

Designing for Security: Secure Voting, Virtual Machines, and Smart Grids

Edoardo Biagioni
University of Hawaii at Manoa
esb@hawaii.edu

Joint work with:
Todd Baumeister, Yingfei Dong, Yoshiaki Iinuma,
Wes Peterson, Kazuo Sugihara

Designing for Security

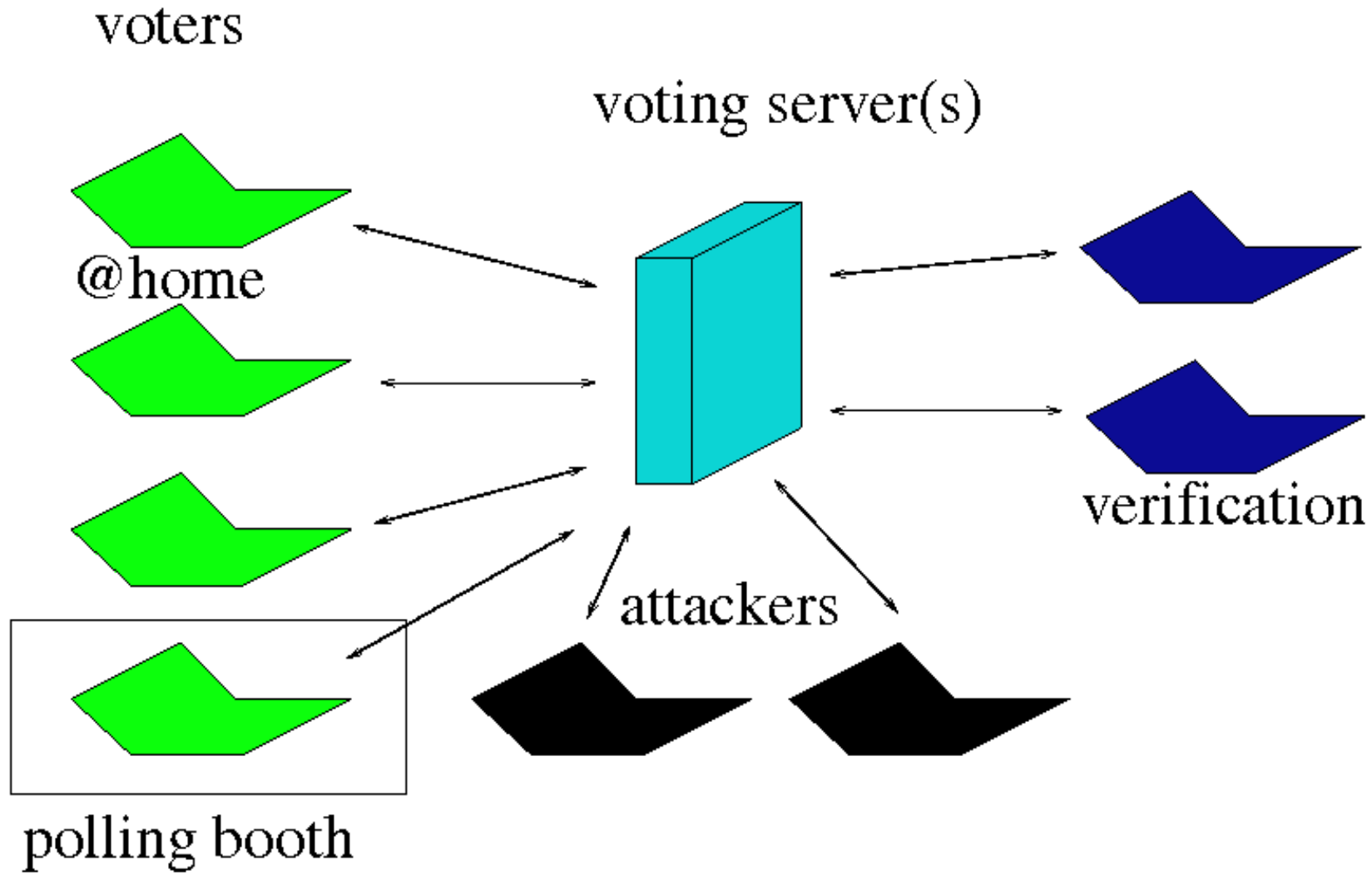
Outline

- Introduction
- Secure Voting System
- Virtual Machine for Operating System security
- Smart Grid Public Key Infrastructure
- Final Considerations

Designing for Security

- There are many insecure systems
 - Security was too hard, so the original Internet was designed without much security
- Multics (1960s/1970s) was designed to be secure
- But still had many vulnerabilities
- As seen with Windows, it is hard to add security after a product is delivered
- What would modern systems look like if security was designed in from the beginning?

Distributed Voting Application



Goals of secure voting system

- Conduct the election successfully:
 - Legitimate votes are counted
 - No illegal votes are counted
 - Resistant to external attacks
 - Resistant to internal attacks
- Allow voter verification of voter's own vote, and of vote totals
- Distributed servers should be administered by independent entities

Secure Distributed Voting

- Votes are always sent encrypted
- Signing server(s) know who may vote
- Voter submits blinded vote to signing server(s)
- Each signing server allows one vote per voter
- After unblinding, any vote signed by a majority of the signing servers is valid
- Vote must include a random unique ID
- Any number of tallying servers can record votes
 - Compromising any single server is insufficient to affect the election

Security properties

- A majority of signing servers must be compromised in order to create votes
- A majority of signing servers must be shut down to give denial of service
- A signed vote is always valid:
 - A vote stored in any server can be counted
 - Votes stored by NGOs can be compared to votes in the official tally
- Tallying servers cannot delete votes, and cannot create valid votes

Networking Protocol

- Minimize exposure to denial of service attacks
- Assume there is enough bandwidth that it is not necessary to do congestion control
- If so, a stateless request/reply protocol, with regular retransmission, is resistant to denial of service attacks
- Signing servers and tallying servers must be able to quickly discard invalid votes

Protecting Signing Servers

- Each voter has a public ID and a secret credential
- Vote sent to signing server includes ID, vote, and hash of message and secret credential
- Signing server can quickly check hash, discard invalid messages or new messages from same voter
 - 730MHz PC could discard 78Mb/s of DoS traffic and still answer one in 7.4 legitimate requests

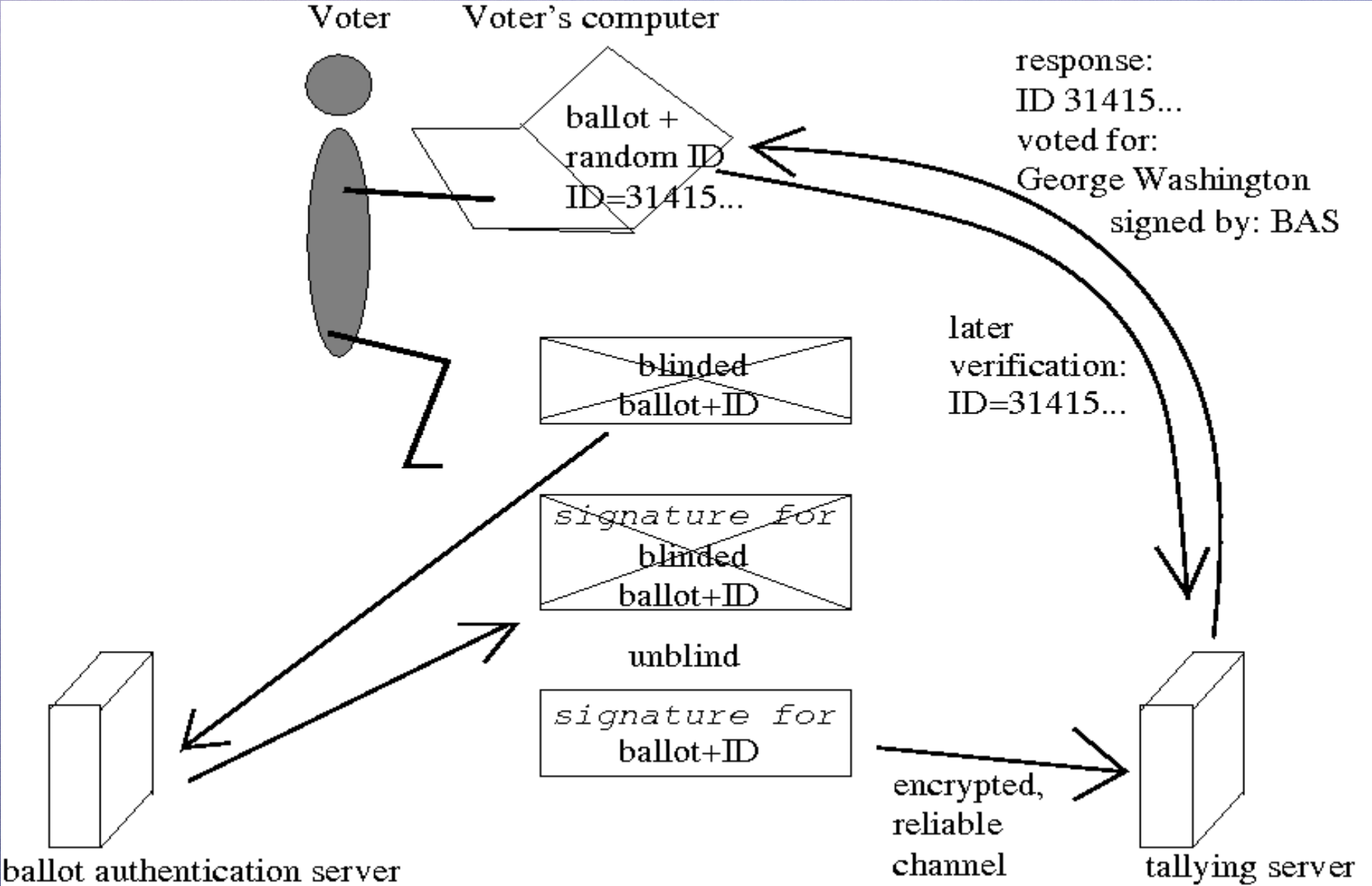
Protecting Tallying Servers

- The vote is encrypted for anonymity, using the tallying server's public key
- The tallying server must decrypt, then verify the signatures
- This is slow, but can be done massively in parallel, independently for each vote
- 730MHz PC, 9Mb/s DoS traffic, approximately one in 10.3 valid votes was acknowledged

Implementing a fast Signing Server

- Voting data is relatively small: less than a Gigabyte per million voters
- Must accomodate peak traffic
- Essential data structures in memory, backed up to disk
- On reboot, reload state saved on disk
- Fast checks to eliminate bad incoming packets
- Duplicate packets elicit duplicate responses

Summary of Distributed Voting



Virtual Machine Monitor for Malware Detection

- Run OS inside a virtual machine
- The virtual machine can observe the kernel data structures, and see if they change
- Virtual memory techniques make this efficient
- The idea is not new, and new variants keep appearing in the literature
- Why is this useful?

Theory of Operating Systems

- The operating system provides processes with a protected environment
- The OS uses the same mechanisms to protect its own data structures
- This provides protection as well as flexible sharing



- win-win!
- Why then monitor the operating system?

Operating Systems Practice

- Sharing requires careful user involvement
- OS does not protect against device drivers (kernel modules, window system)
- OS is too complicated to be correct (4MLoC in Linux kernel)
- With hardware assist, Virtual Machine can be simple, and is more likely to be correct
- Virtual Machine does not provide sharing (which may lead to its being circumvented)
- Guest device drivers are unprivileged

Operating Systems Redesign

- More careful balance between protecting and sharing
- Simple, secure “kernel”
- No privileged execution of user code, device drivers, window systems
- A Virtual Machine does this with acceptable performance, so an OS should be able to also!
- Should the file system be privileged? The network interface? The mouse and keyboard?

Microkernels vs. Virtual Machines

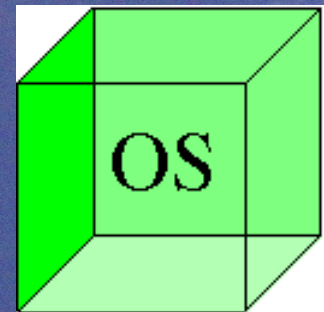
- Simple, secure “kernel”
- Mach, Minix, even OpenBSD
- Still privileged execution of device drivers
- The advantage of using a Virtual Machine is in not having to re-implement the OS functionality
- and in relying on the machine architecture

Limitations of Virtual Machines

- Even with hardware assist, still some performance impact
- Software (malware) can detect that it is running on a virtual machine
- More difficult sharing
- Monitoring software must introspect the OS data structures and code locations, which is guest-OS dependent
- Main remedy in case of malware detection is to crash the OS

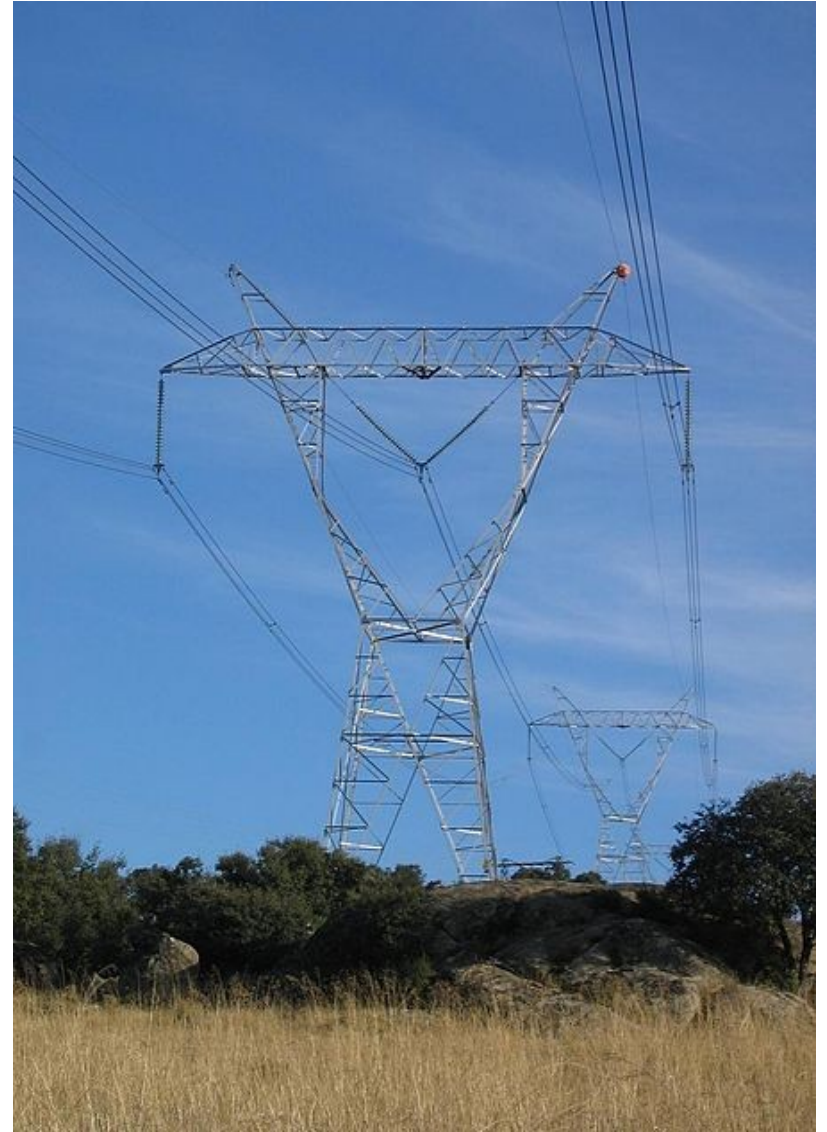
Summary of Virtual-Machine based security

- Modern operating systems are too complicated to be secure (is this true?)
- Modern operating systems are too vulnerable (is this true?)
- Putting the OS in a box is a good way to:
 - have our cake (full functionality), and
 - eat it too (monitoring for malware)
- Hardware assist makes this feasible



Smart-Grid Security

- Electricity distribution used to be conductors, switches, and relays
- More and more, computers are being used to monitor and control the flow of electricity
- security matters!



Smart-Grid Security Challenges

- The generator may not be the distributor
- Customers may have a choice of suppliers
- Who should get what data?
- Should thieves know when my lights are on?
- To whom does the data belong?
- Can one competitor learn about another competitor's pricing?
- Can one competitor shut down another?

Public Key Infrastructure Architecture

- The owner of each device must determine who has access to it
- Principals are identified by public keys
- The PKI Architecture allows certification of keys
- Every power company may want to issue keys
- How do power companies authenticate to each other?
 - *directly*: mesh architecture
 - through an *intermediary*: bridged architecture

Bridged Architecture

- Each organization can have its own PKI trust model
- The centralized bridge can enforce common security policy
- The bridge can be operated by a central authority or an industry group

Smart Grid Security

- These mechanisms must be used as part of coherent policies
- For example, only the electricity supplier(s) has access to a customer's data
- Customers should have unlimited access to their own data (can they post it on the web?)
- Only the owner of the equipment should be able to control that equipment (or delegate control)
- Current security is like OSs 20-30 years ago

Design of a Secure Grid

- Determine who should have access to what data, and who can control what equipment
 - can the power company turn off my fridge?
 - if they don't do it for too long?
 - if they give me a discount?
- Find a way to identify individuals and groups:
 - transfer of ownership, key revocation, lost key recovery
 - the mechanism cannot be too expensive
- Answering these policy questions should give a coherent design

Towards a Secure Grid

- Create security protocols to implement the design
- Lack of electricity can vary between inconvenient and life-threatening
- If there is no network connectivity, the electricity should still flow
- Hard problem: there must be a mechanism to recover lost keys, but it should not be available to thieves

Secure Grid – summary

- The smart grid can improve the efficiency of electric usage, and integrate smaller-scale generation of electricity
- But it must be secure
- Insecurity can lead to:
 - destruction of equipment (Stuxnet)
 - loss of property (theft) and other damages
 - changes in the power relationship between customers and suppliers



Designing for Security – summary

- Often, the design of a more secure system is nontrivial
- Secure design may require thorough redesign
- There is always tension between security and convenience, privacy and customization



Designing for Security – summary

- Our voting system protects the voter's anonymity, but only as long as the voter is voting alone and without supervision
- and the voter's IP address is not traceable
 - choices have to be made
- The public may be more likely than the manufacturers to choose security
- Progress is fast, so ethics must keep up with technology (as is very common today)