De Cifris Trends
in
Cryptographic Protocols

University of Trento and *De Componendis Cifris*
October 2023

Lecture 10

Advanced Cryptography in E-Voting

Riccardo Longo

Fondazione Bruno Kessler
Center for Cybersecurity

Electronic Voting

- Exploit **digital** technology to enhance the voting process
- Increase **accessibility** and encourage **participation**
- Improve **tallying**: speed, accuracy, verifiability

Security of E-Voting

Desired properties of an election:

- **Vote Privacy** No one should get to know how someone has voted
- **Fairness** No one should learn (partial) results before the end of the voting period
- **Eligibility** Only eligible voters should be able to cast a valid vote, and no more than one *per capita*
- **Verifiability** It should be possible to check that the whole process has run correctly
- **Coercion Resistance** Voters should be able to vote freely

Cryptography to the Rescue

- In the next slides we will see how we can use **cryptography** to achieve some of the previous properties
- We will start from the naïve approach and try to refine the solutions
- We will mention and briefly discuss a lot of primitives and protocols, so we will stay on a **high level**

DISCLAIMER

- Electronic voting, and in particular *Remote* or *Internet* Voting is a **hard problem**
- We will **not** see a complete protocol or a whole solution
- The focus will be on how we can use cryptography to tackle some problems and which are the limitations

Privacy: Encryption

Idea

- To **hide** vote content: encrypt it
- **Asymmetric Encryption**:
  - All voters use the same public key
  - Only the authority that has the private key can decrypt and compute the results

Problems

- The **authority can see** everyone's vote's content
- Verification would require to publish the plaintexts, undermining privacy

Privacy and Fairness: Threshold Cryptography

Idea

- **Distribute trust** among multiple parties
- No single entity can decrypt votes to subvert privacy or fairness
- **Distributed Key Generation** and **Threshold Decryption**:
  - No single party has control of the private key
  - t out of n authorities have to collaborate to decrypt

Problems

- There's still a clash between privacy and verification

Privacy and Fairness: Homomorphic Encryption

Idea

- **Aggregate** votes before decryption
- **Partially Homomorphic Encryption**:
  - Tallying performed as a sum
  - An additively-homomorphic encryption scheme allows to perform the tally on the ciphertexts
  - Only the final results are decrypted

Problems

- How can we be sure that each vote counts as one?

Example:
An Additively-Homomorphic Encryption Scheme

Definition (Exponential ElGamal)

- $\mathbb{G}$ cyclic group of prime order p, g_0, g_1, g_2 generators
- The **private key** is $x_1, x_2 \in \mathbb{Z}_p$, the **public key** is $h = g_1^{x_1} \cdot g_2^{x_2}$
- A message $m \in \mathbb{Z}_p$ is **encrypted** as
  $\mathcal{E}_h^r[m] = (\alpha, \beta, \gamma) = (g_1^r, g_2^r, h^r \cdot g_0^m)$, with $r \in \mathbb{Z}_p$
- If m is small we can **decrypt** by computing
  $g_0^m = \gamma/(\alpha^{x_1} \cdot \beta^{x_2})$ and brute-forcing the DLOG
- The **homomorphic addition** of ciphertexts is the point-wise multiplication over $\mathbb{G}^3$: $\mathcal{E}_h^r[m] \cdot \mathcal{E}_h^{r'}[m'] = \mathcal{E}_h^{r+r'}[m + m']$

Example:
A Ballot Encoding for Homomorphic Tallying

- Suppose there are **n candidates** $c_1, \dots c_n$, and that each voter can express up to **p preferences**
- A vote is encoded as $v = (v_1, \dots, v_n) \in \{0,1\}^n$ where $v_i = 1$ if and only if the voter wants to vote for c_i
- With $\sum v = (\sum v_1, \dots, \sum v_n)$ you obtain the number of preferences obtained by each candidate
- A valid vote v satisfies:
  - $v_i = 0 \bigvee v_i = 1 \quad \forall i = 1, \dots, n$;
  - $0 \leq \sum_{i=1}^{i=n} v_i \leq p$;
- Encrypted ballot: $(\mathcal{E}_h[v_1], \dots, \mathcal{E}_h[v_n])$
- Note that the total number of preferences is a relatively small number, so **we can easily decrypt** it

Verifiability: Non-Interactive Zero-Knowledge Proofs

Idea

- We want to **prove** that the protocol has been followed **without revealing** any secret data to preserve privacy
- **NIZKPs**:
  - A ZKP allows to prove a statement about secret data without revealing it
  - Apply the *Fiat-Shamir* transformation to a *Sigma-Protocol* with *special soundness* to make it non-interactive
  - The NIZKP can be attached to every computation step to prove that it is correct and anyone can check it

Problems

- Not every cryptographic protocol is ZKP-friendly

Example: ZKPs for ElGamal

- The base ZKP is **Schnorr**'s proof of knowledge of a DLOG: given $g, h \in \mathbb{G}$, p.k.o. $\rho \in \mathbb{Z}_p$ such that $h = g^\rho$
- A variant (by **Okamoto**) allows to prove the knowledge of a plaintext given an ElGamal ciphertext
- From it we can derive a proof of plaintext equality: given $h \in \mathbb{G}$, $m \in \mathbb{Z}_p$, $(\alpha, \beta, \gamma) \in \mathbb{G}^3$ p.k.o. $r \in \mathbb{Z}_p$ such that $(\alpha, \beta, \gamma) = \mathcal{E}_h^r[m]$
- The **Cramer-Damgård-Schoenmakers** technique allows to prove the disjunction of statements:
  - The challenge fixes the sum of the sub-proofs challenges
  - This gives free choice on all but one of the challenges
  - We can cheat in all but one of the sub-proofs, i.e. at least one statement is true
- We can prove that a ciphertext encrypts either 0 or 1

Privacy and Verifiability: Verifiable Shuffled Re-Encryption

Idea

- Like in a physical ballot-box, **shuffle** the ballots before tallying
- **Verifiable Shuffled Re-Encryption**:
  - Re-encrypt the encrypted ballots in a different order
    - The re-encryption of an additively homomorphic ciphertext is just a homomorphic addition with an encryption of 0
  - A NIZKP proves that the new list of ciphertexts is just a re-encryption of the original list modulo a permutation
  - If multiple authorities make a mix each, then no-one can track the ballots without colluding

Problems

- This sub-protocol is quite expensive
- Shuffling can interfere with some Eligibility checks (e.g. preventing double voting)

Eligibility: Digital Signatures

Idea

- Prove that the ballot has been created by an **eligible** voter
- **Digital Signature**:
  - Ballots are signed before casting
  - Public keys of eligible voters published on a Bulletin Board

Problems

- Anyone can see who votes and track their ballot: clash with Coercion Resistance (forced abstention)

Eligibility and Coercion Resistance: Linkable Group Signatures

Idea

- **Don't reveal who** signed, but only that it was someone eligible
- Link signatures created by the same voter, so that double-voting can be avoided
- **Linkable Group Signatures**:
  - Reveals only that the public key associated to the private key used to sign belongs to a group
  - The actual key stays hidden, the voter's identity is not revealed
  - Signatures created with the same private key reveal a common element, so they can linked together

Problems

- Classic LGS schemes have size proportional to the number of public keys in the group:
  - Impractical to use all eligible public keys
  - Choice of subset can hinder Coercion Resistance
- The coercer can still ask for another signature to track someone's votes

Coercion Resistance: Designated-Verifier NIZKP

Idea

- Let voters **fool coercers** by pretending to comply
- We need some sort of spoof info, indistinguishable from the real one, that can be easily forged
- For Verifiability voters should be convinced that the received info is genuine, while leaving the coercer in doubt
- **Designated-Verifier Zero-Knowledge Proofs**:
  - ZKP that "info is real" $\vee$ "prover knows the private key"
  - The voter controls the key-pair, thus knows the private key
  - The authority that proves that the info is real does not know the private key so the voter is convinced
  - A coercer cannot be convinced because anything given by the voter can be forged (since they know the private key)

Problems

- How can we make sure that the voter knows the private key?
- What is this forgeable info and how is it used?

Verifiability: Benaloh Challenges

Idea

- How can voters **trust the encrypting device** not to change the vote contents?
- **Benaloh Challenge**:
  - When challenged, the encrypting device reveals the randomness used and another device is used to check the correctness
  - To preserve privacy the challenged ballot is spoiled and not cast
  - To preserve privacy spoiled ballots should contain a random preference
  - Randomly chose whether to spoil or cast, so the device is forced to behave honestly to avoid being caught

Problems

- Poor **usability** (difficult to understand and to perform correctly)
- The actual cast ballots are not audited

Final Remarks

- Coercion Resistance is **hard** and requires some assumptions (e.g. untappable channels or safe environments)
- Verifiability often **clashes** with Privacy and Coercion Resistance
- **Usability** has also to be taken in consideration
- Besides the cryptographic protocol, the **implementation** brings a lot of security implications
- And then there is the public's **trust** in the system...

De Componendis Cifris

https://www.decifris.it