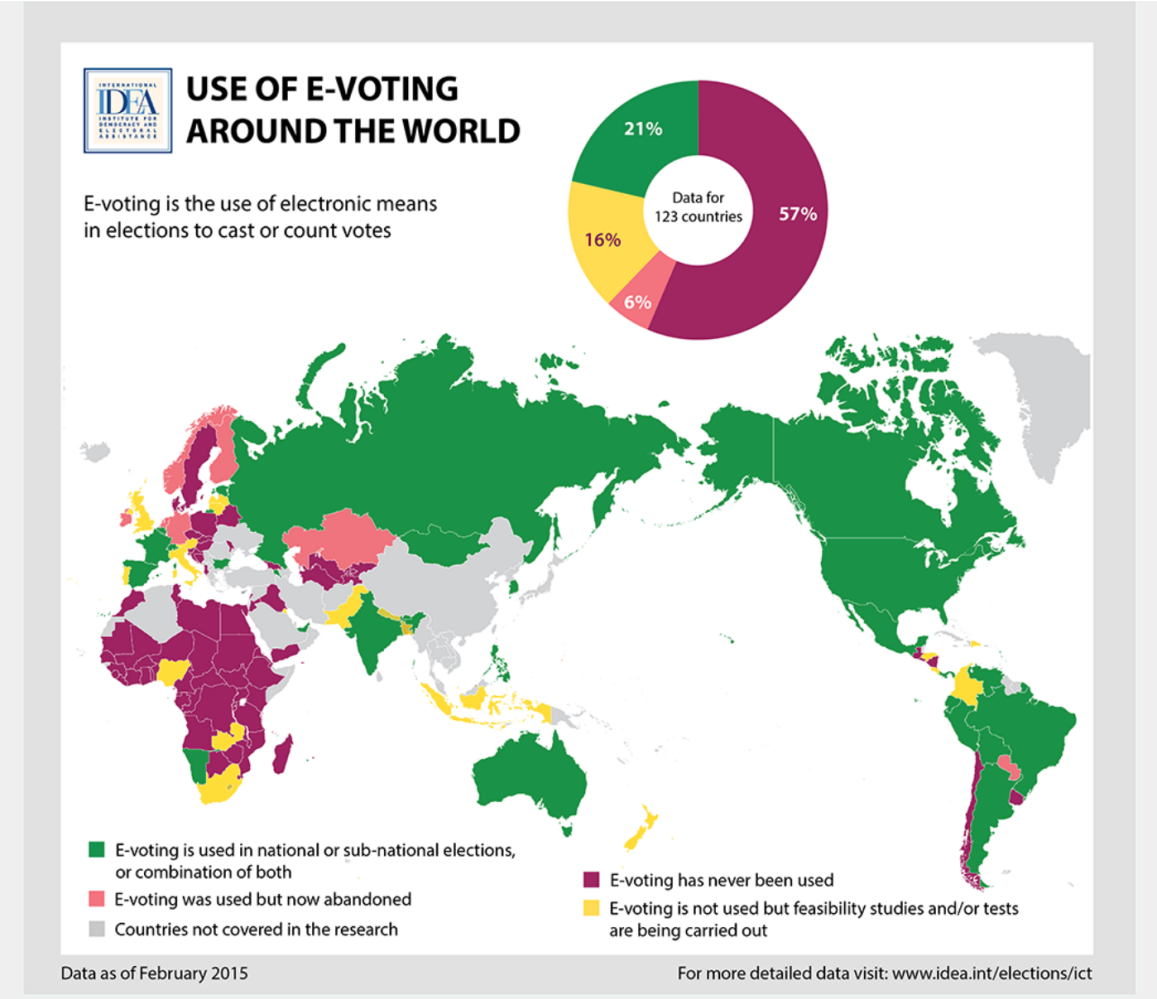
Authentication with weaker trust assumptions for voting systems

Elizabeth A. Quaglia and Ben Smyth

Africacrypt 2018

May 9, 2018

E-voting around the world



What is a voting system?

A **voting system** is a decision-making procedure used by *voters* to choose a representative from some *candidates*, with the support of an *administrator*, often known as *tallier*.
It typically consists of (at least) the following four phases.

- **Setup**: An administrator initialises the system
- **Vote**: Each voter constructs and casts a ballot for their choice
- **Tally**: The administrator tallies the recorded ballots and announces an outcome
- **Verify**: Voters and other interested parties check that the outcome corresponds to votes expressed in recorded ballots.

What is a voting system?

A **voting system** is a decision-making procedure used by *voters* to choose a representative from some *candidates*, with the support of an *administrator*, often known as *tallier*.
It typically consists of (at least) the following four phases.

- **Setup**: An administrator initialises the system
- **Vote**: Each voter constructs and casts a ballot for their choice
- **Tally**: The administrator tallies the recorded ballots and announces an outcome
- **Verify**: Voters and other interested parties check that the outcome corresponds to votes expressed in recorded ballots.

The outcome is a distribution of choices used to select a representative.

In *first-past-the-post* elections, for example, it corresponds to the choice with highest frequency.

Security requirements of a voting system

In a voting system, we expect that

Security requirements of a voting system

In a voting system, we expect that

1. Choices are made freely

Security requirements of a voting system

In a voting system, we expect that

1. Choices are made freely
2. Voters are convinced the announced outcome is correct

Security requirements of a voting system

In a voting system, we expect that

1. Choices are made freely — Ballot secrecy
2. Voters are convinced the announced outcome is correct

Security requirements of a voting system

In a voting system, we expect that

1. Choices are made freely
2. Voters are convinced the announced outcome is correct — Verifiability

Security requirements of a voting system

In a voting system, we expect that

1. Choices are made freely
2. Voters are convinced the announced outcome is correct

We will provide an intuition for how to model these requirements later in the talk.

Example of a voting system

Helios [Adi08]: an open-source, web-based voting system that has been used by the IACR, the Catholic University of Louvain, and Princeton University, for instance. Helios works (roughly) as follows.

- **Setup**: An administrator generates a key pair, publishes the public key and a proof of correct key construction.
- **Vote**: Each voter encrypts their choice with the public key, proves correct ciphertext construction, and casts the ciphertext coupled with the proof, as their ballot.
- **Tally**: The administrator collects any cast ballots, discards any ballot for which proofs do not hold, combines the ciphertexts in the remaining ballots to derive the encrypted outcome, decrypts, proves correctness of decryption, and announces the outcome and proof.
- **Verify**: Anyone can recompute the combination, check the proofs, and output 1 if these checks succeed and 0 otherwise.

Helios 2.0

Helios was first implemented as Helios 2.0.

It is intended to satisfy
- Ballot secrecy (due to encryption)
- Verifiability (thanks to the proofs)

Voting systems with external vs internal authentication

Voting systems need to ensure that choices are made by eligible voters.

Voting systems with external vs internal authentication

Voting systems need to ensure that choices are made by eligible voters.

- Some voting systems, like Helios, rely on external authentication services (via Facebook, Google and Yahoo using OAuth).
- Others, like Civitas/JCJ [JCJ05], use cryptography to implement their own (internal) authentication mechanisms.

Voting systems with external vs internal authentication

Voting systems need to ensure that choices are made by eligible voters.

- Some voting systems, like Helios, rely on external authentication services (via Facebook, Google and Yahoo using OAuth).
- Others, like Civitas/JCJ [JCJ05], use cryptography to implement their own (internal) authentication mechanisms.

Systems that build upon cryptography are typically preferable to systems trusting external service providers.

Voting systems with external vs internal authentication

Voting systems need to ensure that choices are made by eligible voters.

- Some voting systems, like Helios, rely on external authentication services (via Facebook, Google and Yahoo using OAuth).
- Others, like Civitas/JCJ [JCJ05], use cryptography to implement their own (internal) authentication mechanisms.

Systems that build upon cryptography are typically preferable to systems trusting external service providers.

In this work, we will construct voting systems with *internal* authentication from systems with *external* authentication, weakening the trust assumptions for voting systems.

Our contribution

In this paper we

- extend the formal models for voting systems with internal authentication
- provide a construction from voting systems with external authentication to systems with internal authentication
- prove our construction satisfies ballot secrecy and verifiability
- apply our results to Helios

Modelling voting systems with external and internal authentication [SFC15]

- **Setup**$(\kappa) \rightarrow (pk, sk, mb, mc)$ is run by the tallier.
- **Register**$(pk, \kappa) \rightarrow (pd, d)$ is run by the registrar. //*credentials*
- **Vote**$(d, pk, nc, v, \kappa) \rightarrow b$ is run by voters.
- **Tally**$(sk, nc, \mathbf{bb}, L, \kappa) \rightarrow (\mathbf{v}, pf)$ is run by the tallier. //*electoral roll*
- **Verify**$(pk, nc, \mathbf{bb}, L, \mathbf{v}, pf, \kappa) \rightarrow s$, where s is either 0 or 1, is run to audit an election.

Modelling voting systems with external and internal authentication [SFC15]

- **Setup**$(\kappa) \to (pk, sk, mb, mc)$ is run by the tallier.
- **Register**$(pk, \kappa) \to (pd, d)$ is run by the registrar. //*credentials*
- **Vote**$(d, pk, nc, v, \kappa) \to b$ is run by voters.
- **Tally**$(sk, nc, \mathbf{bb}, L, \kappa) \to (\mathbf{v}, pf)$ is run by the tallier. //*electoral roll*
- **Verify**$(pk, nc, \mathbf{bb}, L, \mathbf{v}, pf, \kappa) \to s$, where s is either 0 or 1, is run to audit an election.

The system must satisfy *correctness.*

Our construction

The intuition

- We make use of a digital signature scheme and a non-interactive proof system
- Each voter publishes a ballot constructed using the underlying system (with external authentication), along with a signature on that ballot and a proof they constructed both.

Our construction

The intuition

- We make use of a digital signature scheme and a non-interactive proof system
- Each voter publishes a ballot constructed using the underlying system (with external authentication), along with a signature on that ballot and a proof they constructed both.

The ingredients

- A voting system with external authentication $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}_\Gamma, \mathsf{Tally}_\Gamma, \mathsf{Verify}_\Gamma)$
- A digital signature scheme $\Omega = (\mathsf{Gen}_\Omega, \mathsf{Sign}_\Omega, \mathsf{Verify}_\Omega)$
- A non-interactive proof system, derived from a sigma protocol Σ and a hash function $\mathcal{H}$ using the Fiat-Shamir transform, $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$

Our construction - the details

We define voting system with internal authentication Ext2Int$(\Gamma, \Omega, \Sigma, \mathcal{H})$ as

- Setup computes Setup_Γ.
- Register computes Gen_Ω to obtain the credential pair (pd, d).
- Vote computes
  - ballot b by running Vote_Γ
  - signature σ by running Sign_Ω on b
  - proof τ by running Prove_Σ on a relation that ensures ballot and signature are correct

  and outputs ballot (pd, b, σ, τ).
- Tally computes Tally_Γ on set auth$(\mathbf{bb}, L)$
- Verify computes Verify_Γ on set auth$(\mathbf{bb}, L)$

where auth$(\mathbf{bb}, L)$ ensures the tallied ballots are authorised and discards ballots submitted under the same credential.

Security results

Our construction results in a voting system with internal authentication satisfying

Security results

Our construction results in a voting system with internal authentication satisfying

- Ballot secrecy

Security results

Our construction results in a voting system with internal authentication satisfying

- Ballot secrecy
- Verifiability
  - Individual verifiability
  - Universal verifiability
  - Eligibility verifiability

Ballot secrecy

We extend the definition of ballot secrecy of [SFC15] to systems with internal authentication.

The intuition

We challenge the adversary to determine whether the voting oracle produces ballots for *left* or *right* input, by giving the adversary the oracle's output, the election outcome and the tallying proof.
We prevent the adversary from winning trivially by requiring that the ballots on the tallied bulletin board output from the oracle are *balanced*.

Ballot secrecy

We extend the definition of ballot secrecy of [SFC15] to systems with internal authentication.

The intuition

We challenge the adversary to determine whether the voting oracle produces ballots for *left* or *right* input, by giving the adversary the oracle's output, the election outcome and the tallying proof.
We prevent the adversary from winning trivially by requiring that the ballots on the tallied bulletin board output from the oracle are *balanced*.

Theorem 1

If

- Γ is ballot secret
- Ω is strongly unforgeable
- the proof system is zero knowledge

then our construction satisfies **ballot secrecy**.

Individual verifiability

The intuition

Individual verifiability challenges the adversary to generate a collision from algorithm Vote for voters who have not been corrupted.

Individual verifiability

The intuition

Individual verifiability challenges the adversary to generate a collision from algorithm Vote for voters who have not been corrupted.

Theorem 2

If Ω satisfies strong unforgeability, then our construction satisfies **individual verifiability**.

Universal verifiability

The intuition

Universal verifiability challenges the adversary to concoct a scenario in which either:

- Verify accepts, but the election outcome is not correct, or
- Tally produces an election outcome that Verify rejects.

Hence, universal verifiability requires algorithm Verify to accept if and only if the election outcome is correct.

Universal verifiability

The intuition

Universal verifiability challenges the adversary to concoct a scenario in which either:

- Verify accepts, but the election outcome is not correct, or
- Tally produces an election outcome that Verify rejects.

Hence, universal verifiability requires algorithm Verify to accept if and only if the election outcome is correct.

Theorem 3

If

- Γ satisfies universal verifiability
- Ω is strongly unforgeable
- the proof system is zero knowledge

then our construction satisfies **universal verifiability**.

Eligibility verifiability

The intuition

Eligibility verifiability challenges an adversary, who can corrupt voters, to generate a valid ballot under the private credential of a voter who has not been corrupted.

Eligibility verifiability

The intuition

Eligibility verifiability challenges an adversary, who can corrupt voters, to generate a valid ballot under the private credential of a voter who has not been corrupted.

Theorem 4

If

- Ω is strongly unforgeable
- the proof system is zero knowledge

then our construction satisfies **eligibility verifiability**.

Application to Helios

We consider the most recent formalisation of Helios, *Helios'16*, adapted to defend against attacks on Helios identified over the years [SFC15].

Using our construction we can derive a voting system with internal authentication from *Helios'16*.

Application to Helios

We consider the most recent formalisation of Helios, *Helios'16*, adapted to defend against attacks on Helios identified over the years [SFC15].

Using our construction we can derive a voting system with internal authentication from *Helios'16*.

Existing security results on *Helios'16* [SFC15, S18] together with our security results can directly be applied to ensure the resulting system satisfies ballot secrecy and verifiability.

Application to Helios

We consider the most recent formalisation of Helios, *Helios'16*, adapted to defend against attacks on Helios identified over the years [SFC15].

> Using our construction we can derive a voting system with internal authentication from *Helios'16*.

Existing security results on *Helios'16* [SFC15, S18] together with our security results can directly be applied to ensure the resulting system satisfies ballot secrecy and verifiability.
Our results can also be applied to the variant of Helios that uses mixnets [Adi08, BGP11].

Comparison to HeliosC [CGGI14]

Voting systems derived from Helios using our construction are similar to *Helios-C* [CGGI14].

Comparison to HeliosC [CGGI14]

Voting systems derived from Helios using our construction are similar to *Helios-C* [CGGI14].
Indeed, they use ballots that include a Helios ballot and a signature on that Helios ballot.

Comparison to HeliosC [CGGI14]

Voting systems derived from Helios using our construction are similar to *Helios-C* [CGGI14].
Indeed, they use ballots that include a Helios ballot and a signature on that Helios ballot.

Crucially, the systems derived by our construction also include proofs of correct construction, unlike Helios-C.

Comparison to HeliosC [CGGI14]

Voting systems derived from Helios using our construction are similar to *Helios-C* [CGGI14].
Indeed, they use ballots that include a Helios ballot and a signature on that Helios ballot.

Crucially, the systems derived by our construction also include proofs of correct construction, unlike Helios-C.

This inclusion is key to ensure ballot secrecy.

Conclusion

In this talk, we

- formalised voting systems with internal authentication
- showed how to derive one from a system with external authentication
- provided intuition for ballot secrecy and verifiability, fundamental security requirements in e-voting
- presented an application to Helios, a widely used e-voting system

Thank you for your attention! ☺

References

- [Adi08] Adida. Helios: Web-based Open-Audit Voting. *USENIX 2008*.
- [BGP11] Bulens, Giry & Pereira. Running Mixnet-based elections with Helios, *EVT/WOTE 2011*.
- [CGGI14] Cortier, Galindo, Glondu & Izabachene. Election verifiability for Helios under weaker trust assumptions, *ESORICS 2014*.
- [JCJ05] Juels, Catalano & Jakobsson. Coercion-resistant electronic elections, *WPES 2005*.
- [S18] Smyth. Ballot secrecy: Security definition, sufficient conditions, and an analysis of Helios, *ePrint 2018*.
- [SFC15] Smyth, Frink & Clarkson. Election Verifiability: Cryptographic Definitions and an Analysis of Helios, Helios-C and JCJ, *ePrint 2015*.