Design and Development of Voting Data Security for Electronic Voting (E-Voting)

Supeno Djanali*, Baskoro Adi Pratomo, Karsono Puguh Nindyo Cipto, Astandro Koesriputranto, Hudan Studiawan
Department of Informatics, Faculty of Information Technology,
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
*supeno@its.ac.id

***Abstract*—Several countries has moved to e-voting because of many reasons. By using e-voting, we have to pay more attention to protect voter's choice confidentiality and secure voting data.**

This paper proposed a combination of SHA256, digital signature, and RSA asymmetric encryption applied to database as a storage system and protocol to secure our message exchange. The proposed method was implemented as a web service and a simple client application placed in voting place is created as well. The whole system was designed to be as close as the usual parliament voting of a city in Indonesia. Therefore there are several level that need to be considered, i.e. voting place, districts, city, state, country. But we only implemented until the districts level, since the process for the next levels is just the same.

Voting places, as the most bottom level, use the client application to count the voting data, then after some checking and protecting confidentiality and integrity of the data, they are sent to the upper level. At the next level, those data will be checked, signed, and sent to upper level until it reaches country level.

From our experiment, our proposed method was able to handle many kinds of attack, such as replay attack, packet sniffing, and falsifying voting result, moreover privacy of the voter is well protected while the integrity of the data can be protected as well. Therefore we believe that our system will be able to replace conventional voting to a better system.

Keywords*—*e-voting, voter's privacy, message exchange security

I. INTRODUCTION

Storage design of e-voting system plays important roles in a real world implementation. Because how we store the voting data is the key to protect privacy and data integrity, including total number of vote for each candidate. By protecting privacy, we mean that no one would be able to see who were chosen by the voter, even the administrator of the system. On the contrary, we need to make sure that the vote is true, really come from legitimate voter and cannot be changed on its way to the country level. Total number of vote for each candidate must be protected as well to prevent fraud like double vote or improper addition to some candidates.

In a parliament traditional voting system in Indonesia, each voter come to the voting place that is determined in advance. At the voting place, there must be committees and witnesses who make sure that the voting goes well. When the voter comes to voting place, he/she must show his/her identity card to show that he/she is a legitimate voter. Then the committees usually give them three kind of ballot and they have to put a hole inside the party's logo or the candidate's name. After the ballots is collected in a secure box, they will be counted.

This counting process usually takes a long time. From the lowest level, the voting place, counting process could take several hours and it is hard to determine whether the result is true or not. In each step of sending data to the next level could take several days. Full usage of electronic voting system can speed up this process, but if e-voting system is suddenly used, our infrastructure is not ready yet. Therefore we proposed an e-voting system that works as close as the traditional one, but can ensure voter privacy and integrity of the voting data.

The rest of this paper is structured as follows, section 2 covers related works in e-voting area. Section 3 shows our methodology and architecture. Result of our experiment is then explained in section 4. Lastly, we conclude our work on section 5.

II. RELATED WORKS

Rivest et.al introduced a method to encrypt data which can only be decrypted at specified time later [1]. This can be useful to encrypt some time sensitive data like bidding offer or electronic vote. They used a combination of public key encryption and hash function to enable decryption only at certain time. But this method does not cover communication between client and server and how to store votes in the database in a secure manner.

Another work on e-voting was done in [2]. To protect the confidentiality of the voters, they design a paper ballot that will be teared after people have given a vote. The teared paper ballot then can be used to count the voting result while maintaining voter privacy. Unlike the conventional paper ballot which always have parties and candidates printed on the same order, this method randomized the order, but still can be correctly counted.

Our main work is focus on the verifiability of e-voting vote count. There are some related research regarding this issues. The overall design of e-voting infrastructure was proposed in [3]. They built a working ecosystem to deploy a remote voting and ensure its security especially the verifiability to ensure the votes are valid and able to detect unauthorized one. The mechanism was to match several parts of the secure key in some servers. The operating servers should provide the same count of votes unless there was an anomaly to be checked.

The attack to the verifiability of vote data was given in [4]. The clash attack was simple since it exploited the voting machine to supply different votes from the same voter. The author provided the countermeasure by using the serial number on printed receipt to Wombat and Helios e-voting systems.

Another ballot integrity procedure was proposed by employing entanglement between two parties [5]. There were three phases included: initial, voting, and verification phase. Cortier et.al also wrote a formal model for both weak and strong verifiability [6]. They proof the proposed model to Helios-C (Helios with Credential) system. However, we propose another system to provide more secure ballot in e-voting environment built on top of our own system.

III. Methodology

This section talks about architecture and protocol design of our proposed e-voting system, as well as the database structure design. As written above, our proposed system was designed to be as close as the traditional e-voting system in Indonesia, therefore the architecture is more suitable for Indonesia, but it is still applicable in other countries with slight modification.

A. E-Voting Architecture

In Indonesia, there are several levels of counting process, from the lowest level, voting place, to the highest level, country level. In between, there are sub-district, city, and province. Each sub-district has several voting places and after each voting place has finished counting, they will send the result to corresponding sub-district. Then each sub-district will send them to the corresponding upper level.

The proposed system is designed to support any kind of voting methodology, from conventional method, which involving a paper ballot, scanned, and stored on the local server, to fully computerized e-voting using touch screens as a medium. The touch screen computers send the votes to a local server, which will be forwarded to the upper level server like the conventional method does. Intermediary server will keep forwarding the votes until it reaches the central server. In each level, there is an administrator who will check the incoming result and forward them to upper level server.

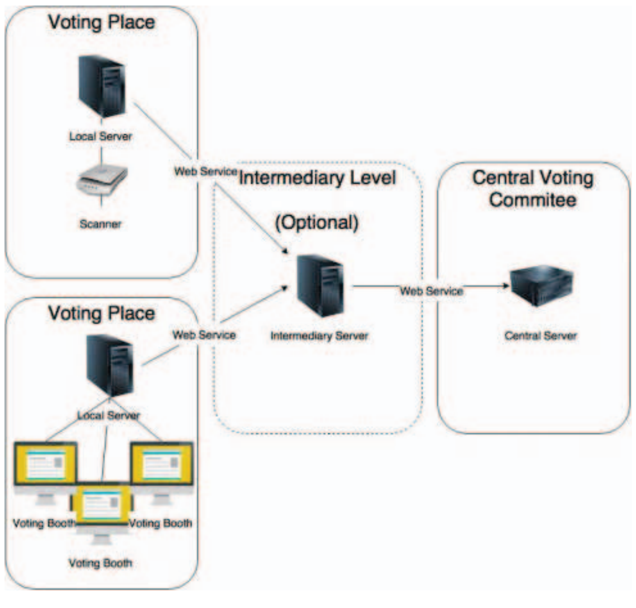


Fig. 1 Proposed Architecture

To ensure flexibility of client application, regardless of the techniques used behind the voting places application, from web based to desktop based application, whether using Java, C#, or other languages, we used a web service for communication between servers.

B. Flowchart

In general, the overall process of parliament member voting is people come to a voting place, show their identification (card, fingerprint, etc), vote with paper ballot or computerized booth, then at the end of allocated voting time, the committees count and send the results to the upper level. We won't explain further about the conventional business process, we rather focus on how to send the voting results to the upper level. Table 1 shows us the eight implemented web method and how information flows among those methods is shown on

First of all, authentication is needed to ensure that the results are coming from verified voting place; this includes sending the proper HELLO message to the server. After the local server, server in the voting place, is authenticated, it then sends each vote to the server. If anything happens while sending the votes, usually due to bad network connection, the failed votes are sent again next after the local server gives its session ID to the upper server. To prevent any man-in-the-middle attacks, connection between local server and the upper one is protected with HTTPS, since we used web service. Before sent, each vote is preprocessed to ensure the privacy and integrity by using encryption and digital-signature. Detail of this processing is explained further in the following paragraphs.

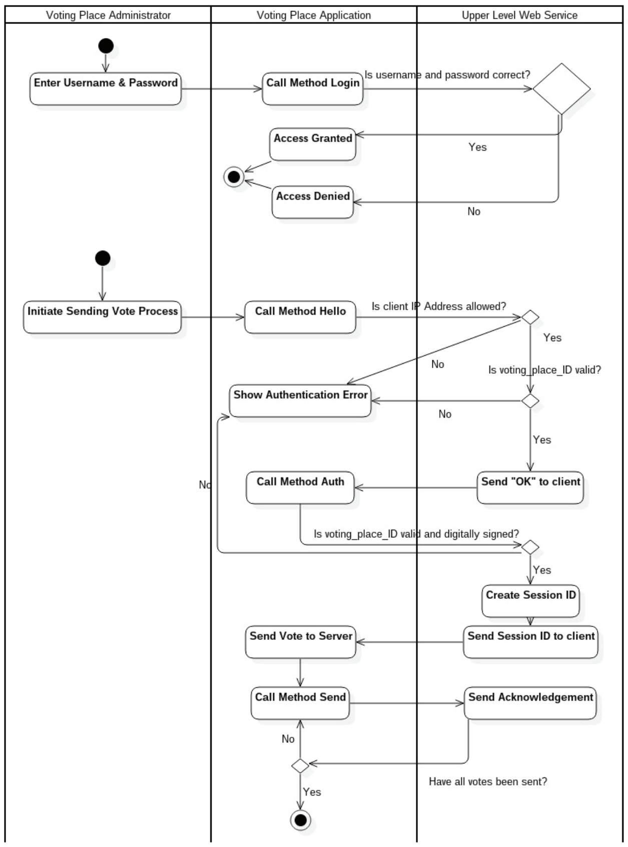

Voting Place Administrator
Voting Place Application
Upper Level Web Service
Enter Username & Password
Call Method Login
Is username and password correct?
Access Granted
Yes
Access Denied
No
Initiate Sending Vote Process
Call Method Hello
Is client IP Address allowed?
Yes
No
Is voting_place_ID valid?
Show Authentication Error
No
Yes
No
Call Method Auth
Send "OK" to client
Is voting_place_ID valid and digitally signed?
Yes
Create Session ID
Send Vote to Server
Send Session ID to client
Call Method Send
Send Acknowledgement
No
Yes
Have all votes been sent?

Fig. 2 Communication Flowchart

Table 1 List of Implemented Web Methods

Method Name	Explanation	Input Parameter	Return Value
Login	Used by the administrators at the voting places to log on to local system.	Username, Password	True/False
Hello	Used as a handshaking between local server and upper level.	H_{SHA256}(Voting_Place_ID)	"OK"/"Error"
Auth	Used to authenticate the local server to the upper level server. Return the Session ID to be used in the following communication messages.	H_{SHA256}(Voting_Place_ID), E_{RSA}($K_{Private\_VotingPlace}$, Voting_Place_ID)	Session_ID / "Authentication Error"
SendPartyVotes	Sending the total amount of each party votes. The amount is for quick count only.	Session_ID, Number_Of_Party_Votes	"Success"/"Failed"
SendCandidateVotes	Sending the total amount of each candidate from a party votes. The amount is for quick count only.	Session_ID, Number_Of_Candidate_Votes	"Success"/"Failed"
Send	Sending the real and encrypted data as described in the Table 2	Session_ID, Vote	"Success"/"Failed"
GetSession	Used to get Session ID in case of connection to the upper level has been disconnected.	H_{SHA256}(Voting_Place_ID)	Session_ID
GetAmountSent	Get the amount of data sent by method Send()	SenderCode	(Integer)

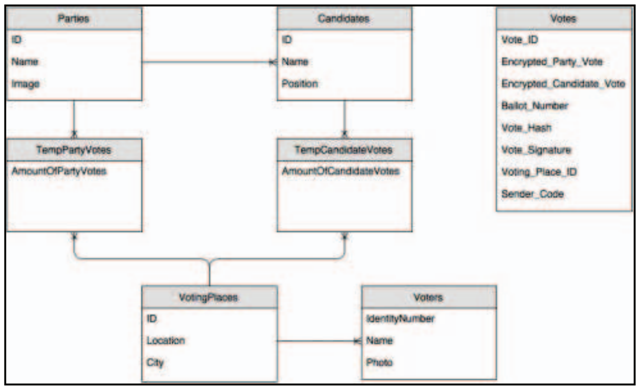


Fig. 3 Simplified Entity Relationship Diagram

C. *E-Voting Database Design*

As mentioned earlier, database design is the most important thing to look at. We have to ensure users privacy while making sure that the voting data is countable. To accomplish that, we have a table in the database of our web service which stores voting data. There are few other tables in the database, but those tables are not directly relevant to the security aspect, thus we omit it in this paper as well as some columns. Some relevant tables are Votes, Voters, Party, Candidates, and two temporary tables.

As you may have guessed, Voters save any important data regarding to the people who will vote, while Candidates is for saving the candidates from any parties. As shown on Table 2, Votes table, which is used as storing the people's choices, is not related to any tables. There are two reasons behind this unrelated table. Firstly, we don't want to relate the choices to the voters, because it will break privacy, therefore anyone who have access to the database can see who chooses whom.

Table 2 Votes Table Design

Attribute Name	Explanation
Vote_ID	H_{SHA256}(Chosen_Party_ID || Chosen_Candidate_ID || Vote_Time || Ballot Number || Voting_Place_ID). Used as primary key.
Encrypted_Party_Vote	E_{RSA}(K_{Public_CVC}, Chosen_Party_ID). Stored as base64 encoded string.
Encrypted_Candidate_Vote	E_{RSA}(K_{Public_CVC}, Chosen_Candidate_ID). Stored as base64 encoded string.
Ballot_Number	Generated number, mainly used to identify the ballot paper. The format of this number is as follows: P || Chosen_Party_ID || C || Chosen_Candidate_ID || Vote_Hash || Voting_Place_ID || "." || Serial_Number
Vote_Hash	H_{SHA256}(Chosen_Party_ID || Chosen_Candidate_ID)
Vote_Signature	E_{RSA}($K_{Private\_VotingPlace}$, Vote_Hash)
Voting_Place_ID	
Sender_Code	First 10 characters of Vote_ID. Used as part of sending results protocol.

Secondly, we used encryption to protect the votes before storing them to the database, this includes the foreign key that is usually used to relate data from a table to another. Detail about the encryption process will be explained in the following paragraph.

Votes table consists of nine columns, they are Vote_ID, Encrypted_Party_Choice, Encrypted_Candidate_Choice, Ballot_Number, Vote_Hash, Vote_Signature, Voting_Place_ID, and Sender_Code. As can be seen on table 2, most of the columns are encrypted value and encoded with base64 to make it readable and searched faster.

Let Hx(M) is the hash function with algorithm SHA256, taking value of M to output a hash value. M must be a single string or several concatenated strings by using double pipe symbol (||). We denote Ex(K, M) as encryption of message M with key K by SHA256 algorithm. These symbols are used to explain the columns of Votes table. This research uses RSA to encrypt most of the confidential messages, thus we need a pair of keys from the Central Voting Committee (CVC), public and private key.

IV. Experimental Result

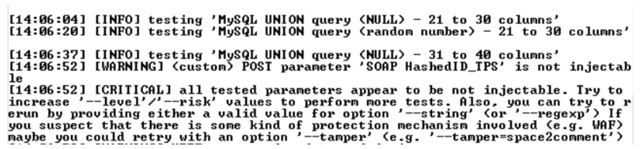
For the purpose of testing the functionality of our proposed method, we built a simple application to represent an application for scanning paper ballots, count them, and then send them to the upper server. To ensure the functionality was work as it needs to be, here are what we do as functionality testing:

1. Sent 250 records from local server at the voting place to upper level server.
2. Sent 250 records from the intermediary server to the next server.
3. Interrupt a sending process, then reconnect again to make sure the process does not restart from beginning.
4. Log on to the local application using username and password stored on the central server.

From our experiments, all of those four points were passed. But we did not stop there, there was another test. The main point of a secure e-voting system is obviously its security, which means our proposed system must be able to handle cheating method that could happen, prevent any kind of attacks to a web service, and maintain votes integrity.

To prove that our system is secure, we conducted four tests. First test was deliberately forge votes by changing the data that was about to be sent. Our system was able to detect such kind of cheating by checking the signature before storing them to the database. After that, to make sure no one is able to see the communication between servers, we used Wireshark to sniff the network and because we used HTTPS to protect the communication line, no one can see the packets, as long as the digital certificate is from trusted third party.

The third step was doing a replay attack. This attack could be happen when a vote sent twice, either deliberately or not, maybe due to a bad connection. Our proposed system was able to handle this due to checking of ballot number. Since ballot number was designed to be not easily guessed, attacker who intend to send double data would be failed, because if he send the same ballot number, our system will reject it. Even if he send the same vote with different ballot number, our system still rejects due to wrong crafted number.



```
[14:06:04] [INFO] testing 'MySQL UNION query (NULL) - 21 to 30 columns'
[14:06:20] [INFO] testing 'MySQL UNION query (random number) - 21 to 30 columns'

[14:06:37] [INFO] testing 'MySQL UNION query (NULL) - 31 to 40 columns'
[14:06:52] [WARNING] (custom) POST parameter 'SOAP HashedID_TPS' is not injectab
le
[14:06:52] [CRITICAL] all tested parameters appear to be not injectable. Try to
increase '--level'/'--risk' values to perform more tests. Also, you can try to r
erun by providing either a valid value for option '--string' (or '--regexp') If
you suspect that there is some kind of protection mechanism involved (e.g. WAF)
maybe you could retry with an option '--tamper' (e.g. '--tamper=space2comment')
```

Fig. 4 Sample SQLMap Result

Our last experiment was using vulnerability assessment tools to identify flaws in our proposed system, they are SQLMap to test for any SQL Injection and other web vulnerability scanner. Our system is a web service application, while SQLMap is commonly used for a usual website. Thus we had to make some adjustment to make SQLMap can send a SOAP request, instead of regular HTTP POST. Then SQL Injection attacks were launched against the each web method and none of them were a successful attack. A sample result of this experiment is shown on Figure 4.

V. Conclusion and Future Works

Based on the experiment conducted, our system is able to serve as an electronic voting system and maintain security of votes data as hard as possible. This can be proven from the penetration testing that our system does not have critical flaw that can be exploited. This is the core point of an electronic voting system.

For next research, it could be emphasize on the client side, which is the application on the voting place. The application could be a touch screen booth or may use a mobile phone as a voting medium. Security challenge on that topic is great, particularly on the mobile phone as a voting medium. We believe fraud on a mobile application is far greater and harder to handle than the conventional computer based.

VI. Acknowledgements

This research is fully supported by Directorate of Higher Degree, Ministry of Research, Technology, and Higher Degree with National Strategic Research scheme.

References

[1] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," MIT Technical Report, 1996.

[2] H. Pan, E. Hou, and N. Ansari, "Ensuring voters and candidates' confidentiality in E-voting systems," 34th IEEE Sarnoff Symposium, pp. 1-6, May, 2011

[3] A. Hassan and X. Zhang, "Design and build a secure e-voting infrastructure," IEEE Systems, Applications and Technology Conference (LISAT), pp. 1-7, May, 2013.

[4] R. Kusters, T. Truderung, and A. Vogt, "Clash attacks on the verifiability of e-voting systems," IEEE Symposium on Security and Privacy (SP), pp. 395-409, May, 2012.

[5] H. Alshammari, K. Elleithy, K. Almgren, and S. Albelwi, "Group signature entanglement in e-voting system," IEEE Systems, Applications and Technology Conference (LISAT), pp. 1-4, May, 2014.

[6] V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, "Election verifiability for Helios under weaker trust assumptions," Computer Security-ESORICS, pp. 327-344, 2014.