Winterschool NISNet-Finse 25-30 April 2010

Tuesday April 27 9:00-12:30h

Electronic Voting

Berry Schoenmakers
Coding & Crypto group
Department of Mathematics & Computer Science

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Outline

- **Evoting and cryptography:**
  - **Intro & high-level approach**
- **Building blocks**
  - **Classification of approaches**
  - **Overview of (cryptographic) primitives**
- **Example voting schemes**
- **Specific protocols:**
  - **Interval proofs**
  - **Client/server tradeoffs**

Throughout, we consider national/general elections, hence high(est) security level.

Introduction

Voting & Cryptography

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

The Software Crisis

- Coined by F.L. Bauer in 1968; publicised by Edsger W. Dijkstra in his Turing award lecture in 1972.
- Symptoms:
  - Projects running over-budget.
  - Projects running over-time.
  - Software was of low quality.
  - Software often did not meet requirements.
  - Projects were unmanageable and code difficult to maintain.
- New software engineering methodologies tend to help, no 'silver bullet' though
  - even to this day

Source: Wikipedia

Recent developments … DREs Direct-Recording Electronics

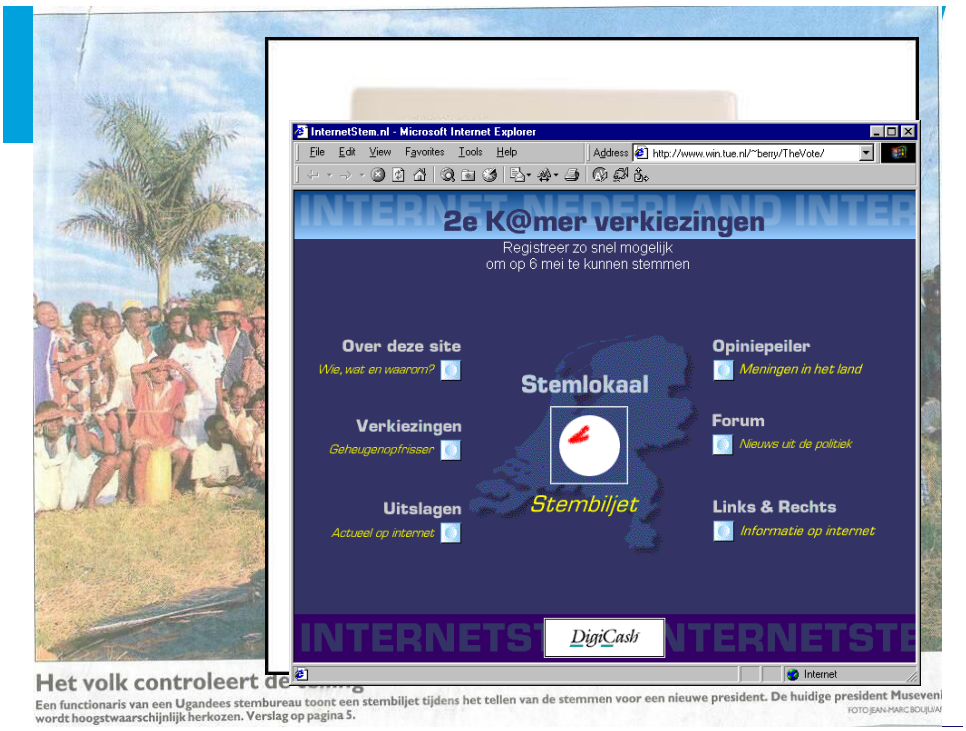
- US (California), August 3, 2007
  - Secretary of State Bowen decertifies all electronic voting systems for California
    - Diebold, Hart Intercivic, and other systems
- Netherlands, September 27, 2007
  - Secretary of State Bijleveld decertifies all electronic voting systems for the Netherlands
    - percentage e-voting grew to close to 100% !
    - NedAp and NewVote SDU machines
- Critical evaluations revealed lots of problems
  - Prior certification apparently meaningless!!
- Back to paper-based elections?!
  - Currently, in the Netherlands, parts of Germany, and other places (e.g., things in England got rather messy too)

The E-Voting Crisis

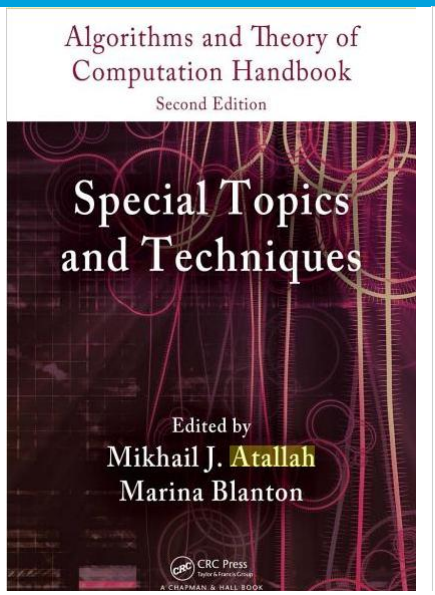
- Symptoms:
  - E-voting laws apparently meaningless, under-developed
  - Certification apparently meaningless, not strict enough
    - Voter confidence seriously hit, or indifference?
  - Implementations of systems of low quality
  - Management/operation of systems poor
  - No consensus among scientists; argue (or, fight?) with each other – what should policy makers make of this?
  - Lack of international standards and open specifications

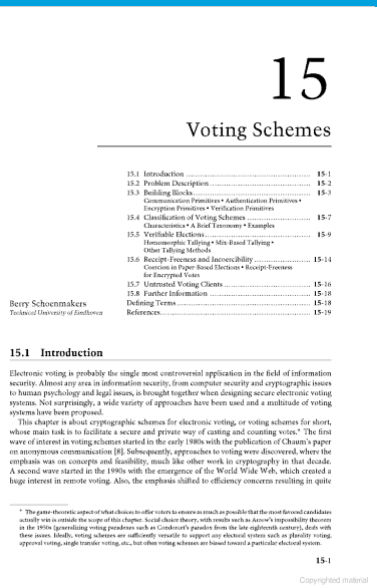
Nature of E-Voting

- **E-voting: probably single most controversial application in the field of information security**
  - **involves almost any area in information security, from *computer security* and *cryptographic issues* to *human psychology* and *legal issues***
  - **a minefield of paradoxes**
  - **a truly wide variety of approaches and e-voting systems have been proposed**



Recently appeared handbook chapter





15

Voting Schemes

Berry Schoenmakers
Technical University of Eindhoven

15.1 Introduction

Electronic voting is probably the single most controversial application in the field of information security. Almost any area in information security, from computer security and cryptographic issues to human psychology and legal issues, is brought together when designing secure electronic voting systems. Not surprisingly, a wide variety of approaches have been used and a multitude of voting systems have been proposed.

This chapter is about cryptographic schemes for electronic voting, or voting schemes for short, whose main task is to facilitate a secure and private way of casting and counting votes.* The first wave of interest in voting schemes started in the early 1980s with the publication of Chaum's paper on anonymous communication [8]. Subsequently, approaches to voting were discovered, where the emphasis was on concepts and feasibility, much like other work in cryptography in that decade. A second wave started in the 1990s with the emergence of the World Wide Web, which created a huge interest in remote voting. Also, the emphasis shifted to efficiency concerns resulting in quite

* The game-theoretic aspect of what choices to offer voters to ensure as much as possible that the most favored candidates actually win is outside the scope of this chapter. Social-choice theory, with results such as Arrow's impossibility theorem in the 1950s (generalizing voting paradoxes such as Condorcet's paradox from the late eighteenth century), deals with these issues. Ideally, voting schemes are sufficiently versatile to support any electoral system such as plurality voting, approval voting, single transfer voting, etc., but often voting schemes are biased toward a particular electoral system.

Electronic elections

- **Solve security and privacy issues:**
  - **By trust?**
  - **By legal measures?**
  - **By technology? Yes, using cryptography!**
- **Cryptographic approaches to electronic elections have been studied since early 80s, in three waves:**

1980
- **Chaum on anonymous communication (MIXes), J. ACM 1981**
  - **Concepts and feasibility of voting protocols**

1990
- **emergence of the World Wide Web, huge interest in remote voting. Emphasis shifted to efficiency concerns, resulting in quite practical schemes.**

2000
- **the `Florida 2000' US election fiasco, which renewed interest in voting from polling stations, leading to voting schemes which combine cryptographic and physical aspects.**

"Cryptography for Democracy"

- **Focus on role of cryptography**
  - **classical cryptography**
    - **basic authentication and encryption techniques**
  - **modern cryptography**
    - **zeroknowledge proofs, secret sharing, commitments, secure multiparty computation, blind signatures**
    - **good for resolving paradoxes**

- **Cryptography helps to *isolate* and to *solve* particular *subproblems.***

Remote vs polling station e-voting

- **In both cases, encrypted votes can be used**
  - **tallied without revealing individual votes**

- **Remote e-voting:**
  - **voters themselves compute encrypted votes**
  - **can be fully electronic**

- **Advanced polling station e-voting:**
  - **authorities compute encrypted votes**
  - **typically voter marks vote on some physical (paper) ballot (Pret-a-Voter / Punchscan)**

Solution = Scheme + Infrastructure

- **Voting scheme:** cryptographic core of the system, protects even against insiders (who run the system)
- **Security infr** o stop a multitude of a
  - **Security of c** ers
  - **Security of (** are
  - **Security of c** hese computers
  - ……
- **Shortcomings of a cryptographic scheme, in particular, the lack of universal verifiability, cannot be remedied by strengthening the security infrastructure**

In the end, a risk analysis should determine the residual risk

Six commandments (M. Shamos '93)

I. **Thou shalt keep each voter's choices an inviolable secret.**
II. **Thou shalt allow each eligible voter to vote only once, and only for those offices for which she is authorized to cast a vote.**
III. **Thou shalt not permit tampering with thy voting system, nor the exchange of gold for votes.**
IV. **Thou shalt report all votes accurately.**
V. **Thy voting system shall remain operable throughout each election.**
VI. **Thou shalt keep an audit trail to detect sins against Commandments II-IV, but thy audit trail shall *not violate Commandment I*.**

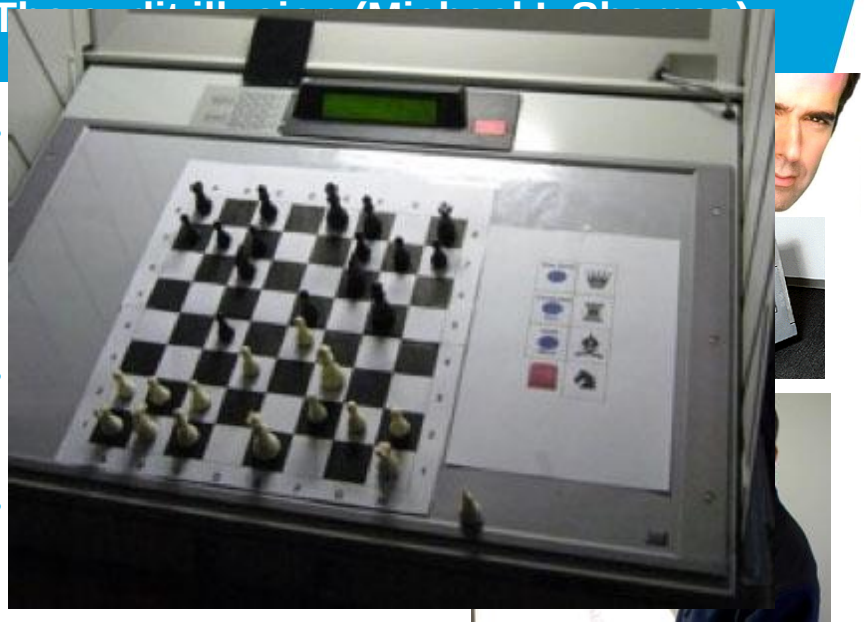
Old, informal list. But do we really have a better one ?!

Hard nut to crack

- Privacy and verifiability at the same time
- Ballot Secrecy:
  - even when the system is fully audited, all individual votes should remain private
    - vs. outsiders *and* insiders
- Universal Verifiability:
  - anyone (incl. observers, auditors) is able to verify the integrity of the election result against the encrypted votes cast by legitimate voters

Two “opposite” approaches

- Go extremely “small”:
  - special-purpose computer, to execute nothing else than the voting protocol
  - master key stored inside
- Go distributed (and scalable):
  - multiple voting servers & talliers
    - threshold security
    - extreme case: “board room elections”
      - all voters are talliers too (like in peer-to-peer systems)
      - currently not realistic for performance reasons



Scalable, distributed trust

- **Single tallier sees everything:**

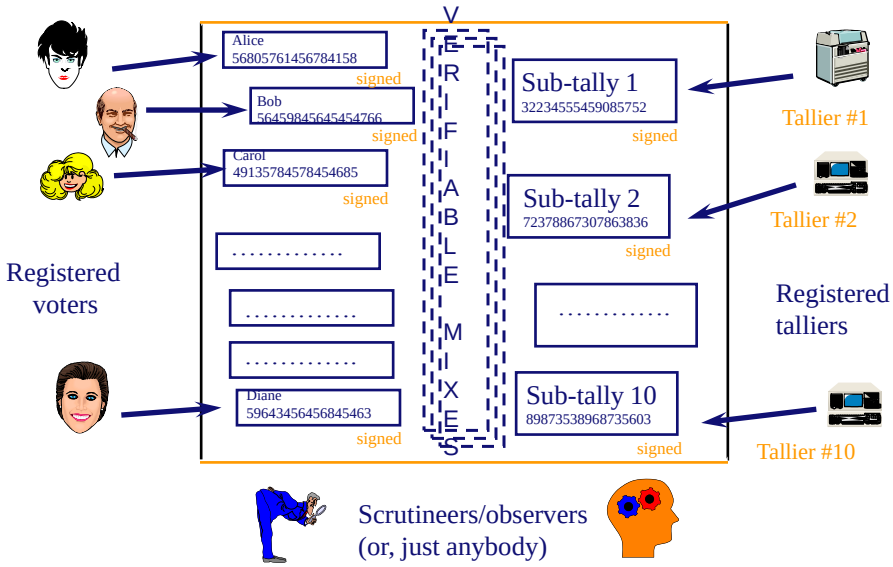
		Tallier
Alice	Yes	1
Bob	No	0
Carol	Yes	1
Diana	No	0
Total		2

- **Random split between two talliers:**

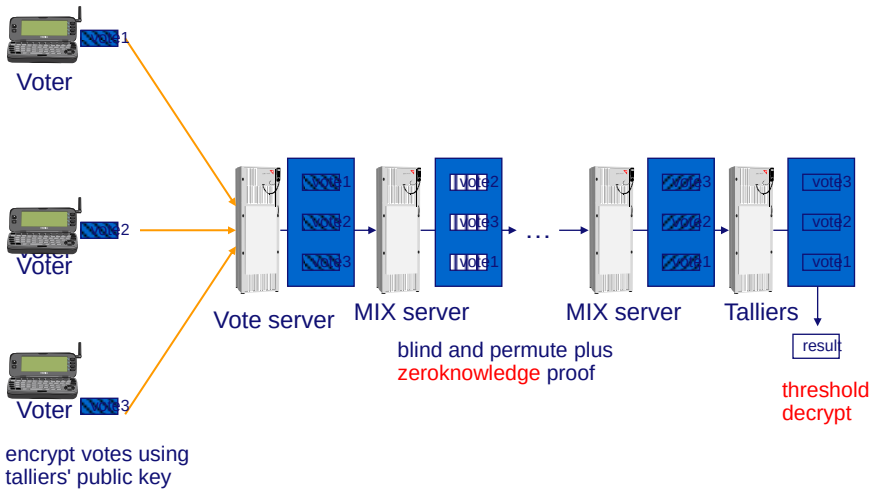
		Tallier 1	Tallier 2	
Alice	Yes	-1287	+1288	
Bob	No	-1999	+1999	
Carol	Yes	-769	+770	
Diana	No	-1334	+1334	
Total		-5389	+5391	+2

- **Number of talliers can be varied *independently* of other characteristics of the voting system**

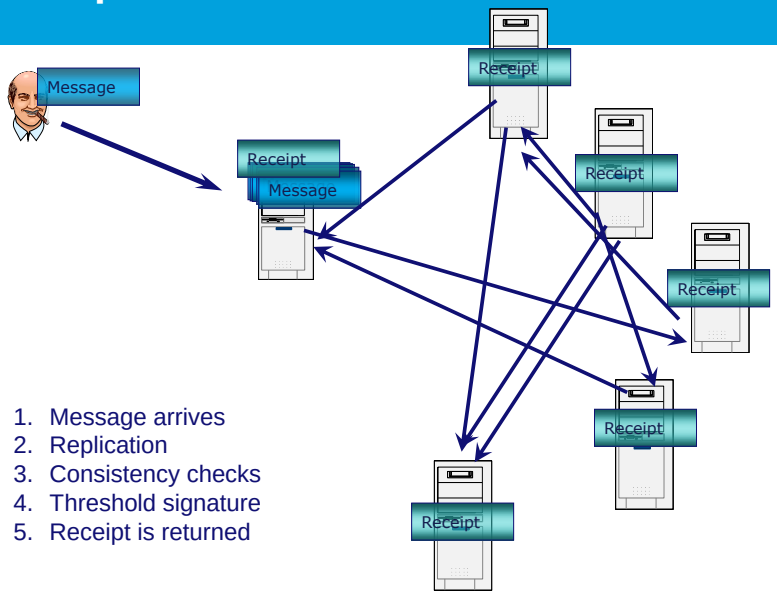
Bulletin Board model



Verifiable MIXes



Implementation of BB

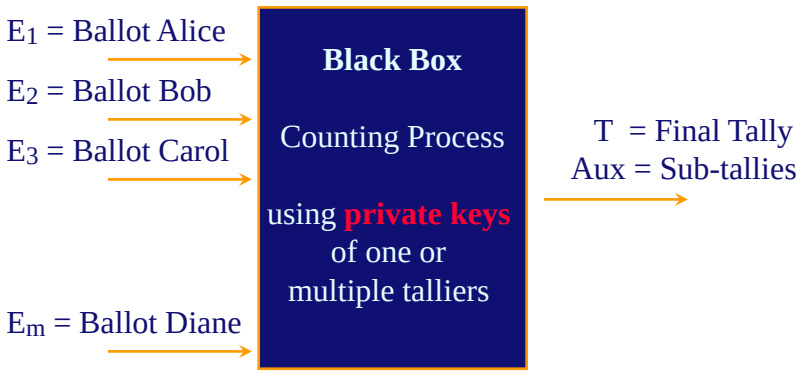


1. Message arrives
2. Replication
3. Consistency checks
4. Threshold signature
5. Receipt is returned

Bulletin Board

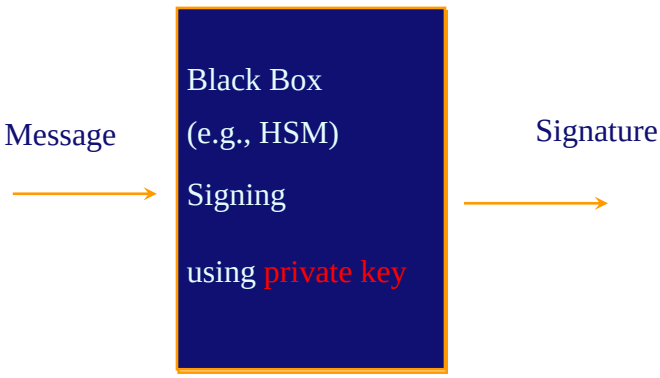
- **Properties (public broadcast channel):**
  - **Anyone can read BB**
  - **Nobody can erase anything from BB**
  - **Voters, talliers, officials write ballots to their own sections, signed with their public keys**
  - **BB produces signed receipts (threshold signature)**
    - **VOTE & VERIFY & GO PROPERTY**
      - **if receipt OK, the protocol guarantees that the vote will be counted as cast**
      - **no further checking by voters afterwards**

Universally Verifiability of Elections



Verify (E_1,…,E_m, T, Aux, **public keys of talliers**) = accept or reject

Universal Verifiability of Digital Signatures



Verify (Message, Signature, public key of signer) = accept or reject

Inside the verifiable 'tally' black box

- Essentially, a secure multiparty computation
  - input: encrypted votes $E(x_1),\ldots,E(x_n)$
  - output: election result $f(x_1,\ldots,x_n)$
- Note: election result need not contain the individual tallies of candidates/options, e.g.:
  - outcome of referendum could be "majority says 'no' " (Shamir suggested this already in 1980s, see Benaloh's PhD thesis)
  - outcome of a parliamentary election could be the assignments of seats
  - hard/impossible for paper-based elections !!

Homomorphic vs. mix-based tallying

- Homomorphic: decrypt product of all encrypted votes
  - fast tallying
    - in total, one threshold-decryption
  - zeroknowledge proof of validity per vote
    - can be costly
- Mix/shuffle based: cascade of verifiable mixes, followed by decryption of ind. votes
  - sequential, relatively slow tallying
    - one threshold-decryption per vote
  - works for any type of vote
    - constant amount of work for voter

What about coercion?

- Two levels of ballot secrecy (privacy):
  - Level 1: ballot secrecy cannot be broken without help of voters
  - Level 2: ballot secrecy cannot be broken even with help of voters (receipt-freeness, no vote selling)
- Encryption of votes
  - Should not be deterministic:
    - From $E_{PK}(v)$ anyone can find vote *v*
  - But probabilistic:
    - $E_{PK}(v,r)$ with *r* a sufficiently long, random string
- Problem: voter reveals *r* to prove to a coercer what its vote *v* is

Use of randomizers

- Randomizer (e.g., special smart card):
  - must cooperate to cast a vote successfully
  - cannot change vote
  - adds randomness s.t. voter doesn't know *r* anymore
- Randomizers ensure receipt-freeness:
  - voter cannot prove to someone else what vote is
  - does not help against family voting
- Number of randomizers can be varied (per voter):
  - no randomizer: if coercion is considered unlikely
  - 1 randomizer: if randomizer is trusted
  - multiple randomizers: if randomizers might be corrupted

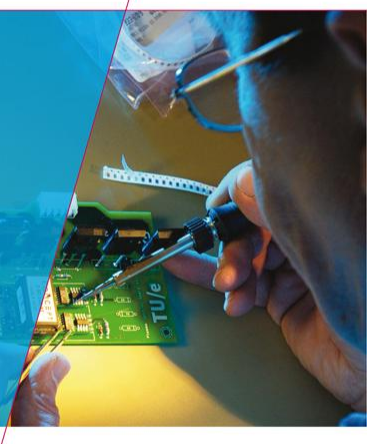
Alternative: polling-station e-voting

- Voting machine will compute encrypted vote for the voters
- Issue: how to check the voting machine?!
- Example scheme: Pret a Voter.

Summary of what it takes …

- Secure bulletin board (messages cannot be deleted)
- Use of threshold cryptography for scalable, distributed trust
  - threshold signature for receipts
  - threshold decryption of votes
- Efficient implementation of verifiable black box
- Strong voter authentication
  - preferably, assuming a PKI
- Open specifications/standards for protocols
  - independent implementations of vote clients, tally clients etc
    - required to get true "threshold cryptography"
    - not necessarily open-source (gives competitive advantage, and results of tallying are verifiable anyway)
- Also, randomizers: component to supply randomness in voting protocol, to prevent the voter being able to "open" its encrypted vote

Building Blocks



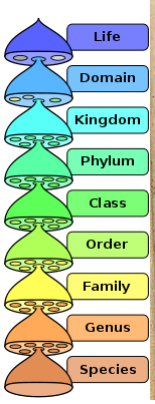
TU/e Technische Universiteit Eindhoven University of Technology

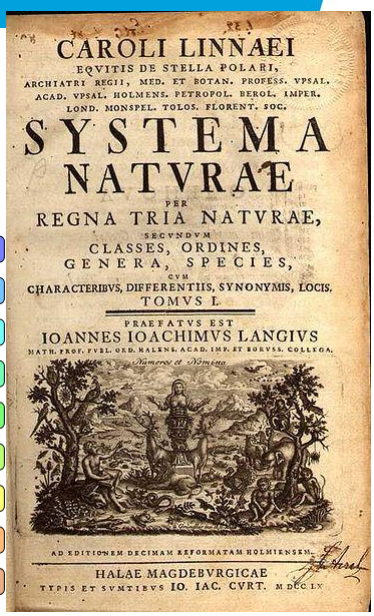
Where innovation starts

Taxonomy

Wikipedia

- **Taxonomy = practice and science of classification**
- **Classification based on suitable *characteristics***
- **E.g., kingdoms of Plantae versus Animalia and Fungi:**
  - **characteristic = "Use of photosynthesis"**




CAROLI LINNAEI
EQVITIS DE STELLA POLARI,
ARCHIATRI REGII, MED. ET BOTAN. PROFESS. VPSAL.
ACAD. VPSAL. HOLMENS. PETROPOL. BEROL. IMPER.
LOND. MONSPEL. TOLOS. FLORENT. SOC.
SYSTEMA
NATVRAE
PER
REGNA TRIA NATVRAE,
SECVNDVM
CLASSES, ORDINES,
GENERA, SPECIES,
CVM
CHARACTERIBVS, DIFFERENTIIS, SYNONYMIS, LOCIS.
TOMVS I.
PRAEFATVS EST
IOANNES IOACHIMVS LANGIVS
HALAE MAGDEBVRGICAE
TYPIS ET SVMTIBVS IO. IAC. CVRT. MDCC LX

Towards a Taxonomy of Voting Schemes

- **Characteristics:**
  - **Presence/absence "easy" to determine**
  - **Independent of security claims/properties**

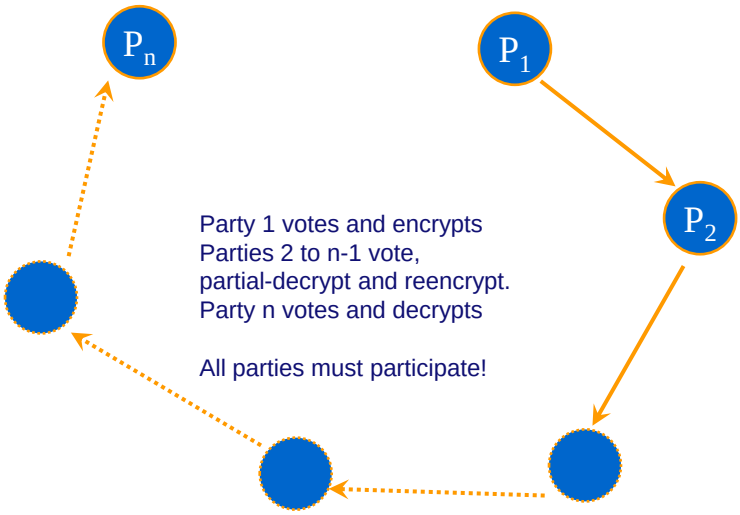
voted ballot := all information recorded upon completion of the voting protocol

- **Useful characteristics:**
  - **voting protocol requires interaction between voters, or voters can proceed in parallel**
  - **voted ballot is anonymous vs. identifiable.**
  - **voted ballot contains vote in the clear (derivable from public information only), or in encrypted form**
  - **how votes or voted ballots are authenticated**

Vote casting

Vote tallying

Vote casting and tallying combined



Building blocks

- **Primitives are divided into 4 categories:**
  - **Communication primitives**
  - **Authentication primitives**
  - **Encryption primitives**
  - **Verification primitives**
- **Most of these primitives are cryptographic primitives while some are beyond the scope of cryptography.**

Communication Primitives

- **Secure channel:** communication channel protected against eavesdropping, possibly providing end-to-end authentication as well, achieving computational security.
- **Private channel:** same as secure channel, but achieving information-theoretic security.
- **Untappable channel:** totally unobservable (out-of-band) communication channel.
- **Anonymous channel:** communication channel which hides the sender's identity; possibly allows for an acknowledgement by the receiver as well.
- **Bulletin Board:** publicly readable broadcast channel, possibly with authenticated write operations.
- Also directed links, and combinations

Authentication Primitives

- **External vs. internal voter authentication:**
  - **External authentication links with the list of eligible/registered voters**
  - **Internal authentication used in addition to external authentication; internal to the voting scheme**
- **External authentication primitives:**
  - **Anything from an electoral roll, ID cards, to symmetric crypto (incl. passwords) to asymmetric crypto (digital signatures, one-time signatures, PKI)**
- **Internal authentication primitives:**
  - **Anonymous signatures:**
    - **Blind / group / restrictive blind / ring / list signatures**
    - **signer does not learn message and signatures**

Encryption Primitives

- **Used for vote encryption**
- **Symmetric and asymmetric cryptosystems**
- **Homomorphic encryption: $E(x + y) = E(x) * E(y)$**
  - **Ex. ElGamal $E(x) = (g^r , h^r g^x)$ with r random**
- **Threshold decryption**
  - **t out of n parties must cooperate to decrypt the election result, and are assumed not to decrypt anything else**
  - **no single party ever knows the decryption key**
- **Also covers use of commitments and secret sharing:**
  - **Ex. voter commits to vote and distributes shares of vote between talliers**
    - **gives information-theoretic privacy**
    - **requires private channels**

Verification Primitives

- **Zero-knowledge proof:** proof for a statement without giving away why it is true.
- **Validity proof:** zero-knowledge proof showing the validity of an input, or step performed in a protocol.
- **Designated verifier proof**: noninteractive zero-knowledge proof bound to a particular public key which only convinces the holder of the corresponding private key.
- Examples:
  - **Voters: prove encrypted ballots are correctly formed**
  - **Talliers: prove decryption is done correctly**
  - **Voting machine gives designated verifier proof to voter**

Non-interactive zeroknowledge proofs give universal verifiability

Homomorphic encryption

- Public key cryptography:
  - Diffie-Hellman key-exchange in 1976
  - RSA encryption, RSA signatures in 1977

- "Privacy homomorphisms" in 1978 by Rivest et al.:
  - Product of two RSA encryptions corresponds to product of the two messages

$c_1 = m_1^e \bmod n$ "encryption of message m_1"

$c_2 = m_2^e \bmod n$ "encryption of message m_2"

$c = c_1 \quad c_2 = (m_1 \quad m_2)^e \bmod n$ "encryption of message $m_1 \quad m_2$"

Homomorphic encryption

WARNING

- Cryptosystems with homomorphic properties are bad
  - enable adaptive chosen-ciphertext attacks
  - make ciphertexts malleable (making known changes to an unknown plaintext)

- But these properties are very useful for cryptographic protocols when used with care
  - malleability must be stopped by other means such as zero-knowledge proofs

Homomorphic encryption

- Popular choices:
  - Goldwasser-Micali encryption 1984, RSA setting
  - ElGamal encryption 1985, Discrete Log setting
  - Paillier encryption 1999, RSA setting
- These schemes handle either addition or multiplication but not both at the same time.

- New development:

  "Fully homomorphic encryption" by Craig Gentry, 2009
  - Handles both addition and multiplication of encrypted values (or, equivalently, the NAND function)
  - Based on lattices, the "geometry of numbers"
  - Disadvantage: not efficient in practice

Secret sharing

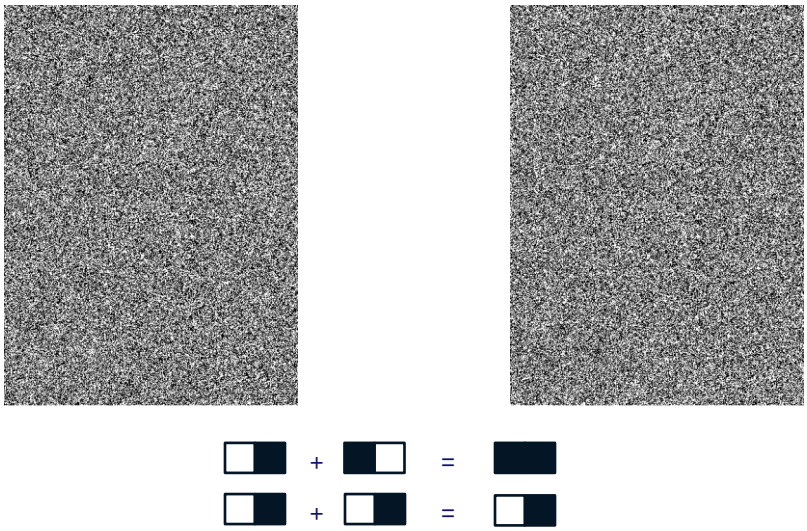
- How to split a secret value into two shares such that:
  - Individual shares do not reveal the secret
  - Combining both shares yield the secret

  E.g., a secret key 010101011111001101100001.
  Split it in half? No, bad idea.

- How to "split a secret bit" s into two shares?
- - choose a bit r uniformly at random
- - put s1 = r and s2 = r + s (mod 2)
- - then s1 and s2 are fully random on their own but when combined yield s = s1+s2

Visual Secret sharing

Naor, Shamir '97



Applications of secret sharing

- **Basically serves to avoid the use of a single trusted party:**
  - **trusted party is replaced by multiple parties of which, say, a majority is assumed to be honest.**
- **Also related to random masking as used to counter 1st –order DPA (Differential Power Analysis) attacks**

Zero-knowledge proofs

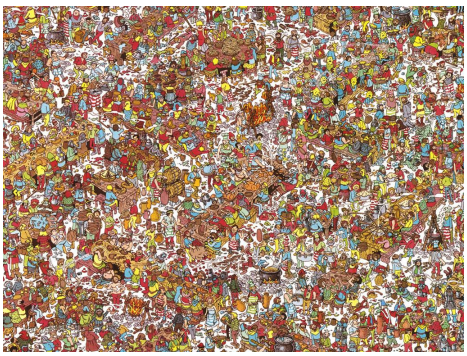
- **Q1: How to convince a (skeptical) verifier that you know the solution to a puzzle?**

  A1: Simply show the solution!

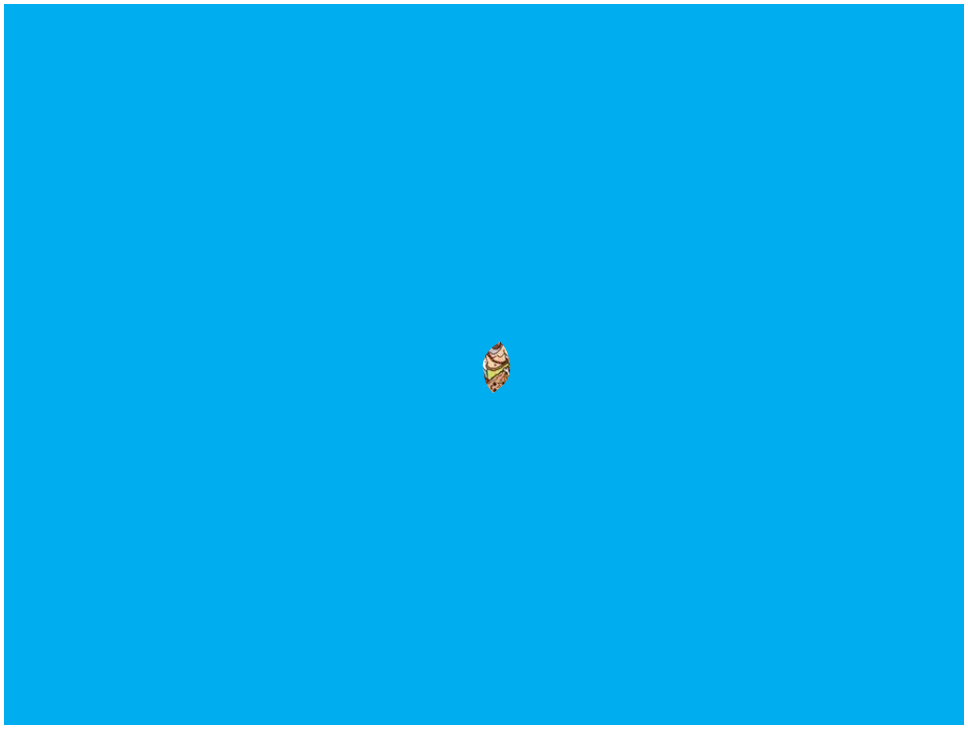
- **Q2: Same question, but now you don't want to give away any information on the solution?**

  A2: Prove in zero-knowledge that you know the solution.

Where's Waldo?

Find  **in** 

Waldo
(a.k.a. Wally)



In what sense is this zero-knowledge?

- Peephole shows nothing but Waldo, and we already know how Waldo looks.
- In other words, you could make this final view yourself without any help of the prover:
  - Stick a small picture of Waldo behind the peephole
  - In crypto speak: you can simulate the view
- **Simulation paradigm:**

"everything that the verifier learns from the proof, could be constructed (efficiently) by the verifier itself – without knowing the solution"

Soundness

- The proof should also be sound:
  - Prover only be able to convince the verifier if the prover actually knows the solution
- So, cheating as done in a simulation should be excluded:
  - make sure prover doesn't have small pictures of Waldo hidden somewhere – checking this may be awkward
- Achieving soundness & zero-knowledge can be done efficiently based on number-theoretic constructions.

How about arbitrary puzzles ?

- NP-complete problems are notoriously hard "puzzles"
  - Example: traveling salesman problem
- Proving that you know a solution amounts to proving that you know how make a certain Boolean formula φ evaluate to true

Example: $\varphi = (w \vee \neg x \vee y) \wedge (x \vee \neg y \vee z) \wedge \ldots$

Formula φ is very big, and it gets very hard to make all clauses true at the same time.

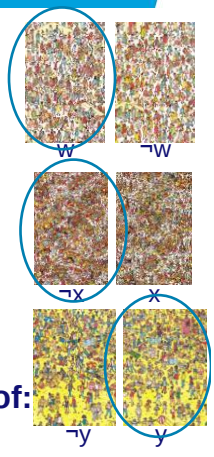
Suppose we know a solution, let's prove this in zero-knowledge – using lots of Waldo puzzles.

Boolean connectives: AND, OR, NOT

- **AND "∧"**
  - **take two puzzles**
  - **prove that you know solutions to both of them**
- **OR "∨"**
  - **take two puzzles**
  - **prove that you know solution to one of them without showing which puzzle you are solving**
- **NOT "¬"**
  - **cut one puzzle in half**
  - **prove where Waldo is in one piece**
  - **then Waldo is not present in the other piece**

Zeroknowledge satisfiability of φ

- **Let φ = (w ∨ ¬x ∨ y) ∧ (x ∨ ¬y ∨ z) ∧ … and suppose you know what value to assign to w,x,y,… to make φ = true**
- **For each variable, say w, do this:**
  - **cut a puzzle in half**
  - **label the piece where Waldo is with w if w=true, and with ¬w if w=false**
  - **label the other piece with the complement**
- **For each clause, say w ∨ ¬x ∨ y, do an OR proof:**
  - **select the pieces labeled by w, ¬x, y**
  - **Waldo is in at least one of these pieces**
  - **prove that you know where Waldo is in one of these pieces without showing which piece**



Zero-knowledge applications.

- **Secure identification: prove that you know the secret key without giving away any other information**
  - **contrasts sharply with use of passwords**
  - **more like challenge-response protocols**
- **Allows for selective disclosure of information:**
  - **prove that you are over 21, without giving any more information on your age**
- **Generally, zero-knowledge proofs are used in cryptography to let a party prove that it behaved according to the protocol**

Basic Voting Schemes

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Electronic elections

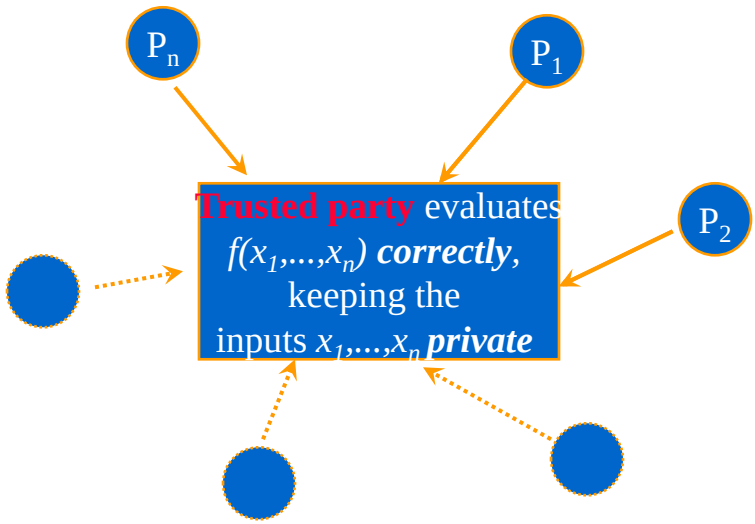
- Electronic elections form a primary example of a secure multiparty computation.
  - referendum with yes=1, no=0:

    $$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

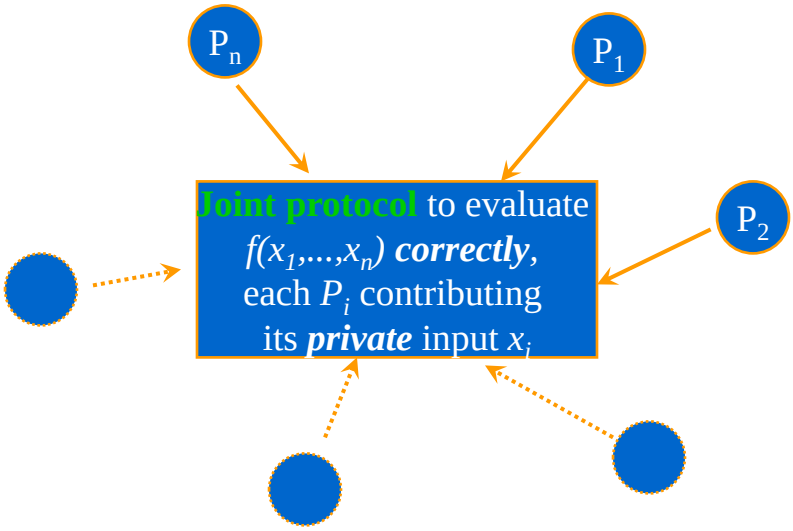
  - STV (Single Transfer Voting), where votes x_i are ordered lists of (all) candidates:

    $f(x_1, x_2, \ldots, x_n)$ = "complicated function defined in rounds determining the winners"

Secure multi-party computation



Eliminating the trusted party



Ideal voting scheme vs real one

- **Ideal voting scheme assumes trusted party. Real scheme must do without.**
- **Note that in the ideal scheme:**
  - **Votes remain absolutely private, except what is implied by the election result**
    - **Voters cannot even prove how they voted if they wanted to (receipt-freeness)**
  - **No intermediate results are revealed**
  - **By definition the election result is correct and everybody is convinced of that fact**
- **How to do this without a trusted party?!**

Problem: level of trust in insiders

- Attackers
  - Outsiders, i.e., anyone on the Internet:
    - May try to attack the SSL connection or the server.
    - Relatively easy to counter
      - basic encryption/authentication techniques
  - Insiders, i.e., those who run the election:
    - May try to alter the election result
    - May try to learn people's votes
    - Much harder to counter
      - using modern cryptographic tecniques

"Those who cast the votes decide nothing.
Those who count the votes decide everything."
Josef Stalin

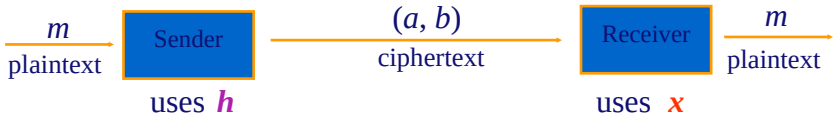
ElGamal encryption

- Receiver's private key: x
- Receiver's public key: $h = g^x$
- Sender encrypts plaintext m:

  $(a, b) = (g^w, h^w m)$,

  using a random w
- Receiver decrypts ciphertext (a, b):

  $b / a^x = m$



One-way functions

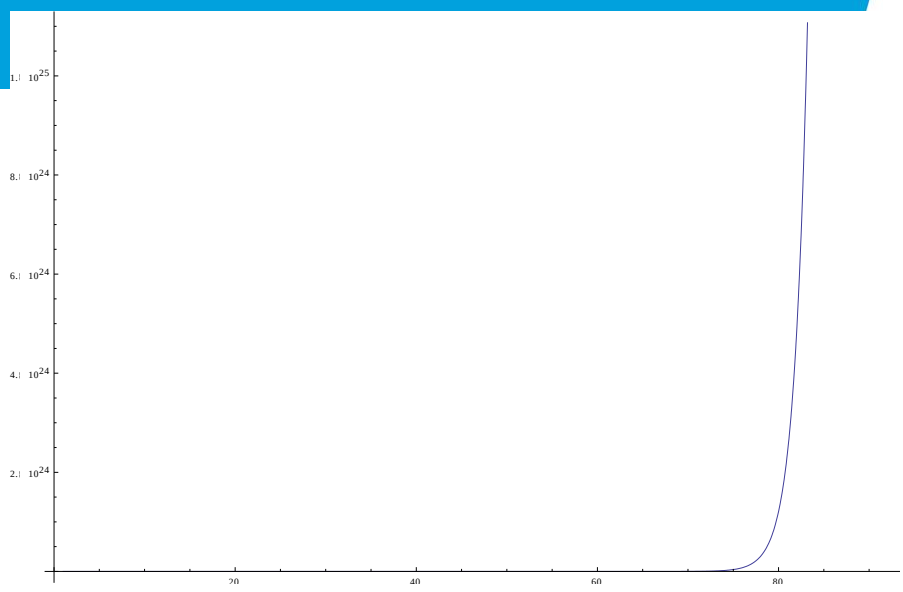
- **Example: discrete exponentiation**

$$x \rightarrow g^x$$

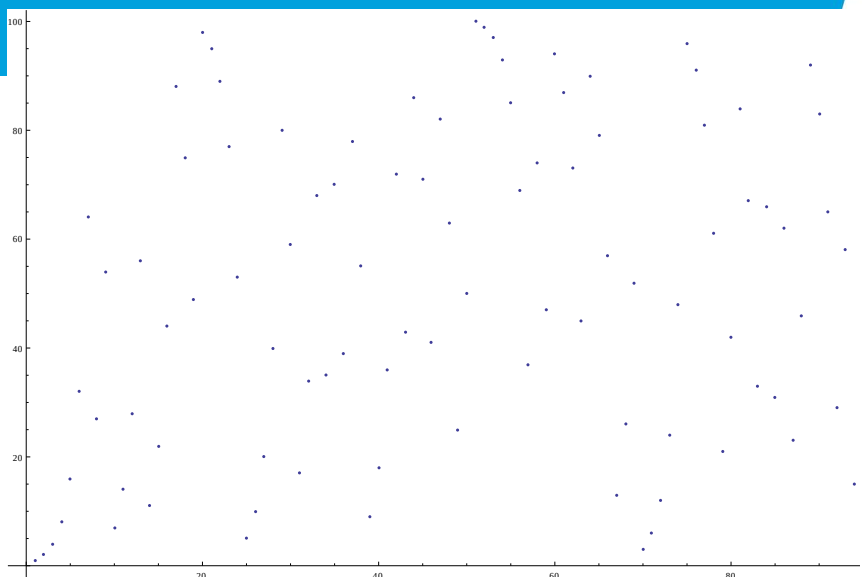
- **If well-chosen:**
  - **Function is easy to apply (forward direction)**
  - **But very hard to invert (reverse direction)**

- **Toy example:** $x \rightarrow 2^x \mod 101$

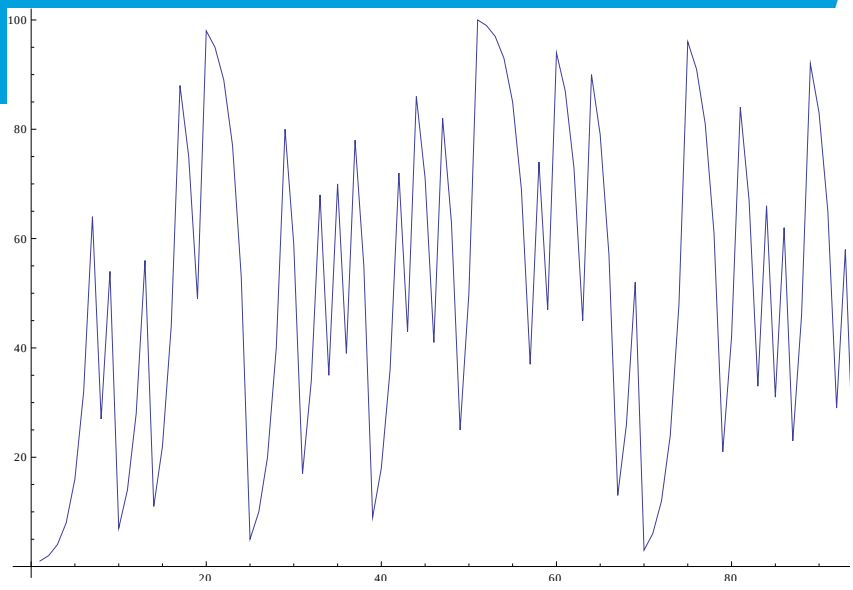
- **For cryptography:**
  - **use a very large prime p (say 1024 bits)**
  - **one can also use elliptic curves, of course.**



Plot of 2^x for x = 0 to 100



Plot of 2^x mod 101



Plot of 2^x mod 101

Homomorphic ElGamal encryption

- Consider a vote $v \in \{1,0\} \approx \{\text{yes,no}\}$
- Ballot is ElGamal encryption of vote g^v:
  $(a, b) = (g^w, h^w g^v)$,
  using a random w
- Homomorphic property:

$$(a, b) * (a', b') = (a\,a', b\,b') = (g^{w+w'}, h^{w+w'} g^{v+v'})$$

- Tallying: decrypt product of all ElGamal encryptions to find sum of votes.

Use of zeroknowledge proofs

- Question: How to prevent voters from sending in ballots like these?

$(a, b) = (g^w, h^w g^2)$	double yes
$(a, b) = (g^w, h^w g^{-4})$	-4 times yes
$(a, b) = (g^w, h^w g^{1000})$	1000 times yes

- Answer: use zero-knowledge proofs to prove that each ElGamal encryption contains either g^0 or g^1 without revealing any additional information.

Schnorr's identification protocol

- **Discrete log setting (same as for ElGamal):**
  - **generator *g* of order *q*,** $g^q = 1$
  - **cyclic group:** $1, g, g^2, \ldots, g^{q-1}$
- **Prover's private key: random** $x \in \mathbb{Z}_q$
- **Prover's public key:** $h = g^x$
- **Prover convinces the verifier that it knows private key *x*, corresponding to public key *h*, without leaking information on *x*.**

Schnorr's identification protocol

Prover		Verifier
random $w \in \mathbb{Z}_q$ $a = g^w$	"commitment" a →	
	← challenge c	random $c \in \mathbb{Z}_q$
$r = w + x\,c$	response r →	
		$g^r \stackrel{?}{=} a\,h^c$

Efficient: prover needs to do a cheap exponentiation only

Zero-knowledge proofs

- Protocol between prover P and verifier V
- P must convince V that P knows some secret information without leaking any additional information related to the secret.
- Soundness (in the verifier's interest):
  - only if the prover actually knows the secret, the prover may successfully complete the protocol
- Zero-knowledge (in the prover's interest):
  - no information on the prover's secret should leak at all, even after many executions of the protocol

Soundness property

After sending commitment *a*, only a prover knowing the secret key can answer *any* challenge of the verifier

Lemma: if a prover is able to answer two different challenges $c \neq c'$ with proper responses r, r', for the same initial message *a*, then the prover actually knows *x*

Proof: If $g^r = a\, h^c$ and $g^{r'} = a\, h^{c'}$

then (divide) $g^{r-r'} = a\, h^c / a\, h^{c'} = h^{c-c'}$

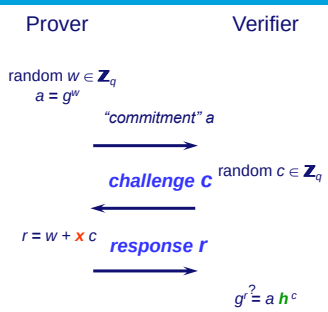
so $h = g^{(r-r')/(c-c')}$

hence (since $h = g^x$) $x = (r-r') / (c-c')$

Zero-knowledge property

- **Captures the idea that no information leaks.**
- **For Schnorr's protocol: consider triples (*a,c,r*) for arbitrary interactions between the prover and the verifier. These triples are from a distribution that can easily be simulated without using the prover's secret key *x*:**
  - **pick values for challenges *c'* and responses *r'* at random, and set $a' = g^{r'} h^{-c'}$.**
  - **simulated triples (*a',c',r'*) follow exactly the same distribution as triples (*a,c,r*) from real executions of the protocol**

Zero-knowledge intuition



Simulator

1. random $c' \in \mathbb{Z}_q$
2. random $r' \in \mathbb{Z}_q$
3. $a' = g^{r'} h^{-c'}$

Game 1:
verifier repeatedly executes protocol and writes down (a,c,r)

Game 2:
verifier repeatedly runs simulator and writes down (*a',c',r'*)

Games 1 and 2 give the same results
(triples follow same probability distribution)

$\Longrightarrow$

Zeroknowledge: executing protocol with prover does not help verifier in learning the secret key *x*

Some other ZK proofs

- Given two generators g_1, g_2
- Given two "public keys" h_1, h_2

- ZK proof that two discrete logs are equal:

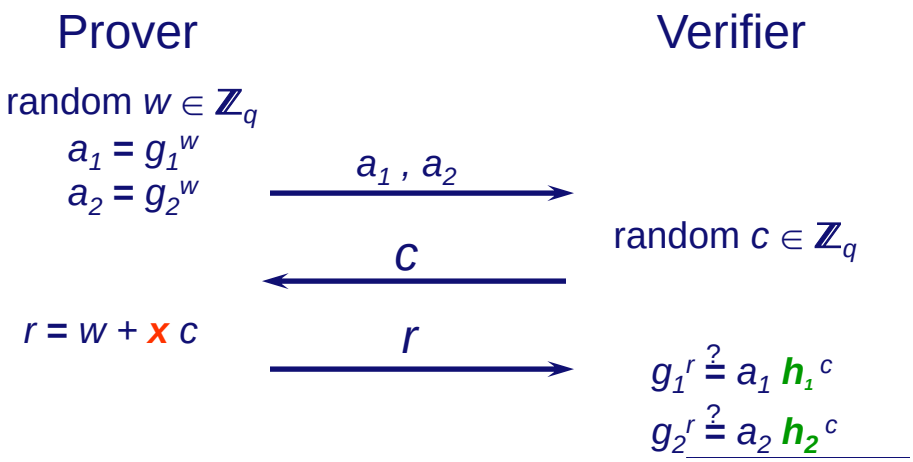
  $\log_{g1} h_1 = \log_{g2} h_2$

  without revealing this value.
- ZK proof for 1-out-of-2 discrete logs, but without revealing which of the two discrete logs one actually knows:

  $\log_{g1} h_1$ *OR* $\log_{g2} h_2$

Equality of two discrete logs

- Proof of knowledge for $\log_{g1} h_1 = \log_{g2} h_2$:
  - prove knowledge of x s.t. $h_1 = g_1^x$, $h_2 = g_2^x$



1-out-of-2 discrete logs

- **Proof of knowledge for $\log_{g1} h_1$ or $\log_{g2} h_2$**
- **Suppose that the prover knows $x_2 = \log_{g2} h_2$, but not $\log_{g1} h_1$**
  - **run an instance of Schnorr's protocol**
    - **given x_2 and $h_2 = g_2^{x_2}$**
  - **simulate an instance of Schnorr's protocol**
    - **given h_1**
  - **allow *one degree of freedom* for the prover by letting the prover split the challenge c into two parts c_1, c_2 such that $c = c_1 + c_2$**

1-out-of-2 proof

Prover (knowing x_2)		Verifier
Pick random r_1, c_1, w_2 $a_1 = g_1^{r1} h_1^{-c1}$ $a_2 = g_2^{w2}$	a_1, a_2 →	
	← c	Pick random challenge c
$c_2 = c - c_1$ $r_2 = w_2 + x_2 c_2$	c_1, r_1, c_2, r_2 →	Verify: $c_1 + c_2 = c$ $g_1^{r1} = a_1 h_1^{c1}$ $g_2^{r2} = a_2 h_2^{c2}$

Blue: **simulated** proof for h_1
Green: **real** proof for h_2

Noninteractive proofs

- Similar to Schnorr signatures.
- Let *m* be the message to be signed.
- Essentially put challenge $c = H(m,a)$ in Schnorr's protocol, where H is a cryptographic hash function.
- Verification of Schnorr signature (c, r) on message m:

$$c \stackrel{?}{=} H(m, g^r h^{-c})$$

Threshold Homomorphic approach

- Each voter posts a homomorphic ElGamal encryption:

$$(a_i, b_i) = (g^{w_i}, h^{w_i} g^{v_i})$$

plus a zero-knowledge proof that $v_i=0$ or $v_i=1$

- Compute $(\Pi_i a_i, \Pi_i b_i) = (g^W, h^W g^T)$ with $W = \Sigma_i w_i$ and $T = \Sigma_i v_i$
- Talliers threshold-decrypt $(g^W, h^W g^T)$ to get g^T and finally T *(Pollard-λ)*

Threshold ElGamal cryptosystem

- Threshold cryptosystem, public key $h = g^s$:
  - talliers $A_1,\ldots, A_n$ share secret s
  - key generation & decryption [Pedersen'91]
- Shared-key generation protocol yields share s_j of secret s for tallier A_j , public part $h_j = g^{s_j}$
- Joint decryption: given $(X,Y) = (g^W, h^W g^T)$:
  - each A_j produces share $z_j = X^{s_j}$
  - A_j proves share's correctness: $\log_g h_j = \log_X z_j$
  - Lagrange interpolation:
    - X^s from sufficiently many correct z_j values

Encrypted Voting without computer (schemes like Pret-a-Voter, PunchScan)

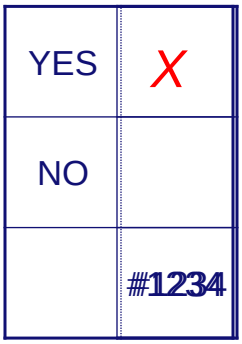
YES	
NO	
	#1234

NO	
YES	
	#5678

Ballot forms:

- perforated in the middle
- options permuted in random order on each ballot form
- numbered for tallying

Encrypted Voting without computer (schemes like Pret-a-Voter, PunchScan)



Voting:
•mark choice;
•remove left-hand side;
•right-hand side is tallied
(and serves as receipt)

Tallying:
seq # links to encrypted vote;
flip if necessary;
tally verifiably
(encrypted votes prepared beforehand by authorities)

Technically same as "masked ballots"

- **Sako/Kilian 1994, also in CFSY96, CGS97**
- **Used in implementations such as CyberVote.**
- **Example:**
  - **consider binary yes/no votes in {0,1}.**
  - **prepare an encrypted bit uniform random *b*, including the validity proof that *b*=0 or *b*=1.**
  - **voter votes *v* by setting 'flip' bit $s = b \oplus v$**
- **By the way, note that CFSY96 already uses commitments rather than encryptions to encapsulate the votes.**

Usability concerns

- Permuted candidates on ballots, etc., etc. ….
  - not even allowed, can cause lots of confusion
  - user-interface should be as simple as possible
- Else we may see these book titles:
  - "Voting for Dummies"
  - "Vote verification for Dummies"
  - "Learn how to vote in 21 days"
  - "Refresh your voting skills – subscription payable every 4 years"
  - "Voting courses–now on HD-DVD and Blu-ray"

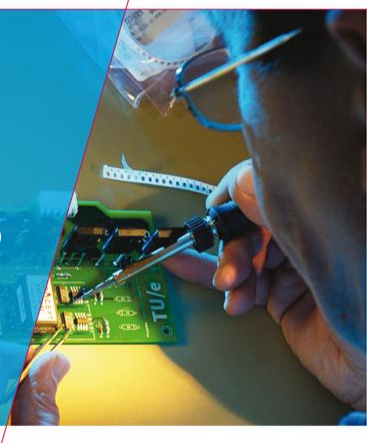
Security issues Pret-a-voter/Punchscan

- Ballot stuffing not excluded:
  - Individual voter verifiability does not guarantee that no votes have been added
    - e.g., votes cast for voters who don't show up
- Randomization attack:
  - Coercer says: mark the top choice
    - Pre: yes 51% no 49%
    - Coerce 4% voters (only yes voters)
    - Post: yes 51-2=49% no 49+2=51%
  - I've noted this problem for Hirt/Sako 2000 paper: also use permuted candidate lists to get receipt-freeness

Practical applications

- A shadow election in May 1998, during the Dutch national elections (with DigiCash, Cap Gemini, D66, NLSign, ...)
- Seattle-based company VoteHere.net used universally verifiable elections schemes
  - many trials, including the first binding Internet election (Alaskan Republican Straw Poll for US president)
- EU project Cybervote
  - TUE provided cryptographic protocol + impl.
  - http://www.eucybervote.org/
  - EADS ran election with 600000 eligible voters in June 2006 (French Ministry of Foreign Affairs).
- Helios, Louvain-la-Neuve, also for IACR election test

Advanced Techniques



TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Zero-knowledge Interval Proofs

- **Discrete log setting:**
  - **$\langle g \rangle = \{1, g, g^2, \ldots, g^{q-1}\}$ for prime q, $g^q = 1$**
  - **$h \in \langle g \rangle$ with $\log_g h$ unknown to anyone**
- **Problem:**
  - **given a commitment $C = g^x h^r$, where $x \in [0,L)$ and r is random.**
  - **prove knowledge of x and r with $x \in [0,L)$.**

Easy cases

- **Given $C = g^x h^r$ with $x \in [0,L)$.**
- **L=1: prove "$C = h^r$" (x=0)**
  - ***Schnorr proof***
- **L=2: prove "$C = h^r$ (x=0) or $C = g\, h^r$ (x=1)"**
  - **OR-composition of 2 Schnorr proofs**
- **$L=2^n$: commit to bits of x … n times L=2 case**
  - **Using 2n Schnorr proofs in total**

Arbitrary L: $2^{n-1} < L \leq 2^n$

- Use intersection $[0, L) = [0, 2^n) \cap [L-2^n, L)$
  - AND-composition of two length-2^n intervals
- Or, use union $[0, L) = [0, 2^{n-1}) \cup [L-2^{n-1}, L)$
  - OR-composition of two length-2^{n-1} intervals

- Either way: about 2*2n = 4n Schnorr proofs
- Optimizations, e.g., if $L = 2^{n-1} + R$, for small R
  - Best case: R=1, then reduced to 2n Schnorr proofs
  - Worst case: still 4n Schnorr proofs
- How to do really better?

General approach: case L = a + b

- For $x \in [0,L)$, we have:

  $x \in [0,a)$ or $x-a \in [0,b)$

- Then we prove recursively:

"C commits in $[0,a)$" OR "C/g^a commits in $[0,b)$"

General approach: case L = a b

- For $x \in [0,L)$, we can write (uniquely):

  $x = y\,b + z$ with $y \in [0,a)$, $z \in [0,b)$

- Split $C = g^x h^r$ into commitment D and E:

  $D = g^y h^s$ with $y = [x/b]$, random s

  $E = g^z h^t$ with $z = x \bmod b$, random t

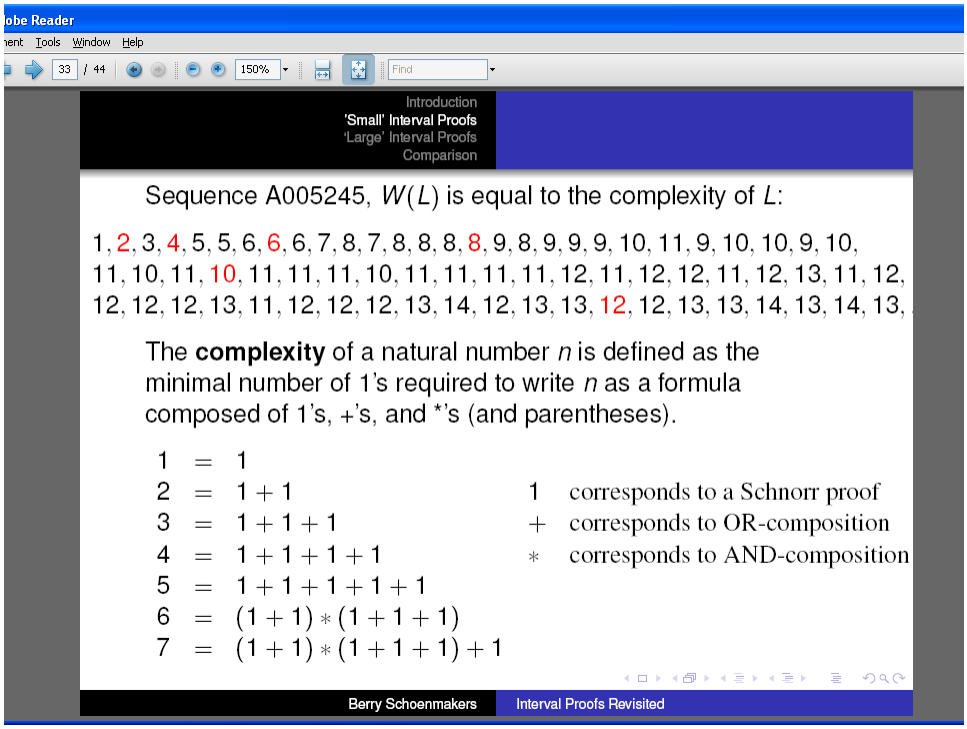
subject to $r = s\,b + t \pmod q$.

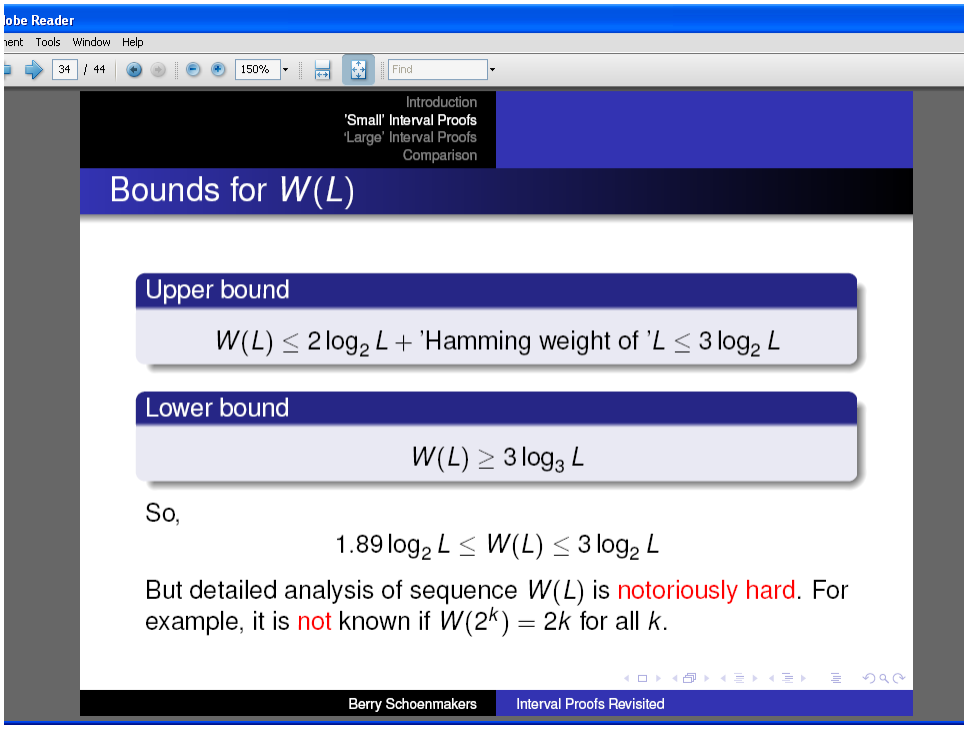
Then $C = D^b E$, and we prove recursively:

"D commits in $[0,a)$" AND "E commits in $[0,b)$"

Leads to interesting combinatorics

- Recall: for $L=2^n$ we need 2n Schnorr proofs
- For general L of bit length n we get close to "optimal" of 2n Schnorr proofs
  - Exact complexity related to so-called integer complexity of L


obe Reader
ent Tools Window Help
33 / 44
150%
Find
Introduction
'Small' Interval Proofs
'Large' Interval Proofs
Comparison
Sequence A005245, $W(L)$ is equal to the complexity of L:
1, 2, 3, 4, 5, 5, 6, 6, 6, 7, 8, 7, 8, 8, 8, 8, 9, 8, 9, 9, 9, 10, 11, 9, 10, 10, 9, 10, 11, 10, 11, 10, 11, 11, 11, 10, 11, 11, 11, 11, 12, 11, 12, 12, 11, 12, 13, 11, 12, 12, 12, 12, 13, 11, 12, 12, 12, 13, 14, 12, 13, 13, 12, 12, 13, 13, 14, 13, 14, 13,
The complexity of a natural number n is defined as the minimal number of 1's required to write n as a formula composed of 1's, +'s, and *'s (and parentheses).
$1 = 1$
$2 = 1+1$
$3 = 1+1+1$
$4 = 1+1+1+1$
$5 = 1+1+1+1+1$
$6 = (1+1)*(1+1+1)$
$7 = (1+1)*(1+1+1)+1$
1 corresponds to a Schnorr proof
+ corresponds to OR-composition
* corresponds to AND-composition
Berry Schoenmakers
Interval Proofs Revisited


obe Reader
ent Tools Window Help
34 / 44
150%
Find
Introduction
'Small' Interval Proofs
'Large' Interval Proofs
Comparison
Bounds for $W(L)$
Upper bound
$W(L) \leq 2\log_2 L +$ 'Hamming weight of' $L \leq 3\log_2 L$
Lower bound
$W(L) \geq 3\log_3 L$
So,
$1.89\log_2 L \leq W(L) \leq 3\log_2 L$
But detailed analysis of sequence $W(L)$ is notoriously hard. For example, it is not known if $W(2^k) = 2k$ for all k.
Berry Schoenmakers
Interval Proofs Revisited

But … FC'2002 paper

- Better and easier approach possible.
  - Helps to split in not-necessarily disjoint intervals.

    L=2m: [0, *L*) = [0, *m*) U [m, *L*)

    L=2m+1: [0, L) = [0, *m+1*) U [m, *L*)
- Given $x \in [0,L)$, let $b \in \{0,1\}$ indicate in which half x lies (breaking ties arbitrarily):
  - Let B denote commitment to b
  - Prove that
    - B commits in {0,1}, and
    - (recursively) C/B^m commits to [0,(L+1)/2)
- Leads to about 2 $\log_2$ L Schnorr proofs

New, optimal solution:

- Assume L=3m for simplicity:

  L=3m: [0, *L*) = [0, *m*) U [m, *2m*) U [2m, *L*)
- Given $x \in [0,L)$, let $t \in \{0,1,2\}$ indicate in which part x lies (breaking ties arbitrarily):
  - Let B denote commitment to t
  - P
    -
    -
- Ca
- Lea

pro

Bet

Optimality:

$f(x) = x \log_x L$ is minimal for $x = e = 2.71828\ldots$

hence take $x = 3$.

Homomorphic vs. mix-based tallying

- Homomorphic: decrypt product of all encrypted votes
  - fast tallying
    - only one threshold-decryption
  - zeroknowledge proof of validity per vote
    - can be costly
- Mix/shuffle based: cascade of verifiable mixes, followed by decryption of ind. votes
  - sequential, relatively slow tallying
    - one threshold-decryption per vote
  - works for any type of vote
    - constant amount of work for voter

What about other solutions?

- Goal:
  - minimize work for voter
  - fast tallying (i.e. homomorphically)
- Motivation:
  - voter's device may be constrained
    - can do a public key encryption
    - cannot generate a sizeable ZK proof of validity
      - either too slow
      - or not enough space/flexibility to load program for computing the ZK proof
      - or both

Client/server trade-offs

- Damgaard & Jurik (PKC'02):
  - client encrypts index (minimal encoding of a vote)
    - plus proof of validity of index (interval proof)
  - servers
    - convert this into homomorphic encrypted vote
    - tally homomorphic votes
- Can be further optimized:
  - only a proof of plaintext knowledge

Secure computation from THCs

- Threshold Homomorphic Cryptosystem (THC):
  - Distributed Key Generation (DKG): to share private key
  - Homomorphic Encryption: under single public key
  - Threshold Decryption: joint decryption protocol
- THCs form basic tool for secure multiparty computation, following [FH93,JJ00,CDN01,DN03,ST04]

“Love Game” for Alice and Bob

Alice’s rule:

for “yes” for “no”

separator:

Bob’s rule:

for “yes” for “no”

Suppose Alice thinks “yes”

Suppose Bob thinks “yes”

Alice and Bob make a random cut and open the deck …

Matching without embarrassments

yes  yes

Match!

yes  no

no  yes

no  no

These 3 cases are indistinguishable.

Hence if you put `no’ you don’t find out the other’s preference.

Boolean AND function

• We have securely computed the AND function $x \wedge y$

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

• Approach can be extended to compute any (computable) function securely.

Basic gates & circuits

- **Addition, subtraction, multiplication by constant:**
  $E(x), E(y), a, b \rightarrow E(ax+by)$ **"for free"**
- **Multiplication, inversion:**
  $E(x), E(y) \rightarrow E(xy), E(1/x)$
- **Random bits, values:**
  'no input' $\rightarrow E(r)$

- **Comparison circuits:**
  $E(x_0),\ldots,E(x_{m-1}), E(y_0),\ldots,E(y_{m-1}) \rightarrow E(x<y), E(\mathrm{sgn}(x-y))$
- **Arithmetic circuits (addition, multiplication, etc.)**
  $E(x_0),\ldots,E(x_{m-1}), E(y_0),\ldots,E(y_{m-1}) \rightarrow E(z_0),\ldots,E(z_{m'-1})$

Binary conversion gate Eurocrypt 2006

- Protocol for computing binary representation (bits) for a given integer value:

$$E(x) \rightarrow E(x_0), \ldots, E(x_{m-1})$$

- Essential to use Paillier THC
  - in general, one cannot use ElGamal THC
    - input: $E(x) = (g^r, h^r g^x)$
    - output: $E(x_0) = (g^r, h^r g^{x0})$, x_0 = least significant bit of x
    - but x_0 is a hard-core bit → would imply computing DLs (for prime order subgroup of Z^*_p and general cyclic groups)

Protocol for LSB gate: E(x) → $E(x_0)$

- Random bit gate $E(r_0)$ $\quad 0 \le r_0 < 2$
- Random value gate $E(r_*)$ $\quad 0 \le r_* < 2^k$
- Threshold-decrypt $E(x + r_0 + 2r_*)$ to get

$$y = x + r_0 + 2r_*$$

- Output $E(x_0) = E(y_0)\, E(r_0)^{1-2y0}$

- Note: $y_0 = x_0 \oplus r_0$ so

$$x_0 = y_0 \oplus r_0 = y_0 + r_0 - 2y_0 r_0$$

Application to voting

- Voter submits encrypted vote $E(x)$
  - x is number of candidate voted for
- Servers convert to $E(x_0),\ldots,E(x_{m-1})$
  - if x is not in range $\{0,1,\ldots,2^m-1\}$, then servers will notice this and vote is verifiably invalid

- Suppose number of voters $\leq N$
- Compute $E(N^{x_i 2^i}) = E(1 + x_i 2^i (N\text{-}1))$ **"for free"**
- Compute $E(N^x)$ using m-1 sec. multiplications
- Tally $E(N^x)$ encryptions homomorphically

Summarizing

- Work for voter is now at "absolute" minimum:
  - homomorphic enc. + proof of plaintext knowledge
  - improves previous "minimum" of Damgaard&Jurik
- Binary conversion is reasonably efficient for, say, x up to 10-bits long.

- Can tallying be done more efficiently ?
- For other types of elections ?

Outlook

Technische Universiteit Eindhoven University of Technology

Where innovation starts

Future: in 25, 50 years ?

- Instant elections (no influence):
  - enitre election in 1 hour; or even in 1 minute.
  - "everybody take your voting device", could be dispensable electronics – plastic/polymer circuits
    - not: the household PC (mom's, dad's, or the kids' PC ?)
    - but: dedicated (clean) device
- Peer-to-peer systems:
  - all voters take part in voting, storing votes and tallying!!
  - national board-room election
- Satellites continuously broadcast contents of the bulletin board
- But also new methods for voting: do a couple of rounds of an instant election… rational cryptography

Rational cryptography

- Parties are not just honest or malicious, but have incentives and a pay-off function.
  - like in game-theory, economic elements …
  - maybe vote buying can even become a feature!
- New forms of decision making (e.g., beyond social choice theory with its voting paradoxes)
- Example:
  - Helger Lipmaa & Edith ElKind, "Interleaving Cryptography and Mechanism Design: The Case of Online Auctions", Fin. Crypto 2004
    - not: given known auction mechanism, implement it cryptographically
    - but: try to use modern cryptography to do a different mechanism, not possible in "real world" with pencil & paper say

Concluding … (1/3)

- Even when e-voting is not used in the near future, we need to set the right example
  - for industry, organizations, where e-voting is simply done
  - for countries where democracy is not so natural: not to give them an excuse of using the same bad voting system
- Paper-based systems not all that good either

International e-voting standards (2/3)

- **Standardization at which levels?**
  - **Level of requirements, properties**
    - **listing ballot secrecy, election integrity, non-coercibility, verifiability can be meaningless (by being too ambiguous)**
  - **Level of solutions (protocol suites, building blocks)**
    - **e.g. for verifiable tallying**
  - **Level of ballot forms (XML structures)**
- **Top-down vs. bottom-up**
  - **top-down: but do we know/understand what we want?**
  - **bottom-up: for a given protocol, perform risk analysis**

Indirect trust (3/3)

"Elegance is not a dispensable luxury but a quality that decides between success and failure." E.W. Dijkstra

- **Transparency:**
  - **not: all voters understand voting system details**
  - **but: voter trust voting system indirectly because**
    - **developed by international group of experts**
    - **in an open, well-documented process**
    - **described in internationally available standards**
    - **on which open, detailed specifications are based**
    - **choice of many, independent implementations by a whole range of vendors**
- **In principle, voters, talliers, scrutinizers can implement their own software**

Author's address

Berry Schoenmakers

Coding and Crypto group
Dept. of Mathematics and Computer Science
Technical University of Eindhoven
P.O. Box 513
5600 MB Eindhoven
The Netherlands

berry@win.tue.nl
l.a.m.schoenmakers@tue.nl

http://www.win.tue.nl/~berry/

Threshold cryptography

- **Open specification**
- **But**
  - **multiple implementations**
    - **programmed by different companies, people**
    - **operated by different organizations of various nature (political parties, consumer organizations, government, activists, etc.)**
    - **at different geographic locations (countries)**
    - **run of different platforms (Windows, Linux, etc.)**
- **E.g. an 50 out of 100 scheme (or even different weights for certain parties)**
  - **if 50 parties conspire they can break ballot secrecy**
  - **or they can break `robustness'**

Multiparty security

- **Wrong way:**
  - **involve multiple parties who each protect certain assets,**
  - **or even, multiple parties protecting the same asset**
    - **Risk increases: more parties are involved each of which can do harm**
- **Right way:**
  - **split protection of assets among multiple parties s.t. no one can cheat on their own; only when sufficienctly many parties cooperate**
    - **Threshold cryptography (uses secret sharing)**