An Adaptive Plug-and-Play (PnP) Interactive Platform for An E-Voting Based Cybersecurity Curricula

Muwei Zheng[1 [0000−0001−5613−8736]], Nathan Swearingen[2 [0000−0001−5106−4088]], William Silva[3 [0009−0001−6221−6128]], Matt Bishop[1 [0000−0002−7301−7060]], and Xukai Zou[2 [0000−0001−5762−8876]]

[1] University of California - Davis, CA, 95616, USA
[2] Indiana University Purdue University Indianapolis, IN, 46202, USA
[3] University of Connecticut, CT, 06269,USA

Abstract. An electronic voting (e-voting) based interactive cybersecurity education curriculum has been proposed recently. It is well-known that assignments and projects are coherent parts of and important for any curriculum. This paper proposes a set of course projects, assignment design, and a coherent online plug-and-play (PnP) platform implementation. The PnP platform and the proposed exemplary assignments and projects, are systematic (derived from the same system), adaptive (smoothly increasing difficulty), flexible (bound to protocols instead of implementations), and interactive (teacher-student and student-student interactions). They allow students to implement parts of the components of this e-voting system, which they can then plug into the PnP system, to run, test and modify their implementations, and to enhance their knowledge and skills on cryptography, cybersecurity, and software engineering.

Keywords: Cybersecurity Education · Electronic Voting (E-Voting) · Interactive Teaching and Learning.

1 Introduction

Electronic voting (e-voting) technology has been an effective tool for cybersecurity education [6, 3, 11, 10, 4, 2] and recently, an e-voting based interactive cybersecurity education curriculum has been proposed [5, 13].

Interactive teaching and learning is advantageous, and interactions among students, instructors and inter-relations among course contents, projects, and assignments are an important part of enhancing students' learning outcomes [7, 8].

In this paper, we propose an interactive course project design, assignment design, and implementation methodology of a plug-and-play online voting platform. Such a PnP platform, plus the proposed exemplary assignments and projects, is systematic, adaptive, flexible, and allow students to implement parts of the components of this e-voting system, which they can then plug into the PnP system,

to run, test and modify their implementations, and to enhance their knowledge of cryptography, cybersecurity, networking, and the architectures of client/server distributed or peer-to-peer systems, as well as skills of protocol designs, internet programming, system implementation and integration.

Contribution and Related Work - To our knowledge, this is the first published set of coding projects dedicating on E-voting education. However, the field of E-voting may be too specific. But we also couldn't find many publications of coding projects dedicating on cybersecurity. There are many educational guidelines, like CSEC 2017 [1], but they don't provide matching coding projects. We hope our work fill this gap, and encourage teachers and professionals to share more cybersecurity coding projects with the community.

The paper is organized as follows. Section 1.1 introduces the mathematical background of the online voting protocol used by the PnP project; section 2 presents the e-voting system, and section 3 discusses the PnP platform. Section 4 describes the assignments, section 5 relates this to the CSEC 2017 cybersecurity guidelines, and the last section concludes the paper.

1.1 Overview of the e-voting protocol for interactive teaching and learning

As introduced in the last section, the authors in papers [15, 14] proposed a novel e-voting protocol involving multiple equivalent tallying servers and voting clients. To facilitate readers reading and understanding our work, and to make the contents of this paper self-contained and complete, we summarize the protocol here. For its technical details, please see papers [15, 14].

Assumptions: There are $n >= 3$ voters $V_1, \ldots, V_n$ and $m >= 2$ candidates $c_1, \ldots, c_m$ running for office. Let $L = nm$. Two or more "tallying authorities" check the validity of each vote and ballot and count the votes. For simplicity, we assume two tallying authorities here, denoted as $\mathcal{C}_1$ and $\mathcal{C}_2$. These tallying authorities have *conflicting interests*, such as representing different candidates or political parties. So they will not share information with each other but will cooperate to perform some tasks when dictated by the protocol. This matches how observers in current election processes work. The tallying authorities are called "collectors" too.

Cryptographic primitives and cryptosystems The first building block is a simplified (n, n) secret sharing scheme [12] (denoted as $\mathbf{S}\_(n, n)\_\mathbf{SS}$). A secret s is split into n shares s_i $(1 \leq i \leq n)$ with $s = \sum_{i=1}^{n} s_i$, over the group $\mathbb{Z}_M$, where $M \geq 2^L + 1$. Each member receives one share. All n members need to pool their shares together to recover s. The scheme is additively homomorphic [12]; the sum of two shares $s_i + s'_i$ (corresponding to s and s', respectively) is a share of the secret $s+s'$. The other is an efficient secure two party multiplication (STPM) protocol [9]. Initially, each authority C_i $(i = 1, 2)$ holds a *private* input x_i. At the end, each C_i gets a (random) *private* output r_i, such that $x_1 \times x_2 = r_1 + r_2$.

Other cryptographic principles and systems include the Discrete Logarithm Problem (DLP), RSA, and the Paillier public key cryptosystems.

The (mutual-restraining e-voting protocol (MR-EV) [15, 14] consists of the following three technical designs.

TD1: Universal viewable and verifiable voting vector. For n voters and m candidates, a voting vector $\mathbf{v}_i$ for V_i is a binary vector of $L = n \times m$ bits. The vector can be visualized as a table with n rows and m columns. Each candidate corresponds to a column. Via a robust location anonymization scheme at the end of registration, each voter *secretly* picks a unique row which no one else including tallying authorities knows. A voter V_i will put a 1 in the entry at his/her row and the column corresponding to a candidate he/she votes for (let the position be L_c^i), and put 0 in all other entries. During tallying, all voting vectors will be aggregated and the final tallied voting vector is public and viewable to anyone. From the tallied voting vector (denoted as $\mathbf{V_A}$), the votes for candidates are all viewable one by one and can be incrementally tallied by anyone. Any voter can check his vote and also visually verify that his vote is indeed counted in the final tally. Furthermore, anyone can verify the vote totals for each candidate.

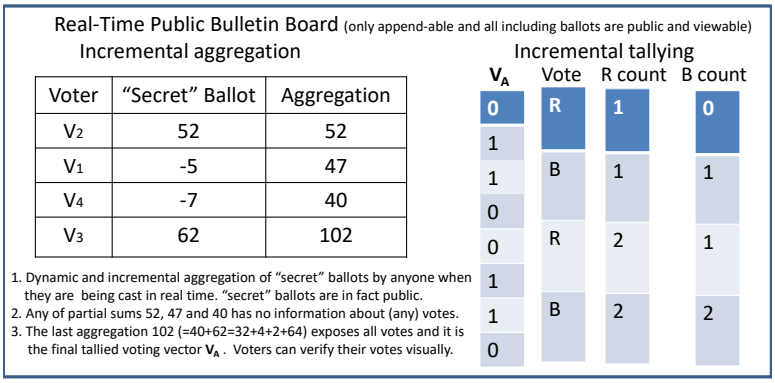
TD2: Forward and backward mutual lock voting. From their voting vector (with a single entry of 1 and the rest of 0es), voter V_i can derive a forward value v_i $(=2^{L-L_c^i})$ and a backward value v_i' $(=2^{L_c^i-1})$. These two values are his/her vote. Importantly, $v_i \times v_i' = 2^{L-1}$, regardless which candidate V_i votes for. During the vote-casting, V_i uses the simplified (n, n)-SS scheme twice to cast their vote using both v_i and v_i' respectively. v_i and v_i' jointly ensure the correctness of the vote-casting process, and will be used by collectors to enforce V_i to cast *one and only one* vote; any deviation, such as multiple voting, will be detected. Denote his/her own share of his/her vote v_i as s_{ii} and similarly, s_{ii}' for v_i'. His/her ballot will be (p_i, p_i') where p_i is the sum of s_{ii} and the sum of $n-1$ shares of $n-1$ votes of the other $n-1$ voters, one per voter. Similarly, for p_i'. Rather than casting their vote (v_i, v_i'), V_i casts their ballot (p_i, p_i').

To avoid the interactions or communications among voters (which of course is not practical at all), any voter only contacts collectors. Collectors generate and send $n-1$ random shares to the voter and the voter computes their share by subtracting the sum of the $n-1$ shares from their vote v_i (similarly for v_i').

To prevent a collector from having all $n-1$ shares for a voter, each collector creates half of the $n-1$ shares.

This e-voting model deliberately distinguishes between a *private vote* and a *secret ballot*. Voter's votes are known only to themselves. But its corresponding ballot, even though it is called a *secret ballot*, is revealed to the public in the vote-casting.

TD3: In-process check and enforcement. During the voting processes, collectors will jointly perform two cryptographic checks on the voting values of each voter (See Sub-Protocol 1 and Sub-Protocol 2 in [14]). The first check uses STPM to prevent a voter from incorrectly generating their share (s_{ii}, s_{ii}') of their vote (v_i, v_i'). The second check prevents a voter from publishing an incorrect ballot (p_i, p_i'). The ballot is the modular addition of a voter's own share and the share summations that the voter received from other voters (in fact, from collectors).



A voting example involving 4 voters and 2 candidates: R & B (numbers in red are kept secret)

Numbers with red or black underline are computed by voters, with blue double line generated by collector C1, and with green wavy line by collector C2

Voter V_i	Secret location L_i	Secret vote v_i	Secret random shares of C1 and C2 For V1: 5: received from C1; 15: from C2; $x_{1,i}$ $x_{2,i}$	"Secret" ballot – published, so they are in fact public For V1: 5: received from C1; 15: from C2; 52: computed as 32+5+15 by voter herself
V1	2	B (32)	5, 15	52 (=32+5+15)
V2	3	R (4)	1, -10	-5 (=4+1+(-10))
V3	4	B (2)	-20, 11	-7 (=2+(-20)+11))
V4	1	R (64)	14, -16	62 (=64+14+(-16))

Fig. 1. Bulletin Board (top section) and its corresponding example (bottom) (Modified and combined from Fig. 2 and Table 3 in [14])

2 E-voting system

To save space, the exact network packet design for each step during communication is attached in the Appendix (Section 8).

The protocol involves three-way communications among the administrator, collectors, and voters. The whole process is divided into 4 parts: Initialization, Registration, Voting, and Publishing Result.

Initialization – To create an election, an administrator first appoints two collectors. The administrator contacts each desired collector, and each collector responds to the administrator with their decision. The administrator can then check whether the desired collectors have accepted or rejected the request.

Once the administrator has chosen two collectors, the administrator sends election metadata (including information about the collectors) to each collector. This includes information enabling a collector to connect to the other collector.

Registration – The voter must prove their right to vote to the administrator before registering. How this is done is out of scope of this protocol. Once the voter is authorized to vote, they are given an ID and they give the administrator a public key. Both are just for this election.

The voter now connects to the administrator and registers, asking for each collector's host, port, and public key, and a list of the candidates.

Once all voters have registered (e.g. the registration period ends), the administrator sends the list of registered voters to each collector.

Voting – Each voter must obtain shares from each collector, then creates their ballots and commitments. They send these to both collectors. There are two sub-protocols used by collectors to communicate with each to perform secure two-party multiplication (STPM) to increment the votes, which is described in detail in the Appendix.

Publishing Result – As the collectors receive and verify ballots from the voters, they send them to the administrator. As the administrator receives ballots from the collectors, they sum them and publish them on the web-based bulletin board. No information about the vote totals is visible until all ballots are received.

3 The Adaptive Interactive Plug-and-Play Platform

Our project comes with an advanced online e-voting platform for students, which serves as a testbed and example. The platform is equipped with a ready-to-use e-voting system students can use to gain an understanding of how the platform works. The system consists of one administrator, two collectors, and a minimum of three voters.

Students who wish to use the platform create fictitious candidate names and begin the election. The election process will automatically run, and all the phases of the election will be printed out. This provides a comprehensive overview of how the system operates and what is expected from student implementations. Additionally, the platform can be used to test the students' own implementations. They can replace either the collector or the voter with their own implementations and connect them to the platform to verify that they are working correctly.

The unique flexibility of the platform is the key feature of this set of projects. Students can freely connect their collector or voter or both to the platform, and collectors and voters can come from different students. This feature makes our platform a plug-and-play e-voting system, where students can easily experiment with their own implementations and see if they work seamlessly with the platform.

Figure 2 shows an exemplary plug-and-play testing platform interface. With this interface, a student can set their test candidates, number of collectors (at least 2), number of voters (at least 3) and fill in their implementations of the collectors and voters. Whatever the remaining collectors (if any) or voters (if any) need will be automatically provided by the platform using the default implementation. If students do not provide anything, they can still run the system using the default implementations of collectors and voters.

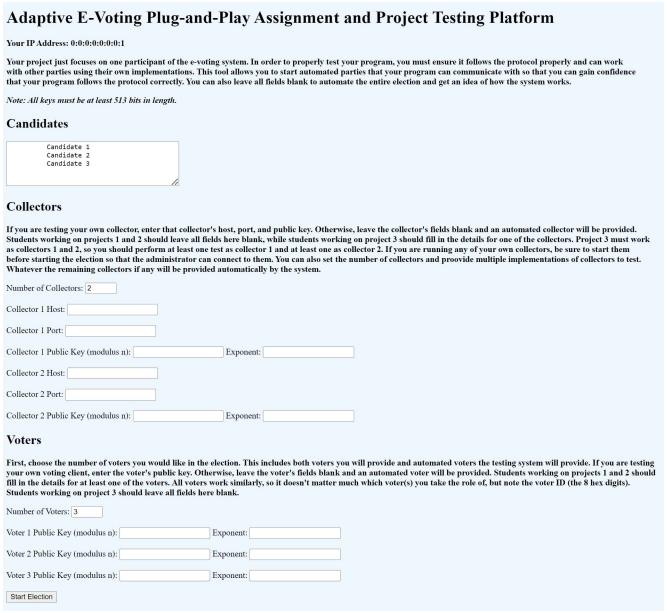


Fig. 2. An exemplary plug-and-play testing platform interface

The online platform and java source code needed for assignments are available on the web.[4]

4 The Assignments

In conjunction with the online platform that we have introduced, we have designed four engaging assignments for students to complete. While our system is written in Java and we will be using Java to demonstrate ideas in the assignment descriptions, students are not restricted to any particular programming language. The only requirement is that the protocol is implemented correctly to ensure a successful outcome.

The project entails implementing an administrator-collector-voter trilateral online voting system that serves as the backbone of the platform. Through the completion of these assignments, students will gain hands-on experience in cryptography, cybersecurity, and software engineering. This immersive approach allows students to apply theoretical knowledge to real-world scenarios and encourages active learning.

4.1 Assignment 1: Preliminary

In the first assignment, students will be introduced to the fundamental concepts of cryptography and the communication protocol between servers and clients.

[4] *http://cs.iupui.edu/~xzou/NSF-EVoting-Project*

Given that the e-voting process relies heavily on encryption techniques, students will need to gain practical experience implementing various cryptographic algorithms. These include hash computation, symmetric encryption, and asymmetric encryption.

The implementation process will involve utilizing specific algorithms for each technique, and students will be provided with the necessary references and guidance, but not the existing library. Thus, they are asked to implement the algorithms step by step. For hash computation, students will use the BLAKE2 algorithm. For symmetric encryption, the Salsa20 algorithm will be employed. As for asymmetric encryption, students will implement both the RSA and Paillier algorithms.

In addition to implementing cryptographic tools, students are required to practice setting up a TCP server and client communication. It is important to note that while TCP ensures the secure delivery of messages, it does not guarantee confidentiality or integrity. Therefore, students will also be required to encrypt and decrypt messages using the cryptographic tools they have implemented, and using private keys to sign their messages.

It is worth emphasizing that these two components of the assignment, implementing cryptographic tools and setting up TCP server and client communication, form the foundational basis for all subsequent assignments. The techniques that students will learn from this assignment will provide them with a solid foundation in the field of secure network communication.

4.2 Assignment 2: Voter

In this assignment, students will implement a voter client that will enable authorized voters to participate in an online election. It is important to note that the scope of the project does not cover whether the user is authorized to vote; it is assumed that any user attempting to vote is authorized. However, it is possible for these valid users to make mistakes, which will be addressed in assignment 4.

In the e-voting protocol, a voter is essentially a client in the communication process. It is also the easiest part of the communication system that includes the administrator, collector, and voter to implement. The primary focus of this project is to practice closely following a defined protocol (in this case, the e-voting protocol), and to implement a systematic way of sending and receiving encrypted messages according to the protocol, while also handling exceptions.

During an online e-voting session, there are five parts in a voter's communication process: registering, obtaining a location, creating ballots, creating commitments, and submitting votes. To register, voters will need RSA private and public keys. They will then obtain shares from collectors, which are used to protect the secrecy of their votes. Using the shares, they will compute ballots and commitments, and send them to both collectors. The votes of the voters are not revealed until all ballots are aggregated, at which point each vote becomes anonymous.

To assist in the implementation process, a framework for a voter class and other support classes will be provided. Students will be required to communicate

with the administrator and collector. At this point, the instructor will provide both the administrator and collector, which will enable students to test their implementations. Sample voters will also be provided, and while their source codes are hidden, students will be able to observe their network behaviors.

By the end of this assignment, students will have accomplished two objectives. The first objective is to gain a clear understanding of the step-by-step processes involved in how voters operate in an E-voting protocol. This will help them to grasp the intricacies of the system and the mechanisms that drive it. The second objective is to equip students with the skills and experience to confidently implement any networking communication protocol when necessary tools are provided (such as cryptographic tools in this assignment).

4.3 Assignment 3: Collector

In this assignment, students will create and implement a collector, which differs from the voter. Collectors are responsible for managing communications with the administrator, the other collector, and voters. This requires collectors to act both as servers and clients, making the implementation process considerably more complex than the voter client from the previous assignment.

During an online election, collectors play an essential role in the communication process, which consists of six stages. These include accepting collection requests, implementing location anonymization schemes, generating shares, communicating with voters, executing secure two-party multiplication, and forwarding verified ballots.

This assignment will allow students to expand and enhance their knowledge and skills in implementing a network communication protocol and the usage of cryptographic tools. Additionally, they will learn and practice secure multiparty computation techniques, a valuable skill set in many cybersecurity fields.

As usual, a framework for the collector class and supporting classes will be provided to the students. These classes will allow students to practice the important concept of encapsulation by reusing previously implemented classes. Additionally, the assignment includes an administrator and the other collector for students to test their implementations.

By utilizing pre-existing classes, students will be able to gain a deeper understanding of how various software components can work together to create complex systems. This will also provide an opportunity for students to hone their skills in software design and implementation, as well as improve their ability to work with existing codebases.

4.4 Assignment 4: Administrator

In this upcoming project, students will be tasked with implementing the administrator class, which is the final component of the e-voting system consisting of administrator-collector-voter. The administrator class has multiple jobs, including creating an election, registering collectors and voters, distributing election

and collector information, and publishing voting results. As both server and client, the administrator class is responsible for communicating with collectors and voters. It is to use the same cryptographic tools developed in previous assignments. Thus, students who have successfully completed previous assignments should find the implementation of the administrator class relatively straightforward.

However, in this assignment, the focus will shift from simply testing the accuracy of the implementation to assessing the reliability of the system. Unlike in previous projects, where students only had to test their implementations with legitimate votes to ensure the protocol was correctly followed, this project will also examine how well the system handles malicious inputs, such as double voting. To address this, students will need to incorporate necessary input sanitation and security measures in both the administrator and collector classes to mitigate the effects of malicious inputs.

After completing these assignments, students should have a working 2-collector online voting system of their own. While the e-voting platform does offer default collectors and voters, students' systems must be able to operate autonomously. Furthermore, as all students will be implementing the same protocol, the voting systems created by different students should be compatible with one another. This means that administrators, collectors, and voters from various students' projects should be able to seamlessly connect and conduct elections. The ability to interoperate is a crucial aspect of online voting systems (as well as many other client/server models) and is a key requirement for the success of the final project.

The four assignments have been designed to increase progressively in difficulty, with each subsequent assignment building upon the skills learned in the previous one while introducing new concepts.

From a software engineering standpoint, the first assignment serves as a gentle introduction to coding, with each component working mostly independently. In contrast, the second assignment requires a deeper understanding of class interactions and emphasizes the importance of following a defined protocol closely.

The third assignment represents a significant step up in complexity, demanding mastery of the three core skills from the previous two assignments: cryptography, networking, and software engineering. Students will need to demonstrate their ability to apply these skills in a more sophisticated manner, working with more intricate systems.

Finally, the fourth and last assignment delves into the realm of secure coding and introduces the basics of attacks and defenses in e-voting systems. This assignment will challenge students to apply their knowledge in a practical way, creating secure and robust systems that can withstand a variety of potential threats.

Each project results in a 2-collector online voting system. As an open invitation, enthusiastic students are encouraged to further enhance this system and create a more sophisticated N-collector online voting system.

This online voting system can be an invaluable resource for future research into security, particularly in areas such as man-in-the-middle attacks, database security for storing a large volume of ballots, and secure transmission and connection.

5 Relevant Topics from Cybersecurity Guidelines

To assist instructors in gaining a better understanding of the project and making informed decisions about its usefulness in their teaching, we analyzed the relevant topics outlined in the CSEC 2017 guidelines (shown in Table 1) and the e-voting curriculum learning objectives (shown in Table 2). The e-voting curriculum was introduced in [5, 13]. Here, a brief overview of CSEC 2017 guidelines will be given.

CSEC 2017 is a set of cybersecurity curricular guidelines developed by a joint task force of the ACM, IEEE Computer Society, the AIS Special Interest Group on Security, and the International Federation of Information Processing Societies' Working Group 11.8, dealing with computer security education. It defines eight Knowledge Areas, each composed of different Knowledge Units broken into topics. We found that the concepts covered by these topics and learning objectives are either already addressed by the project or necessary to understand and successfully execute the project.

This mapping enables instructors to determine the project's alignment with their classes' requirements. This information will help instructors assess the suitability of the project for their students and make informed decisions about integrating it into their teaching curriculum.

6 Conclusion

Our team has developed an innovative 2-collector online voting system that utilizes advanced cryptographic tools and secure multi-party computation techniques to ensure the security and integrity of online elections. The system is designed with modular components that can be easily disassembled and reassembled, making it an excellent tool for students to test and implement their own online voting systems in a step-by-step manner. By providing students with ample opportunities to practice their skills in cybersecurity and software engineering, we believe they will be able to integrate their knowledge from the classroom with real-world development.

We plan to incorporate this system into our future classes and have made it publicly available in the hope that other educators and developers will also find it useful. By sharing this resource, we hope to contribute to the ongoing efforts to promote secure and transparent online elections globally.

Knowledge Area	Knowledge Units	Topics
Data Security	Cryptography	Basic concepts, advanced concepts, mathematical background, symmetric ciphers, asymmetric ciphers
	Data Integrity and Authentication	Authentication strength, data integrity
	Data Privacy	Overview
Software Security	Fundamental Principals	Least privilege, Fail-safe defaults, Separation, Economy of mechanism, Least astonishment, Open design, Layering, Abstraction, Modularity, Design for iteration
	Design	Derivation of security requirements, specification of security requirements
	Implementation	Validating input and checking its representation, using APIs correctly, using security features, handling exceptions and errors properly, programming robustly, encapsulating structures and modules, taking environment into account
	Analysis and Testing	Static and dynamic analysis, unit testing, integration testing, software testing
	Documentation	User guides and manuals
Connection Security	Distributed Systems Architecture	The Internet, protocols and layering
	Network Architecture	General concepts, common architectures
	Network Services	Concept of a service, service models, service protocol concepts, service virtualization, vulnerabilities and example exploits
System Security	System Thinking	Security of special-purposes systems

Table 1. Relevant CSEC2017 Topics

7 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. DGE-2011117 and DGE-2011175. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. ACM/IEEE-CS/AIS SIGSEC/IFIP WG 11.8 Joint Task Force. Cybersecurity curricula 2017: Curriculum guidelines for undergraduate degree programs in cybersecurity. Technical Report Version 1.0, ACM, New York, NY, USA, December 2017.
2. Matt Bishop and Deborah A. Frincke. Achieving learning objectives through e-voting case studies. *IEEE Security & Privacy*, 5(1):53 – 56, 2007.

Module	Learning Objectives
Introduction to E-Voting	0.2 Understand different parties involved in an E-Voting process.
	0.3 Understand law and policy requirements for E-Voting systems.
Authentication	1.2 Describe how voters authenticate themselves.
	1.4 Understand software security principles and practices of robust, secure coding.
Confidentiality	2.1 Understand basic cryptography concepts.
	2.2 Describe public key cryptography and algorithms.
	2.3 Describe how public key cryptography is used in end-to-end encryption protocols.
Data integrity and message(sender) authentication	3.1 Understand different ways to generate and use hash functions.
	3.2 Describe techniques used to store protected data and to verify or compute them without revealing sensitive information.
Cryptographic Key Management	4.1 Describe common key exchange protocols.
	4.3 Explain how secret keys are used in proofs of identity, integrity protection mechanisms, and challenges in doing so.
Privacy and anonymity	5.2 Describe the procedures taken in elections to protect voter privacy.
	5.4 Explain techniques used to help voters verify their votes being recorded correctly without being able to reveal those votes.
Secure Group/Multi-Party Interaction and Secret Sharing	7.1 Describe schemes for multi-party secret sharing.
	7.2 Describe how these schemes handle insider threats.
	7.3 Explain how these schemes protect transmissions of shares.
Secure Multi-Party Computation and Homomorphic Encryption	8.1 Describe different schemes for secure multi-party computation.
	8.3 Explain how voters can verify the correctness of the results of the election.
Attacks and defenses	9.4 Simulate different attacks targeting E-Voting systems.

Table 2. Relevant E-Voting Curriculum Learning Objectives

3. Q. I. Cutts and G. E. Kennedy. Connecting learning environments using electronic voting systems. *Australiasian Computing Education Conf.*, pages 181–186, 2005.
4. Alex J. Halderman. secure digital democracy. *https:www.coursera.orginstructorjhalderm*, 2014.
5. Ryan Hosler, Xukai Zou, and Matt Bishop. E-voting technology inspired interactive teaching and learning pedagogy and curriculum development for cybersecurity

education. In *IFIP World Conf. on Info. Sec. Edu. (WISE'14)*, pages 27–43, 2021.
6. G. E. Kennedy and Q. I. Cutts. The association between students' use of an electronic voting system and their learning outcomes. *Journal of Computer Assisted Learning*, 21(4):260 – 268, 2005.
7. D. A. Kolb. Experiential learning: Experience as the source of learning and development. *Englewood Cliffs, NJ, Prentice Hall*, 1984.
8. D. Laurillard. Rethinking university teaching: a conversational framework for the effective use of learning technology, 2nd edition. *London, RoutledgeFarmer*, 2002.
9. S. Samet and A. Miri. Privacy preserving ID3 using Gini index over horizontally partitioned data. In *Proc. of IEEE/ACS*, AICCSA'08, pages 645–651, 2008.
10. Michael I. Shamos. Electronic voting. *http:http:euro.ecom.cmu.eduprogramcoursestcr17-803*, 2014.
11. Jeffrey R. Stowell and Jason M. Nelson. Benefits of electronic audience response systems on student participation, learning, and emotion. *Teaching of Psychology*, 34(4):253–258, 2007.
12. X. Zhao, L. Li, G. Xue, and G. Silva. Efficient anonymous message submission. In *IEEE INFOCOM'12*, volume 2012, pages 2228–2236, May 2012.
13. Muwei Zheng, Nathan Swearingen, Steven Mills, Croix Gyurek, Matt Bishop, and Xukai Zou. Case study: Mapping an e-voting based curriculum to csec2017. In *Proc. of ACM SIGCSE TS 2023 (accepted, Best Paper Award)*, 2023.
14. X. Zou, H. Li, F. Li, W. Peng, and Y. Sui. Transparent, auditable, and stepwise verifiable online e-voting enabling an open and fair election. *Cryptography, MDPI*, 1(2):1–29, 2017.
15. X. Zou, H. Li, Y. Sui, W. Peng, and F. Li. Assurable, transparent, and mutual restraining e-voting involving multiple conflicting parties. In *Proceedings of the 2014 IEEE Conference on Computer Communications*, IEEE INFOCOM 2014, pages 136–144, Piscataway, NJ, USA, April 2014. IEEE.

8 Appendix

Initialization - Table 3 - 5

Signed	
message_type	TYPE_COLLECT_REQUEST
election_ID	16 bytes
collector_index	1 byte
pk_length	2 bytes
pk	variable length
collector_key_hash	16 bytes

Table 3. Administrator sends to collector

Signed	
message_type	TYPE_COLLECT_STATUS
key_hash	16 bytes
election_ID	16 bytes
acceptance	0x00 or 0x01

Table 4. Collector responds

Signed	
message_type	TYPE_METADATA_COLL
key_hash	16 bytes
election_ID	16 bytes
other_C_host_length	2 bytes
other_C_host	var. length
other_C_port	2 bytes
other_C_pk_length	2 bytes
other_C_pk	var. length
M:	1 byte

Table 5. Administrator distributes election metadata to each collector

Table 6. packets during initialization state

Registration - Table 7 - 9

Voting - Table 11 - 13

Following Network Packet Design Due to the limitation on size of paper, we couldn't present all the packets. However, they can be found on our website. We hope the packets presented above are able to provide readers an overall impression about the project.

Signed	
message_type	TYPE_REGISTER
key_hash	16 bytes
election_ID	16 bytes
voter_ID	4 bytes

Table 7. Voter sends to administrator for registration

Signed	
message_type	TYPE_METADATA_VOTER
election_ID	16 bytes
C1_host_length	2 bytes
C1_host	var. length
C1_port	2 bytes
C1_pk_length	2 bytes
C1_pk	var. length
C2_host_length	2 bytes
C2_host	var. length
C2_port	2 bytes
C2_pk_length	2 bytes
C2_pk	var. length
M	1 byte
name1_length	1 byte
name1	var. length
...	...

Table 8. Administrator responds

Signed	
message_type	TYPE_VOTERS
election_ID	16 bytes
N	4 bytes
voter1_ID	4 bytes
voter1_pk_length	2 bytes
voter1_pk	var. length
...	...

Table 9. Administrator sends the list of registered voters to each collector

Table 10. packets during registration state

Signed	
message_type	TYPE_SHARES_REQUEST
key_hash	16 bytes
election_ID	16 bytes
voter_ID	4 bytes

Table 11. Voter connects to collector

Signed and Encrypted	
message_type	TYPE_SHARES
key_hash	16 bytes
election_ID	16 bytes
N	4 bytes
Si, Cj	k bytes
S'i,Cj	k bytes
~Si, Cj	k bytes
~S'i,Cj	k bytes

Table 12. Collector responds

Signed	
message_type	TYPE_BALLOT
key_hash	16 bytes
election_ID	16 bytes
voter_ID	4 bytes
p_i	k bytes
p'_i	k bytes
g^s_ii	k bytes
g^s'_ii	k bytes
g^(s_ii s'_ii)	k bytes

Table 13. Voter sends ballots and commitments

Table 14. packets between voters and collectors in voting state