

# An Efficient Implementation of Electronic Election System

Naznin Fauzia

Tanima Dey

Inaba Bhuiyan

Md. Saidur Rahman

Bangladesh University of Engineering and Technology (BUET)/CSE, Dhaka, Bangladesh  
e-mail: laboni3\_6@yahoo.com, dey\_trina@yahoo.com, inaba\_bhuiyan@yahoo.co.in  
saidurrahman@cse.buet.ac.bd

**Abstract – Voting is usually recognized as one of the main characteristics of Democracy. Electronic election is a very recent idea regarding voting. Voter, once given his vote, has to rely upon the election system’s honesty and security. Free and fairness of an election is desired by almost everyone associated with it. Hence designing an election system needs special care. Furthermore, an electronic election should be more secure, transparent and trustworthy, as common people have less faith in computers due to system crashes and hacking threats. In this paper, we are going to describe our implementation of an efficient and secured electronic voting system based on the Fujioka- Okamoto-Ohta protocol which is the most practical and suitable protocol for large scale elections. Our implementation contains the automation of an online voting system providing some features which were absent in the previous implementations. We have made our system even more user friendly and secured but faster than the others using recent technologies and resources.**

**Keywords – Secured Electronic Election, Cryptography, RSA algorithm, Private key, Public key, Digital Signature, Blind Signature, Hash Algorithm**

## I. INTRODUCTION

Election, voting etc. are now well known terms in modern days of Democracy. Our paper is regarding such an election system that is called electronic election. Electronic election, as the name implies, is the election process held over electronic media, i.e. computers. For such a sensitive issue like election, security is one of the main concerns. But simplicity is also necessary to ensure the participation of common people. Besides security and simplicity, there may be some other issues that need to be considered. In that respect, we need to identify all such issues or properties that the election system must possess. A well-defined protocol (set of rules) is necessary to take care of all such requirements.

Computers and Internet are reaching almost every corner of the world. And with their increasing availability, most

services those require a face-to-face contact are getting replaced by their network version. Banking, shopping, corporate meeting are now being held over the Internet. Holding election over the Internet seems logical from many different points of view. Relief from long queues, minimal chance of voting error, verifiability (not possible in case of face-to-face service) and immigrant voters stands in favor of an electronic election. Recent improvements in network security has made it possible to design election system with high class security. But it is also important that carefully designed protocols and continuous improvements of the implementations are necessary to keep them out of reach from the network threats. From that point of view, electronic election appears to be just another application of computer and network security.

In real life, the whole election process can be taken under the careful observation of the Election Authority or Election Commissioner. Any kind of unfairness can be checked by honest officers while the whole voting process continues. But when we have to implement the whole process by computers, online, then we feel the need of automation of the whole process while ensuring the practicality and fairness. Since the whole process is to be done online through networks we also have to take care of the security.

In order to computerize elections from start to finish, there are many legal and technical problems that must be addressed. In general, the whole election process consists of several stages, such as registration, casting ballots (voting), counting votes and displaying results. To design a protocol for electronic elections, the difficulties those must be overcome are depicted below [5]:

- Ballots must be authentic yet untraceable.
  - Each voter must be able to check whether or not his ballot has been counted without compromising his privacy.
  - The election protocol must be protected against the illegal activities of both the eligible voters and dishonest outsiders.
- As we have depicted the problems and challenges of design-

ing an Electronic Election system, these shortcomings must be overcome and the properties of a secured electronic election system must be ensured by our implementation. Since our implementation is based on Fujioka-Okamoto-Ohta protocol, most of the properties of a secured electronic election protocols are fulfilled. But along with those, we have also ensured the practicality of the voting process. We are describing the main features of our system below:

1. We have ensured in our system excellent security by using RSA algorithm for all the interactions among different module of the system
2. In our system, none can falsify the result of the voting process since the votes are checked by every other component
3. Voters can check whether their votes are counted or not, in short, they can verify their votes but without compromising their privacy
4. No other voter can know whom the voter voted for except himself, that is, his ballot is kept secret from outsiders
5. Our system provides a very simple interface which is easy to operate
6. Our system provides automatic generation of the ballots (questions/ options) through easy to use interface by the election authority
7. We have provided in our system the provision of easy online registration system
8. Only eligible voters can vote using our system, since registration is only allowed to them through secret pin number
9. No voters can vote twice

The remainder of the paper is organized as follows. In Section II, we describe Fujioka-Okamoto-Ohta protocol. Section III deals with major components of our system. Section IV deals with comparisons of our system with other previous implementations. Finally Section V is a conclusion.

## II. FUJIOKA-OKAMOTO-OHTA PROTOCOL

The core theoretical protocol of our system is the Fujioka-Okamoto-Ohta protocol [2, 5]. The players in the protocol are voters, an administrator  $A$  and a counter  $C$ . It is described as follows:

### A. Assumptions

The assumptions of the Fujioka-Okamoto-Ohta protocol are given below:

1. The counter communicates with voters via an anonymous channel.
2. Ballots are computed using a *Bit Commitment Scheme*.
3. Every voter has its own *Digital Signature Scheme*  $SG$ .
4. The bit commitment scheme has two functions  $(f, g)$ . The function  $f$  encrypts binary strings into cryptograms (blobs) and the function  $g$  decrypts the cryptograms (open blobs) and reveals the bits. The blind signature uses two functions  $(B, U)$ . The function  $B$  takes the ballot  $x$  and a random integer  $r$  and computes the blind message  $e = B(x, r)$ . The blind message  $e$  is given to the administrator who signs the blind message and returns the blind signature  $d$ . The function  $U$  allows us to unblind the signature and to retrieve the signature of the administrator as  $SG_A(x) = U(d, r)$ .

### B. Registration stage

The steps of the registration stage of the Fujioka-Okamoto-Ohta protocol are given below:

1.  $V_i$  selects its vote  $v_i$ , which is typically a binary string and creates a blob for it, i.e.,

$$x_i = f(v_i, k_i), \text{ for a random } k_i$$

2.  $V_i$  blinds  $x_i$ , i.e., computes

$$e_i = B(x_i, r_i), \text{ using a random integer } r_i$$

3.  $V_i$  signs  $e_i$  by calculating  $s_i = SG_i(e_i)$  and sends the triple  $\langle ID_i, e_i, s_i \rangle$  to Administrator  $A$ , i.e.,

$$V_i \rightarrow A : \langle ID_i, e_i, s_i \rangle$$

where  $ID_i$  is the identity or name of the voter  $V_i$ .

4.  $A$  verifies whether
  - (a)  $V_i$  is eligible to vote,
  - (b)  $V_i$  has not already applied for registration, and
  - (c)  $s_i$  is valid.

If the three conditions hold,  $A$  generates the certificate  $d_i = SG_A(e_i)$  and sends it to the voter  $V_i$ , i.e.,

$$A \rightarrow V_i : d_i.$$

If any of the three conditions is violated, then the registration of  $V_i$  is declined.

5. When the deadline for registration has passed, the administrator announces the number of voters and publishes the list  $\langle ID_i, e_i, s_i \rangle$  of all registered voters.

### C. Voting stage

The steps of the voting stage of the Fujioka-Okamoto-Ohta protocol are given below:

1.  $V_i$  retrieves  $A$ 's signature for  $x_i$  by unblinding  $d_i$  so we get,

$$y_i = SG_A(x_i) = U(d_i, r_i)$$

2.  $V_i$  checks whether  $y_i$  is a signature generated by  $A$ . If the check fails,  $V_i$  complains by showing the pair  $(x_i, y_i)$ . Otherwise,  $V_i$  sends the pair  $(x_i, y_i)$  to the Counter  $C$  via an anonymous channel.
3.  $C$  verifies the signature  $y_i$  of the ballot  $x_i$ . If the check holds,  $C$  puts the triple  $\langle l, x_i, y_i \rangle$  into a list, where  $l$  is the consecutive number assigned to the ballot.

4.  $C$  publishes the list after voters have cast their ballots, i.e.,

$$C : \{ \langle l, x_i, y_i \rangle | i = 1, \dots, \alpha \}$$

where  $\alpha$  is the number of ballots cast.

### D. Opening and Counting stage

The steps of the opening and counting stage of the Fujioka-Okamoto-Ohta protocol are given below.

Each voter  $V_i$  checks whether,

1. The number of ballots on the list is equal to the number of voters. If the check fails, voters may reveal their secret random numbers  $r_i$  and indirectly indicate which ballots are forged.
2. The ballot  $x_i$  is on the list. If not,  $V_i$  complains by showing the valid pair  $(x_i, y_i)$ .

If the checks are successful,  $V_i$  sends the key  $k_i$  with the number  $l$  to  $C$  via an anonymous channel.  $C$  opens the blob  $x_i$  using the key  $k_i$  and retrieves the vote  $v_i = g(x_i, k_i)$ . The pair  $(k_i, v_i)$  is appended to the entry  $(x_i, y_i)$  on the list. Finally,  $C$  counts the tally and announces the results.

The steps of Fujioka-Okamoto-Ohta protocol are illustrated in Figure 1. Steps (1) and (2) in the figure represent registration stage. In step (1) the voter sends his identity  $ID_i$ , blinded encrypted ballot  $e_i$  and signed blinded encrypted ballot  $s_i$  to the administrator. The administrator verifies voter identity and eligibility and if the voter's identity is valid he generates a certificate  $d_i$  by signing the blinded ballot  $e_i$  and then sends it to the voter. The voter now retrieves the administrator's signature  $y_i$  on the ballot  $x_i$  by unblinding  $d_i$  and checks whether the signature is generated by the administrator. If the signature is valid, the voter sends the

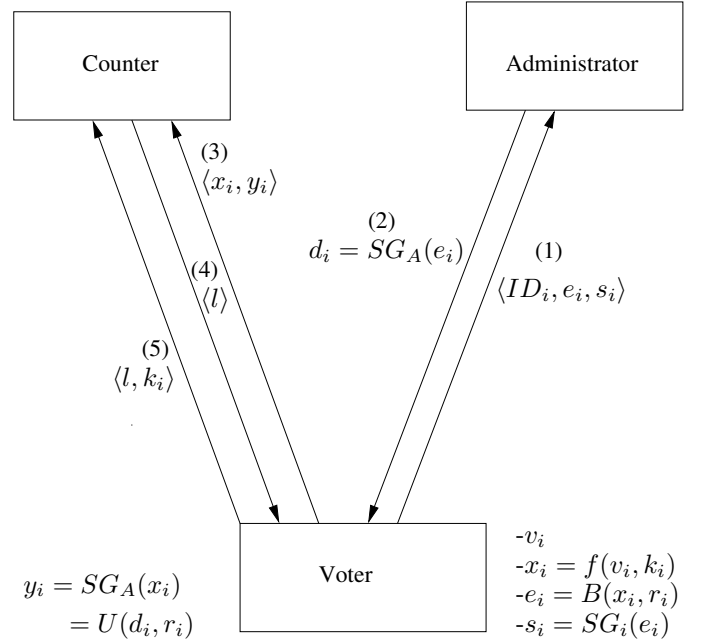


Figure 1: Block diagram of the Fujioka-Okamoto-Ohta protocol

encrypted ballot  $x_i$  and the signature  $y_i$  to the counter which is shown in step (3) in the figure. Counter verifies administrator's signature and if the check holds he puts the  $\langle l, x_i, y_i \rangle$  into a list, where  $l$  is the consecutive number assigned to the ballot (step (4)). Step (5) corresponds to opening and counting stage. The voter verifies if his ballot  $x_i$  is on counter's list and if the check holds he sends the key  $k_i$  with the number  $l$  to counter via an anonymous channel. Counter opens the blob  $x_i$  using the key  $k_i$  and retrieves the vote  $v_i$ . Finally, counter counts the tally and announces the results.

## III. COMPONENTS

Our implementation is based on the theoretical concept of the Fujioka-Okamoto-Ohta protocol and different cryptographic algorithms. In this section, we are going to describe the various components and modules of our voting system.

### A. Topology

The topology of our system is shown in Figure 2. It displays the basic operations step by step. In step (1), the voter's choices are combined in a plaintext vote and then hashed to produce the committed vote. The committed vote is then blinded to ensure voter's privacy. The blinded vote, the voter's userid and password are then sent to the Administrator Server. The combined message is encrypted using Administrator Server's public key. In step (2), after breaking

the encryption, the Administrator Server signs the blinded vote and sends it to the Voting Server. This time the message is encrypted using voter's public key. In step (3), again after breaking the encryption, the Voting Server combines committed vote and the administrator's signature and encrypts it twice, first time using Counter Server's public key and the second time using Anonymizer Server's public key. This encrypted message is then sent to the Anonymizer Server. In step (4), Anonymizer Server replies with an acknowledgement and removes one layer of encryption, using its private key. After deadline, the Anonymizer Server shuffles all the votes and sends them to the Counter Server (step 5). Finally in step (6), the Counter Server sends acknowledgement to Anonymizer Server, retrieves the actual message contents and counts the votes. The detailed description of these steps are discussed in the subsequent subsections of this chapter.

### B. Election Builder

Election Builder is a necessary but out of the protocol part of our system. Election Builder is the module of our system that provides the interface needed by the authority organizing and controlling the voting system to prepare the ballot. User of this module must go through a login procedure for his authentication before he can proceed. After successful login, he will be given three options Add, Edit and Delete a question. Our system provides option for multiple question voting. Each question has multiple choices. Questions can be edited or deleted by the authority anytime before the election process starts, so can be the choices for each of them.

### C. Registration

Registration can be considered as the pre-requisite step of our voting process. First of all, the person intended to vote must go to the registration office. There he is given a unique pair of voter ID and a secret pin number covered by a black paper after his eligibility for the voting system is verified.

Registration Server is responsible for the rest of the registration process. The voter opens the browser and sets up a connection with Registration Server. He then has to fillup a registration form and submit his voter ID and secret pin number from which his eligibility is ensured. The secret pin number prevents a dishonest voter from using other voters' ID, as he can guess a number as voter ID but it is almost impossible for him to guess the corresponding correct secret pin. If both the voter ID and secret pin number are valid, the voter can create a username and password of his own using which he can login to Voting Server to cast his vote.

### D. Voting Server

*Voting Server* is the server the voters interacts with. Voting Server of our system is a web server like *yahoo* or *msn*.

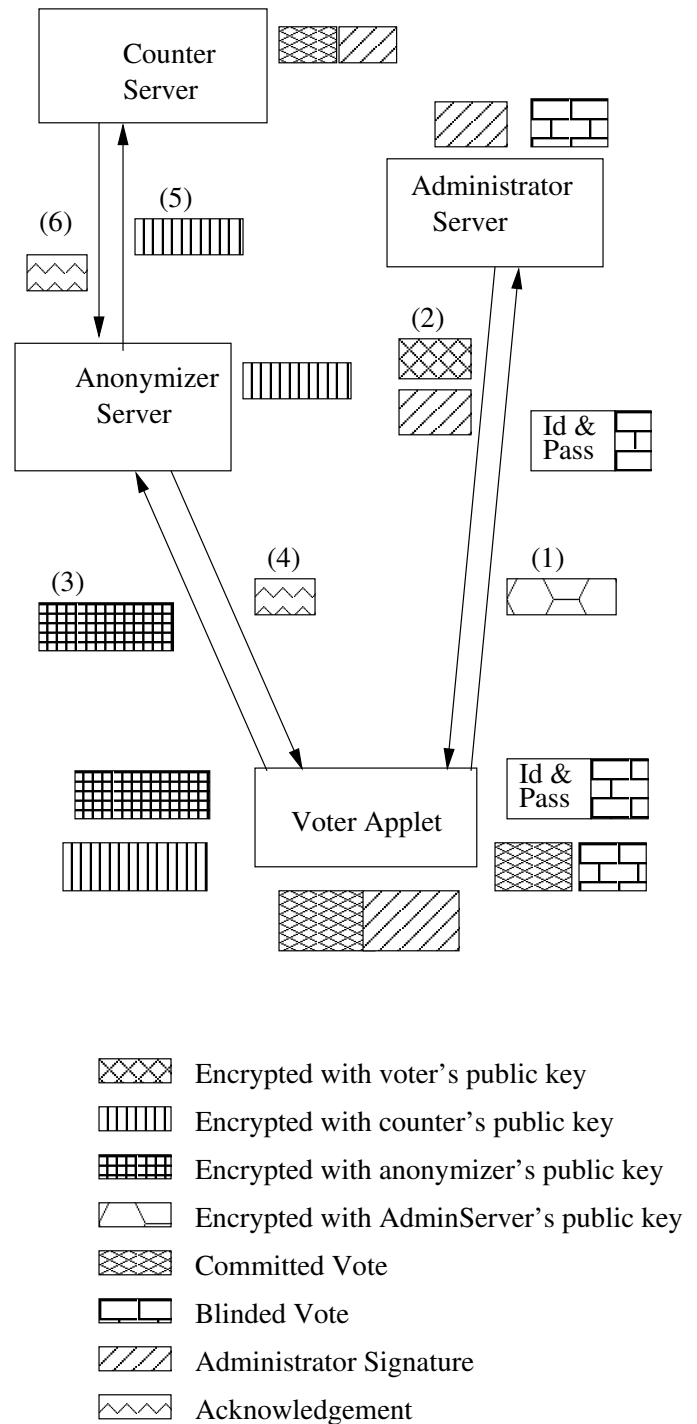


Figure 2: Block diagram of our system

Voter, using the web browser on his personal computer, can browse our voting web site. Then he can vote using three easy steps as described below:

**Login** Voter has to login using his username and password obtained during the registration procedure.

**Choosing** Voter can now choose his option or candidate he wants to vote for after successful login.

**Sending** Now voter just has to click the *send* button and wait until he is shown confirmation of his vote being sent to Anonymizer Server. Voter will be shown a set of information which he will be asked to copy and store for future verification purpose, as voter can check whether his vote is counted by searching through the *Counter List* after voting deadline.

Although it seems a simple procedure from voters' side, the vote sending process is actually a series of complex steps. The choices of the voter is collected in a string named *plain vote*. It is committed first using unique commitment key and Hashing algorithms [6, 7, 9] and then blinded using Blind Signature scheme. Then it sends the voter's username, hashed password and blinded committed vote to *Administrator Server* using RSA encryption algorithm [4, 8]. The signed blinded vote returned from Administrator Server is unblinded to get the actual signed committed vote and sent to Anonymizer Server along with **plain vote**, **commitment key** and **committed vote**.

#### *E. Administrator Server*

The work done by the administrator, *A* in the original Fujioka-Okamoto-Ohta protocol is done here in our Administrator Server component. After decrypting the message sent by the Voting Server, it first tests for any integrity errors. If there is no error, it checks the username and hashed password received with that of the stored in its database. First, it checks if the hashed password is correct for the username and then whether this user has voted previously or not. If the password is correct for that username and this user is voting for the first time, the Administrator Server applies its signature on this blinded committed vote using Digital Signature scheme [4] and sends this signed blinded committed vote to the Voter Server in encrypted form. Administrator Server also maintains two types of logfiles – in one type it keeps the entries of the valid votes along with received date and time and in the other type it logs different error encounters such as, integrity error, password mismatch error, voter's try for voting twice error etc. After voting deadline, it also provides the option for verifying votes in Administrator's list.

After receiving signed blinded committed vote from the Administrator Server, Voting Server verifies the signature of

the Administrator Server and if it is valid, unblinds the received vote and sends this signed committed vote, plain vote, commitment key and committed vote to the Anonymizer Server in double encrypted form. The first encryption is for Counter Server and the second for Anonymizer Server using RSA algorithm.

#### *F. Anonymizer Server*

Fujioka-Okamoto-Ohta protocol had no concept of Anonymizer. It assumes that all communication channels are anonymous channel. Anonymous communication channel ensures untraceability. Sender can send message to the receiver without revealing his identity. The plain vote, commitment key, committed vote and the signed vote are concatenated in a message before it is sent to Anonymizer Server by Voting Server. As was stated earlier, this message is double encrypted and Anonymizer Server removes the outer encryption and gets the inner encrypted message for Counter Server. Since this message is still encrypted, it can't see the message content, that is, the casted vote and saves the message until deadline. After the deadline is over, it suffles them and sends all these encrypted votes saved so far to the Counter Server.

#### *G. Counter Server*

The responsibility of Counter Server is analogous to the opening and counting stage of Fujioka-Okamoto-Ohta protocol. As we have stated earlier, plain vote, commitment key, committed vote, signed committed vote are concatenated in a fixed form and arrives at Counter Server via Anonymizer Server as a single encrypted message. After decrypting the message, Counter Server uses the commitment key with the plain vote and then calculate its hash value by Hash algorithms and compares with the committed vote received. If equality of the result holds, then the integrity of the message is ensured. Then it checks the administrator's signature by verifying the signed committed vote. Finally, if there are no errors, it counts the plain vote and makes tally. It publishes the list containing all plain vote, commitment key, committed vote and signed committed vote. Later any voter can check out the list to verify his vote on the list. This ensures the verifiability property of the voting process.

#### *H. Lists*

When the election deadline is over, the three servers discussed displays three individual lists to the voter containing different information.

The first list shows the poll result, that is the final winners of the election. The voter will not see the entire list. Only the winning choice for each question will be displayed.

The second list is Administrator Server's list used for verification purpose. The voter will be asked to enter his username and password. After his identity is verified, his voting record (blinded committed vote and signed blinded committed vote) stored at Administrator Server will be displayed.

The third list is Counter Server's list which displays the plain votes. In this step, the voter will need the information that he was asked to store upon the voting process completion. Those information included the committed vote, blinded vote and signed committed vote, all of which are unique numbers. The voter will be asked to enter the committed vote that he saved while voting after the notification from Anonymizer Server. If the voter fails to enter any valid number, no information will be shown to him. Otherwise, he will see the choices he has previously made for each of the questions.

#### IV. PREVIOUS IMPLEMENTATIONS AND COMPARISONS

We have found two implementations based on the Fujioka-Okamoto-Ohta protocol, Sensus [1] and E-Vox [3]. We are briefly stating the basic properties of both the systems and our improvements over them.

##### A. Sensus

Sensus is implemented by Cranor and Cytron. Sensus uses blind signatures to ensure that only registered voters can vote and that each registered voter only votes once, while at the same time maintaining the voter's privacy. Sensus allows voters to verify independently their votes.

Our system is more improved because we have introduced another module which we have stated as Anonymizer Server. Its job is to shuffle the vote for ensuring untraceability and more security. All the major modules/servers of our system are interacting through the Internet via encrypted messages which are hard to understand. At the same time, we have two additional modules for the registration and ballot preparing process.

##### B. E-Vox

The code for E-Vox project was developed by Mark Herschberg, Ben Adida and Randy Milbert, under the supervision of Professor Ron Rivest at MIT. E-Vox can easily support elections on the order of a hundred people. With sufficiently fast servers, a few thousand can be quite reasonable.

Our system has certain benefits over theirs. First of all, we have a Registration Server for supporting online registration. Our system is more user friendly, as the voters need not to download any software and have the programs to support it. Any person who can browse the Internet can vote easily. We have implemented Secured Channel as they have, but we

have used two layers of RSA algorithm for better security and more obscurity.

#### V. CONCLUSION

The problem of computer and the Internet security has taken a prominent and important place in today's research area. Since electronic election is a part of these applications, it is of supreme importance as we will consider its emerging advantages in today's modern life. This problem is open, till now researches in different universities and laboratories are still going on. The Internet is emerging into our lives everyday, and due to the application of this kind of advanced technology, we hope that new better protocols will be discovered and our system can be used in practical application in every context.

#### REFERENCES

- [1] L. F. Cranor, R. K. Cytron, *Sensus: A security-conscious electronic polling system for the Internet*, Hawaii International Conference on System Sciences, Wailea, Hawaii, USA, 1997.
- [2] A. Fujioka, T. Okamoto, K. Ohta, *A practical secret voting scheme for large scale elections*, Advances in Cryptology - AUSCRYPT'92, Lecture Notes in Computer Science, 718, Springer, pp. 244-251, 1993.
- [3] M. Herschberg, *Secure electronic voting using the world wide web*, Master's thesis, Massachusetts Institute of Technology, 1997.
- [4] J. F. Kurose, K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, Pearson Education Asia, First Indian reprint, Delhi, 2003.
- [5] J. Pieprzyk, T. Hardjono, J. Seberry, *Fundamentals of Computer Security*, Springer, Germany, 2003.
- [6] B. Preneel, P. vanOorschot, *Building fast MACs from hash functions*, Advances in Cryptology - CRYPTO'95, Lecture Notes in Computer Science, 963, Springer, pp. 1-14, 1995.
- [7] D. E. Eastlake, P. E. Jones, *US Secure Hash Algorithm 1 (SHA1)*, RFC 3174, 2001.
- [8] R. L. Rivest, A. Shamir, L. M. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, 21(2), pp. 120-126, 1978.
- [9] Federal Information Processing Standard, *Secure Hash Standard*, FIPS publication 180-1, 1995. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>