avoided.
- Simple solution: Make the threshold set k s.t. $2k > n$.

Avoiding Diverging Histories

- For peers to be synchronised, concurrent writes must be avoided.
- Simple solution: Make the threshold set k s.t. $2k > n$.
- Thus, no peer can be part of two signing sets at the same time.

Avoiding Diverging Histories

- For peers to be synchronised, concurrent writes must be avoided.
- Simple solution: Make the threshold set k s.t. $2k > n$.
- Thus, no peer can be part of two signing sets at the same time.
- What happens if one of $n - k$ peers refuse to publish a certified message?

References

- The append-only web bulletin board. James Heather and David Lundin. 2008
- A peered bulletin board for robust use in verifiable voting systems. Chris Culnane and Steve Schneider. 2018.