Annotated biography

Free & fair elections

Ben Smyth

May 5, 2025

Foundational ideas are explored in tutorials:

[Smy18a] Ben Smyth. A foundation for secret, verifiable elections. Technical Report 2018/225, Cryptology ePrint Archive, 2018

[QS18c] Elizabeth A. Quaglia and Ben Smyth. A short introduction to secrecy and verifiability for elections. Technical Report 1702.03168, arXiv, 2018

Theoretical foundations appear in technical publications:

Secrecy for freedom: Indistinguishability game

You're the cool kid. Bravado grants freedom of expression. Elsewhere there's fear. The game is the game. You ain't the cool kid in every room; some opulent power observing, social constraints presiding. Freedom to express one's conscience, morals, will, without fear, a century-old innovation: Eighteen-hundreds closed with realisation, we can't vote freely in public. Secret ballots are better. Instead of declaring preferences in public, mark ballots privately. Tally votes. Reveal collective decisions. Keep individual votes private.

Vendors claim their systems ensure secrecy. Hackers devise breaks. Designers argue, "that's not what I meant by security." Formalisation enables confirmation of vendor claims. "[Without] precise definitions, security claims would be a moving target for analysts," explain Koblitz & Menezes. I formalise ballot secrecy as indistinguishability between any election and any other with the same outcome.

[Smy21] Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. *Journal of Computer Security*, 29(6):551–611, 2021

I define an indistinguishability game: A malicious adversary interacts with a benign challenger operating a voting system. The adversary supplies two votes, and the challenger constructs a ballot for one of them, a coin flip deciding which. They repeat the process, with that coin flip determining subsequent vote selection. The adversary casts challenger- and adversary-constructed ballots,

and the challenger tallies those ballots. The adversary wins by determining the result of the coin flip, demonstrating a security-breaking execution, wherein casting some vote combination leaks information.

I also provide a blueprint for construction of voting systems from asymmetric encryption schemes, and prove that the blueprint produces voting systems delivering on secrecy needs, when the underlying asymmetric encryption scheme is perfectly correct, non-malleable, and ill-formed ciphertexts are distinguishable from well-formed ciphertexts.

Cryptosystems are proven secure by reduction; security of one thing 'reduces to' security of another: Analysts prove cryptosystems as secure as underlying parts. There's no winning adversary unless some part is insecure. My proof shows blueprint-constructed voting systems are secure unless the underlying encryption scheme is broken. I derive one such secure-by-design voting system using a non-malleable derivative of ElGamal built by me and Toshiba colleagues:

[SH19] Ben Smyth and Yoshikazu Hanatani. Non-malleable encryption with proofs of plaintext knowledge and applications to voting. *International Journal of Security and Networks*, 14(4):191–204, 2019

[SHM15] Ben Smyth, Yoshikazu Hanatani, and Hirofumi Muratani. NM-CPA secure encryption with proofs of plaintext knowledge. In *IWSEC'15: 10th International Workshop on Security*, volume 9241 of *LNCS*. Springer, 2015

I claimed non-malleability wasn't strictly necessary [Smy21], I was wrong:

[Smy24a] Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. Technical Report 2015/942, Cryptology ePrint Archive, 2024

Should secrecy be preserved when ballots are suppressed? *Bien sûr*! It follows that non-malleable ballots are necessary: Tallying a meaningfully related ballot suffices to violate secrecy. In the extreme, suppose an adversary observes your ballot, suppresses all ballots including yours, and casts only a ballot related to yours. You're led to believe your ballot wasn't successfully cast. Yet the adversary can establish your vote from tallying. The ballot related to yours tallies to a vote related to yours, the relation reveals your vote to the adversary.

Degrees of secrecy. Ballot secrecy is achievable with varying degrees of information leakage: At one extreme, winner-only secrecy reveals just the winning candidate (e.g., [KSRH12]). At the other, secrecy by anonymity reveals anonymised votes. In between, tally-then-decrypt secrecy delivers on traditional privacy expectations, wherein an election reveals nothing more than a frequency distribution of voters' votes.

[Smy24b] Ben Smyth. Championing tally-then-decrypt secrecy, 2024

The gulf between anonymised votes and vote frequency distributions is witnessed by mixnets revealing the contents of every individual ballot (i.e., anonymised

votes), whilst computing an encrypted tally and then decrypting reveals only a frequency distribution. The subtlety between anonymise-then-decrypt and tally-then-decrypt voting systems has gone unnoticed. Voting systems adopted by nation states fall short. (E.g., Swiss Post provide mixnet-based voting for Switzerland and acknowledge need for improvement.) I champion tally-then-decrypt voting systems.

Stronger privacy notions. Ballot secrecy requires voters protect their own privacy: Voters must follow the prescribed voting procedure. Freedom may be sacrificed, votes may be sold. With Ashley and Liz, I surveyed stronger formulations of privacy that demand voters be unable to convince adversaries of their votes, even when they reveal secrets generated during the voting procedure.

[FQS19] Ashley Fraser, Elizabeth A. Quaglia, and Ben Smyth. A critique of game-based definitions of receipt-freeness for voting. In *ProvSec'19: 13th International Conference on Provable and Practical Security*, volume 11821 of *LNCS*, pages 189–205. Springer, 2019

Coercion resistance goes further: Regardless of what a voter does, they cannot reveal how they voted, or whether they abstained, even when following adversarial instruction. Seminal work by Juels et al. proposed a coercion resistance voting system with quadratic complexity. I achieve linear complexity:

[Smy19] Ben Smyth. Athena: A verifiable, coercion-resistant voting system with linear complexity. Technical Report 2019/761, Cryptology ePrint Archive, 2019

I proved verifiability, wanted to prove coercion resistance, surveyed definitions:

[HS20] Thomas Haines and Ben Smyth. Surveying definitions of coercion resistance. Technical Report 2019/822, Cryptology ePrint Archive, 2020

We found all definitions unsuitable. One is unsatisfiable, two label systems coercion resistance when they aren't, another is incompatible with Athena. After nearly three decades of study, coercion-resistance remains poorly understood; establishing better intuition and suitably formalising remain open problems.

Verifiability for legitimacy: Reachability game

Public forums create legitimacy. Everyone can count raised hands. Secrecy creates a quandary: Decisions must reflect stakeholder will. In public, stakeholders vote openly. Legitimacy is irrefutable. Anyone present can determine the outcome. But in public, we aren't free. Conversely, private decisions protect freedom, but lack legitimacy, there's no visible proof.

Everybody lies. Don't trust. Verify. Decisions must demonstrably reflect stakeholder will. There must be an absolute path from decision to stakeholder votes. No one can be fooled into accepting the wrong result. Correct results are always accepted. These two principles aren't captured by existing definitions:

[SC22] Ben Smyth and Michael R. Clarkson. Surveying definitions of election verifiability. *Information Processing Letters*, 177, 2022

[Smy20] Ben Smyth. Surveying global verifiability. *Information Processing Letters*, 163, 2020

Voting systems vulnerable to attack can be proven to satisfy current definitions. With Michael and Steven, I present a new one:

[SFC21] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election Verifiability: Cryptographic Definitions and an Analysis of Helios, Helios-C, and JCJ. Technical Report 2015/233, Cryptology ePrint Archive, 2021

We consider a key holder computing a tally along with evidence of correct tallying. Those outputs are verified to determine whether the tally corresponds to votes expressed in cast ballots. We formulate reachability games capturing the aforementioned principles with respect to tallying and verification procedures. Generalising results for a plurality of key holders leads to a surprising conclusion: Philosophically, we want fairness, not verifiability.

A fair voting system must ensure tallies correspond to votes expressed in cast ballots (soundness) and such tallies must always be computable (completeness). Whilst a single key holder can tally and announce results themselves, a plurality of key holders can't. The idea that a tally is announced and verified doesn't transpose. For a plurality of key holders, tallying comprises three phases: An encrypted tally is computed, each key-holder partially decrypts, and the (decrypted) tally is announced. Formalising fairness in terms of a verification procedure accepting a tally is natural with a single key holder, whereas such a procedure is irrelevant when a plurality of key holders compute partial decryptions. Instead, we want the tallying algorithm guaranteeing soundness in the third-phase, and that algorithm always computing a tally when public key shares are system generated (completeness). I formalise these properties:

publish fair elections

[Smy25a] Ben Smyth. Fair elections, 2025. To appear

My formalisation requires: Any system computed outcome from adversary computed inputs is the sum of votes expressed in authorised ballots, excluding ballots authorised by the same private credential, which are all discarded except for the last such ballot (soundness). For system computed key shares, adversary computed public voter credentials and ballots, an outcome is system computable (completeness).

We mere mortals, even the most studious, can't compute digital ballots. Voters are at the mercy of machines, which mightn't even compute ballots, let alone correctly compute ballots expressing voters' votes. Whilst publishing cast ballots allows each vote to check the presence of their ballot (individual verifiability), expression of their vote is only guaranteed when voters compute their own ballots, which is atypical for digital systems. Further consideration is necessary when machines are untrusted:

[RRS21] Peter B. Rønne, Peter Y. A. Ryan, and Ben Smyth. Cast-as-intended: A formal definition and case studies. In *FC'21: 25th International Conference on Financial Cryptography and Data Security*, volume 12676 of *LNCS*, pages 251–262. Springer, 2021

Cast-as-intended demands a voter be able to check whether a machine-generated ballot expresses their vote.

Digital ballot construction typically mandates sampling bits. If a machine abandons the prescribed sampling procedure, computation delivers something resembling a ballot, rather than a correctly computed ballot: A ballot constructed in disregard for the prescribed procedure is not correctly computed. It may resemble a ballot. A ballot may even be computable in that way. However, ignoring the construction mandate means the result cannot a priori be considered a correctly computed ballot. Herein lies the rub: Voters cannot determine whether machines compute ballots or things resembling ballots. Cast-as-intended and individual verifiability don't compose to enable determination of whether collected ballots express voters' votes, since voters cannot determine whether machines even compute ballots correctly, there's a gap:

Publish mind-the-gap revision

[Smy25c] Ben Smyth. Mind the Gap: Individual- and universal-verifiability plus cast-as-intended don't yield verifiable voting systems. Technical Report 2020/1054, Cryptology ePrint Archive, 2025

Assumptions underpinning cast-as-intended and individual verifiability are mismatched: Cast-as-intended assures a ballot expresses a vote, not whether the ballot is correctly computed, whilst individual verifiability assures correctly computed ballots will be tallied. Cast-as-intended and individual verifiability don't compose in the expected way. A further security notion is necessary for fairness when ballots are computed on untrusted voting kiosks.

Individual verifiability is irrelevant; linkability and unforgeability are what we need: Cast-as-intended assures a ballot expresses a voter's vote, unforgeability guarantees authorised ballot construction by only private credential holders, and linkability allows a voter to verify whether their private credential was misused (a merciless machine will give-up private voter credentials, they're caught when ballots constructed with the same private credential are linkable), finally, soundness ensures a tally corresponds to votes expressed in cast ballots.

Case study: Helios voting system

Véronique tasked me to prove Helios 2.0 secure; a seeming laborious chore 'to prove the obvious' spurned a decade of research. I found a vulnerability: Replaying a voter's ballot amplifies their contribution, observing this amplification in an election result reveals the voter's vote. Exploiting ballot malleability to cast some variation of a voter's ballot has a similar effect.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society, 2011

[SC11] Ben Smyth and Véronique Cortier. A note on replay attacks that violate privacy in electronic voting schemes. Technical Report RR-7643, INRIA, 2011

[Smy12] Ben Smyth. Replay attacks that violate ballot secrecy in Helios. Technical Report 2012/185, Cryptology ePrint Archive, 2012

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013

Ballots remain malleable in Helios 3.1.4, which is similarly vulnerable [Smy21]. Omitting related ballots from tallying has been postulated as a defense. With Michael and Steven, I found this approach violates soundness: An adversary casts some variant of a voter's ballot, such that the variant is processed before the voter's ballot, causing the voter's ballot to be omitted from tallying, violating soundness [SFC21]. Our first attempted fix switched IND-CPA secure ElGamal for a IND-CCA2 secure derivative:

[BCP+11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *ESORICS'11: 16th European Symposium on Research in Computer Security*, volume 6879 of *LNCS*, pages 335–354. Springer, 2011

IND-CCA2 is superfluous. With Michael and Steven, I propose Helios'16 using an NM-CPA secure variant of ElGamal, which we prove sound and complete [SFC21], I prove ballot secrecy too [Smy21].

Helios uses OAuth to authenticate voters' ballots, soundness is proven with respect to third-party authenticated ballots (rather than voter credentials) and Helios is supposed to ensure only the last vote of each voter has influence (strong non-reusability), only it doesn't:

[MS19] Maxime Meyer and Ben Smyth. Exploiting re-voting in the Helios election system. *Information Processing Letters*, 143:14–19, 2019

Strong non-reusability enables voters to change their votes, which provides flexibility, and aids education (voters can "ask the help of anyone for submitting a random ballot, and then re-vot[e] privately afterwards"). An adversary can intercept an early ballot's authorisation token, wait until the voter casts their last ballot, and then release the intercepted token, causing the early ballot to be tallied, rather than the last. Victims may detect malice, but there's no evidence of malpractice, victims have little recourse.

With Liz, I combine digital signatures and non-interactive proofs to eliminate trust assumptions on operators of authentication services such as OAuth:

[QS18a] Elizabeth A. Quaglia and Ben Smyth. Authentication with weaker trust assumptions for voting systems. In *AFRICACRYPT'18: 10th International Conference on Cryptology in Africa*, volume 10831 of *LNCS*, pages 322–343. Springer, 2018

Helios-C made progress in this direction by signing Helios ballots, but we found a ballot secrecy vulnerability due to non-malleability, and realised that proving correct construction of both the Helios ballot and the signature suffices as a fix.

Mixnet variants. Rather than compute an encrypted tally, a variant of Helios applies a mixnet to encrypted votes and decrypts mixed encrypted votes to reveal results, implementations achieve neither soundness nor completeness:

[Smy18b] Ben Smyth. Verifiability of Helios Mixnet. In *Voting'18: 3rd Workshop on Advances in Secure Electronic Voting*, volume 10958 of *LNCS*, pages 247–261. Springer, 2018

Zeus is a fork of Helios 3.1.4 spliced with mixnet code, and helios-server-mixnet extends Zeus, both inherit vulnerabilities from Helios which I exploit.

Score for influence: Honeybee sidestep

Say you prefer Starbucks rather than Prêt. McDonalds more than Buger King. Republicans over Democrats. Someone else fancies the opposite. Another independents. You all like coffee-burgers-America. Yet, two tea-drinking fish-and-chip eating monster-raving looney supporting Brits control the vote.

Two votes for tea, one for each of Starbucks, Prêt, and an independent, tea-drinkers win. Two votes for fish and chips, one for each burger option, fish-and-chips win. As does monster raving looney.

"Equality—impossible," cry social choice theorists. First-past-the-post plagued by vote splitting. Borda dismissing, championing ranked voting. "Bound to lead to error," muses Condorcet. Arrow discovering far more sinister defects.

[Smy22] Ben Smyth. Overcoming Arrow's impossibility: Honeybee sidestep, 2022

I review seminal results. Walk through Arrow's impossibility theorem; teach ranked voting as fallible. And conclude with Smith sidestepping Arrow, guided by dancing honeybees, proclaiming score voting "a larger improvement in 'democracy' than the entire invention of democracy."

Scoring is better. Instead of choosing just one option, rate each. Score Starbucks-McDonalds-Republicans five, Prêt-BK-Democrats four, independents three. Someone else switches the five and four. Another puts independents on top. Coffee-burgers-America win.

Hooking up with Fiat-Shamir

A zero-knowledge proof convinces us of some claim's validity without revealing specifics. For example, a key holder can prove possession of their private key without leaking any information about their key. Zero-knowledge proofs are typically derived by application of the Fiat-Shamir transformation to sigma

protocols. Unfortunately, the transformation is commonly misapplied, introducing security vulnerabilities. I present a JavaScript framework for correct transformation:

[Smy25b] Ben Smyth. Hooking up with Fiat-Shamir, 2025

Sigma protocols are interactive; a prover sends a statement and a commitment to a verifier, the verifier replies with a challenge, and the prover supplies a response. The Fiat-Shamir transformation replaces the verifier's challenge with a hash over the statement and the commitment, reducing exchange of three messages to a single zero-knowledge proof.

I implement the sigma protocol by Chaum et al. for demonstrating knowledge of a discrete logarithm, used by key holders to demonstrate possession of a private corresponding to their public key. I also implement sigma protocols by Chaum & Pedersen for demonstrating knowledge of equality between discrete logarithms and by Schoenmakers for disjunctive equality between logarithms, used to prove correctness of partial decryptions and, respectively, to prove correct ciphertext construction, the latter giving way to non-malleable ElGamal encryption over booleans, and generalising to non-malleable ElGamal over boolean arrays [SHM15, SH19], as used by Helios'16 [SFC21].

Auctions

I was repeatedly asked to review auction papers, observed a striking similarity with voting, had a lovely evening with Adam (who I taught during his undergrad), hired him to demonstrate proof of concept: From free & fair voting systems, we can construct free & fair auction systems.

[MSQ14] Adam McCarthy, Ben Smyth, and Elizabeth A. Quaglia. Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes. In *FC'14: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *LNCS*, pages 51–63. Springer, 2014

I proved general results with Liz:

[QS18b] Elizabeth A. Quaglia and Ben Smyth. Secret, verifiable auctions from elections. *Theoretical Computer Science*, 730:44–92, 2018

Seemingly disjoint research fields are actually related. Unification facilitates progress. Advances in one field can be capitalised to advance the other.

Autonomous reasoning, symbolic formalisations

My PhD years focused on the exploitation of mathematical logic for attack discovery and cryptosystem verification. I explored the applied pi calculus for pen-and-paper analysis, and ProVerif for automation, writing manuals for each.

[Smy11] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011

[RS11] Mark D. Ryan and Ben Smyth. Applied pi calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, chapter 6. IOS Press, 2011

[BS11] Bruno Blanchet and Ben Smyth. *ProVerif 1.85: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2011

I also extended ProVerif's reasoning capabilities: Cryptosystems execute over unbounded spaces. Analysis is undecidable. State-of-the-art automated reasoning techniques focus on sound methodologies that are incomplete; false attacks may be reported, tools mightn't terminate. Bruno defined compilation from ProVerif's dialect of the applied pi calculus to Horn clauses along with automated resolution over clauses. Compilation over-approximates the attacker's power and ProVerif may report false attacks.

Reasoning over biprocesses to prove equivalence has limitations. For example, consider parallel systems that each output two constants, one being designed to output constant A in parallel with constant B, the other being designed in reverse. These two systems are notated as processes $\mathtt{out}(A) \mid \mathtt{out}(B)$ and $\mathtt{out}(B) \mid \mathtt{out}(A)$, respectively. Obviously they're equivalent. But ProVerif cannot prove it due to sequential (rather than parallel) processing of biprocesses: ProVerif takes biprocess $\mathtt{out}([A, B]) \mid \mathtt{out}([B, A])$ as input, attempts to prove equivalence on the left-hand side of the parallel operator, then the right, and concludes equivalence is satisfied when both left and right sides satisfy equivalence (which isn't true here). Introducing barrier synchronisation to ProVerif allows such equivalences to be proved using a swapping technique established with Mark and Stéphanie. With Petr, we implemented proof of concept. And with ProVerif's creator Bruno, developed the theory, proved soundness, and implemented the new reasoning technique in ProVerif:

[DRS08] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008

[KSR10] Petr Klus, Ben Smyth, and Mark D Ryan. Proswapper: Improved equivalence verifier for proverif, 2010

[BS16] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *CSF'16: 29th Computer Security Foundations Symposium*, pages 310–324. IEEE Computer Society, 2016

[BS18] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. *Journal of Computer Security*, 26(3):367–422, 2018

These results enable automated verification of a larger class of equivalence properties, including analysis of ballot secrecy.

False attacks are also typical during analysis of protocols that require temporal secrets, because temporality is lost when compiling to Horn clauses. For example, suppose a participant commits to a nonce before seeing their interlocutor's nonce. It's impossible for the latter nonce to aid construction of the former. But Horn clauses over-approximate reality and ProVerif reports false attacks. These false attacks can be avoided by ensuring compilation preserves temporal order. With Tom, I achieved this objective by injection of phases, allowing more precise compilation:

[CSS15] Tom Chothia, Ben Smyth, and Chris Staite. Automatically Checking Commitment Protocols in ProVerif without False Attacks. In *POST'15: 4th Conference on Principles of Security and Trust*, volume 9036 of *LNCS*. Springer, 2015

These results further extend ProVerif's capabilities.

References

[BCP+11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *ESORICS'11: 16th European Symposium on Research in Computer Security*, volume 6879 of *LNCS*, pages 335–354. Springer, 2011.

[BS11] Bruno Blanchet and Ben Smyth. *ProVerif 1.85: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2011.

[BS16] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *CSF'16: 29th Computer Security Foundations Symposium*, pages 310–324. IEEE Computer Society, 2016.

[BS18] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. *Journal of Computer Security*, 26(3):367–422, 2018.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society, 2011.

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.

[CSS15] Tom Chothia, Ben Smyth, and Chris Staite. Automatically Checking Commitment Protocols in ProVerif without False Attacks. In *POST'15: 4th Conference on Principles of Security and Trust*, volume 9036 of *LNCS*. Springer, 2015.

[DRS08] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008.

[FQS19] Ashley Fraser, Elizabeth A. Quaglia, and Ben Smyth. A critique of game-based definitions of receipt-freeness for voting. In *ProvSec'19: 13th International Conference on Provable and Practical Security*, volume 11821 of *LNCS*, pages 189–205. Springer, 2019.

[HS20] Thomas Haines and Ben Smyth. Surveying definitions of coercion resistance. Technical Report 2019/822, Cryptology ePrint Archive, 2020.

[KSR10] Petr Klus, Ben Smyth, and Mark D Ryan. Proswapper: Improved equivalence verifier for proverif, 2010.

[KSRH12] Dalia Khader, Ben Smyth, Peter Y. A. Ryan, and Feng Hao. A Fair and Robust Voting System by Broadcast. In *EVOTE'12: 5th International Conference on Electronic Voting*, volume 205 of *Lecture Notes in Informatics*, pages 285–299. Gesellschaft für Informatik, 2012.

[MS19] Maxime Meyer and Ben Smyth. Exploiting re-voting in the Helios election system. *Information Processing Letters*, 143:14–19, 2019.

[MSQ14] Adam McCarthy, Ben Smyth, and Elizabeth A. Quaglia. Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes. In *FC'14: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *LNCS*, pages 51–63. Springer, 2014.

[QS18a] Elizabeth A. Quaglia and Ben Smyth. Authentication with weaker trust assumptions for voting systems. In *AFRICACRYPT'18: 10th International Conference on Cryptology in Africa*, volume 10831 of *LNCS*, pages 322–343. Springer, 2018.

[QS18b] Elizabeth A. Quaglia and Ben Smyth. Secret, verifiable auctions from elections. *Theoretical Computer Science*, 730:44–92, 2018.

[QS18c] Elizabeth A. Quaglia and Ben Smyth. A short introduction to secrecy and verifiability for elections. Technical Report 1702.03168, arXiv, 2018.

[RRS21] Peter B. Rønne, Peter Y. A. Ryan, and Ben Smyth. Cast-as-intended: A formal definition and case studies. In *FC'21: 25th International Conference on Financial Cryptography and Data Security*, volume 12676 of *LNCS*, pages 251–262. Springer, 2021.

[RS11] Mark D. Ryan and Ben Smyth. Applied pi calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, chapter 6. IOS Press, 2011.

[SC11] Ben Smyth and Véronique Cortier. A note on replay attacks that violate privacy in electronic voting schemes. Technical Report RR-7643, INRIA, 2011.

[SC22] Ben Smyth and Michael R. Clarkson. Surveying definitions of election verifiability. *Information Processing Letters*, 177, 2022.

[SFC21] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election Verifiability: Cryptographic Definitions and an Analysis of Helios, Helios-C, and JCJ. Technical Report 2015/233, Cryptology ePrint Archive, 2021.

[SH19] Ben Smyth and Yoshikazu Hanatani. Non-malleable encryption with proofs of plaintext knowledge and applications to voting. *International Journal of Security and Networks*, 14(4):191–204, 2019.

[SHM15] Ben Smyth, Yoshikazu Hanatani, and Hirofumi Muratani. NM-CPA secure encryption with proofs of plaintext knowledge. In *IWSEC'15: 10th International Workshop on Security*, volume 9241 of *LNCS*. Springer, 2015.

[Smy11] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011.

[Smy12] Ben Smyth. Replay attacks that violate ballot secrecy in Helios. Technical Report 2012/185, Cryptology ePrint Archive, 2012.

[Smy18a] Ben Smyth. A foundation for secret, verifiable elections. Technical Report 2018/225, Cryptology ePrint Archive, 2018.

[Smy18b] Ben Smyth. Verifiability of Helios Mixnet. In *Voting'18: 3rd Workshop on Advances in Secure Electronic Voting*, volume 10958 of *LNCS*, pages 247–261. Springer, 2018.

[Smy19] Ben Smyth. Athena: A verifiable, coercion-resistant voting system with linear complexity. Technical Report 2019/761, Cryptology ePrint Archive, 2019.

[Smy20] Ben Smyth. Surveying global verifiability. *Information Processing Letters*, 163, 2020.

[Smy21] Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. *Journal of Computer Security*, 29(6):551–611, 2021.

[Smy22] Ben Smyth. Overcoming Arrow's impossibility: Honeybee sidestep, 2022.

[Smy24a] Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. Technical Report 2015/942, Cryptology ePrint Archive, 2024.

[Smy24b] Ben Smyth. Championing tally-then-decrypt secrecy, 2024.

[Smy25a] Ben Smyth. Fair elections, 2025. To appear.

[Smy25b] Ben Smyth. Hooking up with Fiat-Shamir, 2025.

[Smy25c] Ben Smyth. Mind the Gap: Individual- and universal-verifiability plus cast-as-intended don't yield verifiable voting systems. Technical Report 2020/1054, Cryptology ePrint Archive, 2025.