Electronic voting

*Foundations of Secure e-Commerce (bmevihim219)*

Dr. Levente Buttyán
Associate Professor
BME Hálózati Rendszerek és Szolgáltatások Tanszék
Lab of Cryptography and System Security (CrySyS)
buttyan@hit.bme.hu, buttyan@crysys.hu
Electronic voting

- different types of voting embracing both electronic means of casting a vote and electronic means of counting votes

- examples:
  - punch cards and optical scan voting systems
  - Direct Recording Electronic (DRE) voting systems
  - public network DRE systems (voting via the phone or the Internet)

- general advantages:
  - increased speed of counting the ballots
  - less prone to counting errors
  - convenience and lower cost (e.g., voting from home via the Internet)
  - increased volume of participation
  - improved accessibility for disabled voters

- however, security, privacy, and verifiability are of paramount importance
  - in order to have confidence in the election results, people must believe that all election tasks are performed properly
  - traditionally, election fraud has been prevented through the use of physical security measures, audit trails, and observers representing of all parties involved
  - similar measures must be implemented in e-voting systems to avoid electoral fraud
Controversial issues

- electronic ballots
  - pro
    - no risk of exhausting the supply of ballots
    - no need for printing of paper ballots (significant cost reduction)
    - voting machines can be programmed to provide ballots in multiple languages for a single machine
  - contra
    - cost of software validation, compiler trust validation, installation validation, delivery validation and validation of other steps related to electronic voting is complex and expensive (may be more expensive than printing ballots)
    - many problems could be solved by simply printing ballots on-demand at voting locations

- audit trails and auditing
  - how to assure that the votes were recorded as cast and tabulated as recorded?
  - Voter Verified Paper Audit Trail (VVPAT) concept (Mercuri method)
    - the voting machine prints a paper ballot that can be visually verified by the voter
    - the paper ballot is treated as the official ballot and the electronic records are used only for an initial count
    - in any subsequent recounts or challenges, the paper ballot would be used for tabulation
  - however, whenever a paper record serves as the legal ballot, that system will be subject to the same benefits and concerns as any paper ballot system
Public network DRE systems

- a public network DRE voting system is an election system that uses electronic ballots and transmits vote data from the polling place to another location over a public network
  - vote data may be transmitted
    - as individual ballots as they are cast
    - periodically in batches throughout the election day
    - in one batch at the close of voting

- Internet voting can use remote locations (voting from any Internet capable computer) or can use traditional polling locations with voting booths consisting of Internet connected voting systems
  - frequently used by corporations and organizations to elect officers and Board members
  - it has been used for public elections in UK, Ireland, Switzerland, Estonia, and in some parts of the US
A simple (but insecure) protocol

- participants:
  - voters
  - election server run by the election authority (validates voter’s eligibility for voting, and collects and counts votes)

- assumption:
  - eligible voters registered with the election server before the election begins

- protocol:
  - the voter submits its ballot together with the voter ID to the server
  - the server uses the voter ID to check if the voter is registered and has not voted yet
  - if OK, then the server records the vote and adds it to the election tally
Problems with the simple protocol

- voters can be impersonated, ballots can be sent under any voter ID (authentication)
- server can alter and forge any vote, and votes can be eliminated (integrity)
- the server sees how particular voters vote (privacy)
- voters cannot verify if their votes have been counted correctly (verifiability)
Security and privacy requirements

- **authenticity**
  - only eligible voters can vote
  - each eligible voter can vote only once
  - voters cannot be impersonated

- **integrity**
  - it is not possible to alter a vote (either valid or invalid) once it has been casted
  - it is not possible for a validated vote to be eliminated from the final tally
  - it is not possible for an invalid vote to be counted in the final tally

- **privacy**
  - neither election authorities nor anyone else can link any ballot to the voter who casted it (otherwise voters can be extorted)
  - no voter can prove that he or she voted in a particular way (otherwise votes can be sold)

- **verifiability**
  - anyone can independently verify that all votes have been counted correctly
  - a weaker definition: voters can verify if their own votes have been counted correctly
The Two Agency Protocol

- **participants:**
  - voters, validator, tallier

- **assumption:**
  - eligible voters register with the validator
  - the validator distributes a secret identification tag to each voter
  - the validator sends the list of all identification tags (with no record of the corresponding voters) to the tallier

- **protocol:**
  - each voter sends its identification tag and an encrypted file containing a copy of the tag and the voted ballot to the tallier (through an anonymous channel)
    - the tallier can make sure the identification tag is valid, but it cannot see the contents of the ballot
  - the tallier publishes the encrypted file
    - the voter has proof that the file was received on time
  - the voter sends to the tallier the key necessary to decrypt its vote (through an anonymous channel)
  - when the election is over, the tallier publishes the list of all voted ballots together with the corresponding encrypted files
    - the voters can confirm that their votes were counted properly
    - any voter who finds an error can protest by submitting the encrypted file and decryption key again (as the encrypted file was published earlier, the tallier cannot deny having received it)
Properties of the TA Protocol

- **authenticity**
  
  +/- only eligible voters can vote (tallier checks validity of voter IDs, but voter IDs can be sold
  
  + each eligible voter can vote only once (tallier can check this)
  
  + voters cannot impersonate each other (voter IDs are secret)
  
  - but tallier can impersonate voters
    
    • tallier can vote in the name of those who didn’t vote
    
    • tallier can prevent eligible voters from voting (by preempting them)
    
    • victim voters cannot prove that the above misdeeds happened

- **integrity**
  
  + it is not possible to alter a vote (votes are published and voters can complain if the published vote does not match their intended vote)
  
  - it is possible for a validated vote to be eliminated from the final tally (encrypted votes are published and voter can complain if its vote is not published later on – but cannot prove that the tallier received the key ?!?)
  
  + it is not possible for an invalid vote to be counted in the final tally (published votes can be verified by a third party with the keys supplied by the tallier)

- **privacy**
  
  +/- neither election authorities nor anyone else can link any ballot to the voter who casted it (holds only if the validator and the tallier cannot collude)
  
  - voters can prove that they voted in a particular way (by revealing the encryption key they used to file the encrypted vote)

- **verifiability**
  
  + voters can verify if their own votes have been counted correctly (votes are published)
Fujioka-Okamoto-Ohta Protocol

- **participants:**
  - voters, validator, tallier

- **assumption:**
  - eligible voters register with the validator
  - eligible voters have a private signature generation key, and the corresponding public key is known to the validator
  - the validator also has a signature key pair, its public key is known by the other participants

- **protocol:**
  - the voter prepares a voted ballot, encrypts it with a secret key, and blinds the result
  - the voter then signs the blinded ballot and sends it to the validator
  - the validator verifies that the signature belongs to a registered voter who has not yet voted
  - if the verification is successful, the validator signs the ballot and returns it to the voter
  - the voter removes the blinding, and sends the resultant signed encrypted ballot to the tallier (via some anonymous channel)
  - the tallier checks the signature on the encrypted ballot, and if it is valid, the tallier publishes the encrypted ballot
  - voters verify that their ballots are on the published list and send the tallier the decryption keys (via some anonymous channel)
  - the tallier uses these keys to decrypt the ballots and add the votes to the election tally
  - after the election, the tallier publishes the decryption keys along with the encrypted ballots
Properties of the F-O-O protocol

- **authenticity**
  - only eligible voters can vote, each eligible voter can vote only once, voters cannot impersonate each other
  - tallier alone cannot vote in the name of those who didn’t vote (a third party may verify that the tallier has the validator’s signature on each encrypted ballot)
  - tallier and validator cannot collude to vote in the name of those who didn’t vote (a non-voting user may complain and the validator must present a signature from that user)
  - validator can submit fake signed encrypted ballots to the tallier and jeopardize the election (total number of submitted ballots will not match the total number of voter signed blinded ballots received by the validator), but by doing so the validator proves that it misbehaved

- **integrity**
  - it is not possible to alter a vote (votes are published and voters can complain if the published vote does not match their intended vote)
  - it is possible for a validated vote to be eliminated from the final tally (encrypted ballots are published and voter can complain if its key is not published later on – but cannot prove that the tallier received the key?!?)
  - it is not possible for an invalid vote to be counted in the final tally (published votes can be verified by a third party with the keys supplied by the tallier)
Properties of the F-O-O protocol

- **privacy**
  - neither election authorities nor anyone else can link any ballot to the voter who casted it (blind signatures)
  - voters can prove that they voted in a particular way (by showing the decryption key for an encrypted ballot before the key is published by the tallier)

- **verifiability**
  - voters can verify if their own votes have been counted correctly (encrypted ballots and decryption keys are published)
Other problems: Voting platform

- most likely a PC running MS Windows and Internet Explorer or Firefox browser → lot of vulnerabilities exist

- malicious payload
  - software or configuration information designed to do harm
  - a malicious payload on a voting client can actually change the voter’s vote, without the voter or anyone else noticing, regardless of the kind of encryption or voter authentication in place
  - the malicious module can then erase itself after doing its damage so that there is no evidence to correct, or even detect the fraud
  - more and more toolkits have been developed to enable unsophisticated computer users to create and launch such malicious payload

- delivery mechanism (or, how to install malicious payload?)
  - physical installation (breaking into houses, accessing machines in someone’s house when visiting, installing the malware on public machines, etc.)
  - remote automated delivery (virus like propagation in e-mails, ActiveX controls embedded in web pages, browser plug-ins, DLLs)
Other problems: Com. infrastructure

- Cryptography can be used to protect the communication between the user’s browser and the election servers.
  - Technology is mature and can be relied upon to ensure the integrity and confidentiality of the network traffic.

- Massive distributed denial of service (DDOS) attacks.
  - Communications may be disabled during the election due to attackers causing routers to crash, election servers to get flooded, or a large set of hosts, possibly targeted demographically, to cease to function.
Other problems: Social engineering

- fooling people into compromising their security
  - ordinary people have difficulty to follow simple directions (for example, when instructed to circle a candidate’s name, people will often underline it)

- send an e-mail message to a user telling him to click on a link, which would then bring up the fake voting site
  - adversary could then collect the user’s credentials and, in a sense, steal the vote
  - an attacker could also act as a man in the middle by setting up a connection to the legitimate server and feeding the user a fake web page

- attacks on the DNS (Domain Name System)
  - could be used to direct a user to the wrong web server when the user types in the name of the election server in the browser
Other problems: Spec. hardware

- tamper-resistant devices, such as smart cards, can generate and store cryptographic keys and they can perform computations

- home PCs usually don’t have smart card readers
  - some people are more limited in their ability to spend, and it is unfair to decrease the likelihood that such people vote (poll tax, digital divide)

- the smart card does not interact directly with the election server
  - communication goes through the user’s computer
  - at the very least: the code could prevent the vote from actually being cast, while fooling the user into believing that it was
  - at worst: it could change the vote

- the smart card does not interact directly with the user, either
  - communication goes through the user’s computer
  - the computer can display anything to the user, while it can feed something else to the smart card

- other special devices, e.g., mobile phones, could be used as smart card terminals, but…
  - such devices getting more and more closer to a regular PC (platform cannot be trusted)
  - even larger digital divide
DRE system components

- **DRE - Direct Recording Electronic voting machine**
  - usually a touch-screen device used to record the voter’s choices

- **VVPAT - Voter-Verified Paper Audit Trail**
  - a paper-based record of the choices selected by the voter
  - the VVPAT printer is hooked to the DRE
  - the paper record is viewable by the voter, but it is under a transparent cover so that it cannot be modified

- **EMS - Election Management System**
  - System responsible for the initialization of the components that collect the votes and also for the final tallying of the votes
  - usually located at the election central and it is often implemented as software running on a commodity PC

- **Optical Scanner**
  - optical reader that counts votes cast on paper ballots
  - there is usually one scanner at each polling site and one at election central

- **DTD - Data Transport Device**
  - storage device to transfer data (e.g., ballot information to the DREs and voting results to the EMS) between different components of the systems
Examples
Election process

- Prior to the election:
  - ballot information is prepared on the election management system at election central
  - ballot information may be directly entered into the DREs, or it may be written onto DTDs and sent to the polling places
  - if paper ballots are used, they are prepared at election central

- On election day, prior to the start of the voting:
  - if not yet initialized at election central, then DREs and optical scanners are initialized with the appropriate ballot information at the polling site using the DTDs
  - pre-election logic and accuracy testing (pre-LAT): testing with sample votes to see if DREs record everything accurately and scanners work properly
Election process (cont’d)

- When a voter comes to the polling place:
  - he/she registers at a desk
  - the voter is given a PIN or a token (e.g., a smart card) to insert into the DRE to start voting
  - the voter’s selections are displayed on the DRE screen and are also printed on the VVPAT

- If paper ballots are used:
  - the voter is given a ballot and a marking device to cast his/her vote
  - the ballot is handed over to an official
  - the ballot is inserted into the optical scanner to be read and recorded
Election process (cont’d)

- After the election is closed:
  - the results from each of the DREs and scanners at a polling place are collected on a DTD and returned to election central
  - DTDs are read into the election management system to produce a tally for the entire area
Typical problems with DRE systems

- poor integration
  - the system is put together by integrating election components created by different companies or groups
  - often there is no overall system design and no coherent structure

- inappropriate use of cryptography
  - three types of usage: naive, incorrect, or no use at all
  - typical mistake: data on a DTD is encrypted but the encryption key is stored in the clear on the same media
  - other frequent mistake: no cryptographically strong message authentication mechanism is used
Typical problems with DRE systems

- data forgery and lack of input validation
  - components often assume that input data comes from a specific system component, disregarding the fact that in many cases it could easily be forged
  - when data is expected to come from a trusted component, it is not validated → possibility for buffer overflow attacks

- flawed certification process and lack of security testing
  - certified systems often fail in practice when analyzed by independent experts
  - the certification process usually validates the proper functioning of a voting system under ideal conditions
  - real-world deployments often rely on operational procedures, but security should not depend only on those procedures (procedures often underestimate the power of potential adversaries)
Viral infection of the entire system

- the attacker drops an infected DTD into the pool of DTDs (note: DTDs are often reused from one election to the other)
- when preparing the ballot, the bad DTD infects the EMS system (usually run on Windows)
- EMS system infects all DTDs when loading ballot information
- infected DTDs are inserted in DRE devices which can also be infected
- one cannot be sure that the firmware and software running on DRE devices are intact
  - a corrupted DRE may modify a subset of ballots as they are cast
  - it may insert fake ballots with votes for the preferred candidates
  - it may not properly record votes
  - it may simply report a wrong voting summary to election central
Case study

- Diebold AccuVote-TS voting machine
  - Windows CE 3.0
  - Software: C++
  - Touch screen
  - Smart card reader (COM)
  - (Internet or phone connection)

- used in US presidential elections
- system has been analyzed by independent experts, and an alarming large number of problems were identified!!!
Diebold case study: System operation

setup
- prior to an election, the voting terminals must be configured and installed at each voting location
- ballot definitions are distributed to voting terminals
  - removable media, such as floppy disks or storage cards
  - or over a local network, the Internet, or a dial-up connection

election
- a voting card (memory card or smartcard) is given to each voter at the voting site
- voter inserts the voting card into the smartcard reader attached to the voting terminal
- the terminal checks that the smartcard in its reader is a voter card and, if it is, presents a ballot to the voter on the terminal screen
- the voter interacts with the voting terminal, touching the appropriate boxes on the screen (headphones and keypads are available for visually-impaired voters to privately interact with the terminal)
- the voter is given a final chance to review his or her selections
- if the voter confirms this, the vote is recorded on the voting terminal and the voter card is canceled
- the voter returns his or her canceled card to the poll workers, who reprogram it for the next user
Diebold case study: Smart cards

- the smartcards do not perform any cryptographic operations
  - there is no secure authentication of the smartcard to the voting terminal
  - nothing prevents an attacker from using his or her own homebrew smartcard in a voting terminal

- feasibility of creating home made smart cards
  - user-programmable smartcards and smartcard readers are available commercially over the Internet in small quantities and at reasonable prices
  - an attacker who knows the protocol spoken between voting terminals and legitimate smartcards could easily implement a homebrew card that speaks the same protocol

- each voting terminal will make sure that the smartcard has encoded in it the correct m_ElectionKey, m_VCenter, and m_DLVersion variables
  - m_ElectionKey and m_DLVersion are likely the same for all locations
  - m_VCenter value could be learned on a per location basis by interacting with legitimate smartcards from an insider, or from inferences based on the m_VCenter values observed at other polling locations
Diebold case study: Smart cards

- Deactivation of a voter card after the voting actually occurs by rewriting the card’s type, which is stored as an 8-bit value on the card:
  - the adversary could program a smartcard to ignore the voting terminal’s deactivation command \( \rightarrow \) then, she can vote as many times as she wishes
  - no information is recorded about the voter cards \( \rightarrow \) counterfeit votes cannot be identified later
  - if the number of collected votes becomes greater than the number of people who showed up to vote, then election results must be ignored (or election repeated)

- If a malicious voter entered an administrator or ender card into the voting device, then the voter would be able to terminate the election (DoS):
  - the voting terminal will no longer accept new voters
  - such an attack, if mounted simultaneously by multiple people, could temporarily shut down a polling place
  - even if the poll workers were later able to resurrect the systems, the attack might succeed in deterring a large number of potential voters from voting

- In order to create an administrator card, the attacker must know the PIN associated with administrator cards:
  - such PINs are stored on admin cards and sent from the smartcard to the terminal in cleartext
  - it is enough to interact with an admin card to learn its PIN
Diebold case study: Data storage

- **data storage**
  - each voting terminal has two distinct types of internal data storage
    - main storage area contains the terminal’s operating system, program executables, static data files (e.g., fonts), and system configuration information + backup copies of dynamic data files (voting records and audit logs)
    - a removable flash memory storage device is used to store the primary copies of these dynamic data files
  - on Windows CE, removable storage cards are mounted as subdirectories
    - voting software checks the existence of the “Storage Card” subdirectory in the filesystem’s root directory
    - if one creates a subdirectory with this name, then the software believes that the removable memory device is there, while it is not → dynamic data is not stored redundantly

- **data transport**
  - election results can be sent to a back-end post-processing server over a network connection or by transporting the removable flash memory
  - physical card transportation must be robust against real-world analogies of network man-in-the-middle attacks
    - any flaws in the policies and procedures used could lead to opportunities for these cards to be read or written by an adversary
Diebold: Tampering with the system config

- the majority of the system configuration information for each terminal is stored in the Windows registry
  - all of the configuration information is stored in the clear, without any form of integrity protection
  - all an adversary must do is modify the system registry to trick a given voting terminal into effectively impersonating any other voting terminal
  - it is unclear how the tabulating authority would deal with results from two different voting terminals with the same terminal ID

- each terminal keeps track of the total number of votes that have ever been cast on it — the “Protective Counter”
  - the counter is simply stored as an integer in the file system.bin in the terminal’s system directory
Diebold: Tampering with ballot defs

- the ballot definition is contained in the file election.edb, which is not encrypted nor check-summed (cryptographically or otherwise)
  - an attacker could add, remove, or change issues on the ballot, and thereby confuse the result of the election
  - simply changing the order of the candidates as they appear in the ballot definition will change the results
    - candidate information itself is not stored in the results file, it merely tracks that candidate 1 got so many votes and candidate 2 got so many other votes
Diebold: Cryptographic issues

- Vote records and the audit logs are encrypted and check-summed before being written to the storage device.

- Key management issues:
  - All the data on the storage device is encrypted using a single, hardcoded DES key:
    
    ```c
    #define DESKEY ((des_key*) "F2654hD4")
    ```
  - This is not a hexadecimal value but a bit string.
  - This particular key has been used without change since December 1998.
  - In a similar fashion, the terminal uses a hard-coded password ED 0A ED 0A ED 0A ED 0A ED 0A to authenticate itself to the voter and admin cards.

- Encryption:
  - DES is used in CBC mode, but the implementation here always uses all-zero as IV.

- Checksum:
  - 16-bit cyclic redundancy check (CRC).
  - CRC is an unkeyed, public function → it does not provide any meaningful integrity protection for the data.
  - CRC is stored in clear along with the ciphertext in the file → it leaks information about the plaintext.
Diebold: Cryptographic issues

- random number generation
  - the pseudorandom number generator used to generate serial numbers for vote records is not cryptographically secure (linear congruential generator)
  - from comments in the code, it appears that they have chosen this because it appeared in *Applied Cryptography* even though in the same work it is advised that such generators should not be used for cryptographic purposes

- linking voters with their votes
  - each vote is written *sequentially* to the file recording the votes
  - a poll worker could surreptitiously track the order in which voters use the voting terminals
  - later, in collaboration with other attackers who might intercept the “encrypted” voting records, the exact voting record of each voter could be reconstructed
any serious software engineering endeavor should have extensive design documents that specify how the system functions, with detailed descriptions of all aspects of the system, ranging from the user interfaces through the algorithms and software architecture used at a low level

no such documents can be found, and no references to any such documents in the source code exist (despite references to algorithms textbooks and other external sources)

some example comments:

/* Okay, I don’t like this one bit. Its really tough to tell where m AudioPlayer should live. [...] A reorganization might be in order here. */

/* This is a bit of a hack for now. [...] Calling from the timer message appears to work. Solution is to always do a 1ms wait between audio clips. */

/* need to work on exception *caused by audio*. I think they will currently result in double-fault. */
Conclusions

- DRE voting systems:
  - little difference appears in the way code is developed for voting machines relative to other commercial endeavors
  - an open process would probably result in more careful development, as more scientists, software engineers, political activists, and others who value their democracy would be paying attention to the quality of the software
  - open source would not solve all of the problems (it would still be important to verify somehow that compilers are non-malicious and the binary program images correspond to the source code), but open source is a good start
  - proper solution probably needs tamper resistant module in the voting terminal to store secrets such as crypto keys used for encrypting stored votes
  - malicious terminal/interface problem is still tough!
  - Voter-Verified Audit Trail → produce a paper trail that can be seen and verified by a voter → in such a system, the correctness burden on the voting terminal’s code is significantly less

- Internet voting:
  - neat cryptographic solutions, but end-systems are too vulnerable to all kinds of (non-cryptographic) attacks
  - the stakes are at very high → massive attacks are possible, or even likely