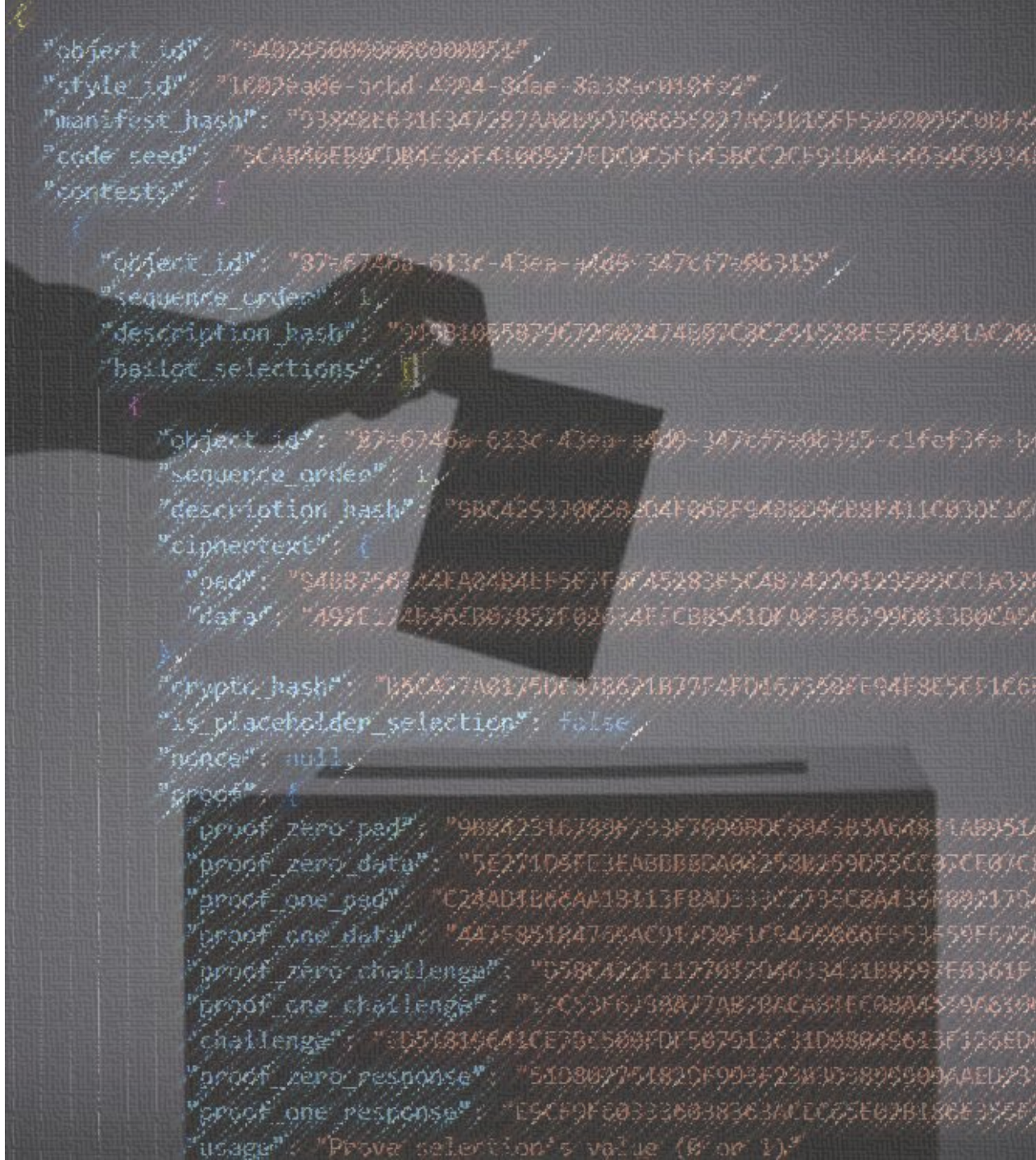
ElectionGuard

A modular approach to verifiable elections

Michael Naehrig

Microsoft Research

E-Vote-ID, October 2, 2025



ElectionGuard

What is it?

- a free, open-source software toolkit,
- **not** a full election system!

Election vendors can incorporate it into their existing election systems.

What it does:

- produces evidence: election record,
- enables end-to-end verifiable elections.

ElectionGuard

What is it?

- a free, open-source software toolkit,
- **not** a full election system!

Election vendors can incorporate it into their existing election systems.

What it does:

- produces evidence: election record,
- enables end-to-end verifiable elections.

Microsoft's role

- Employs researchers.
- Initiated project in 2018, publicly announced in 2019.
- Employed project manager.
- Sponsored 3rd party developers.
- Helped transition to the Election Technology Initiative in 2023.

Microsoft

ElectionGuard

What is it?

- a free, open-source software toolkit,
- **not** a full election system!

Election vendors can incorporate it into their existing election systems.

What it does:

- produces evidence: election record,
- enables end-to-end verifiable elections.

Microsoft's current role

- Employs researchers.

Microsoft

Deployments

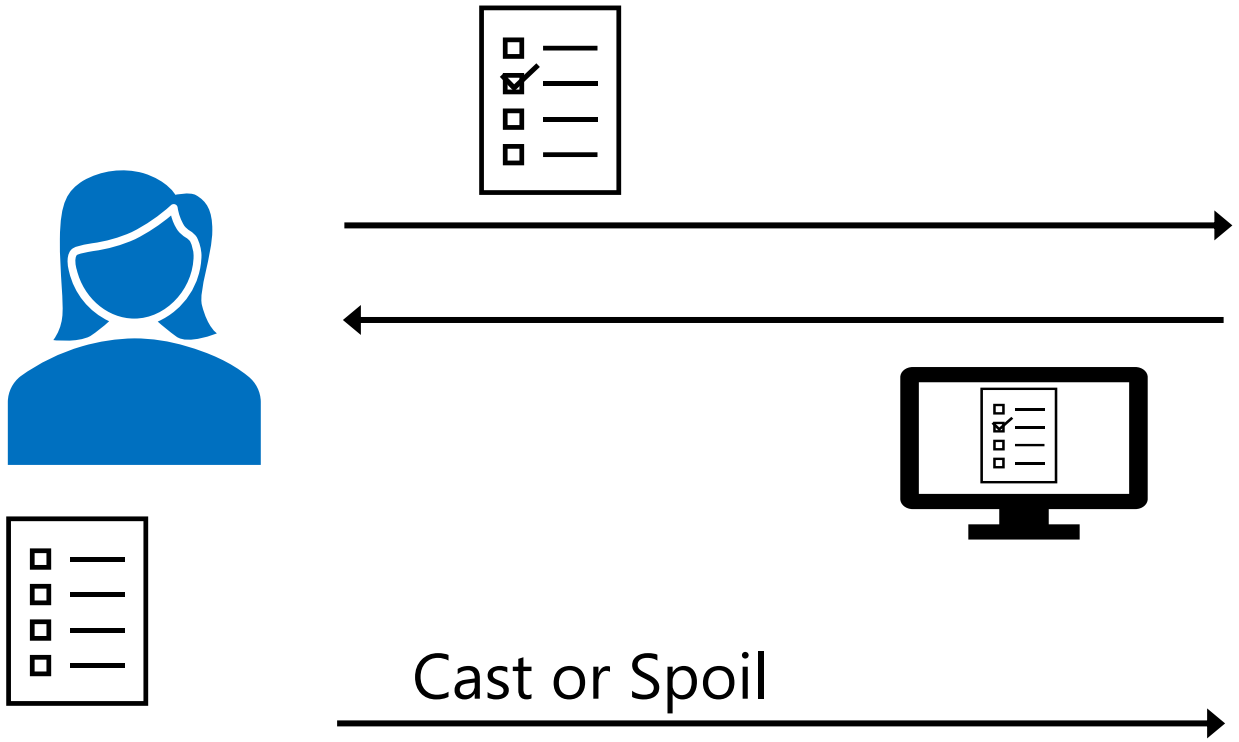
In-person Public Elections

- 2020 – Fulton, WI
  with VotingWorks
- 2020 – Inyo County, CA (audit)
  with VotingWorks
- 2022 – Preston, ID
  with Hart Intercivic
- 2023 – College Park, MD
  with Hart Intercivic

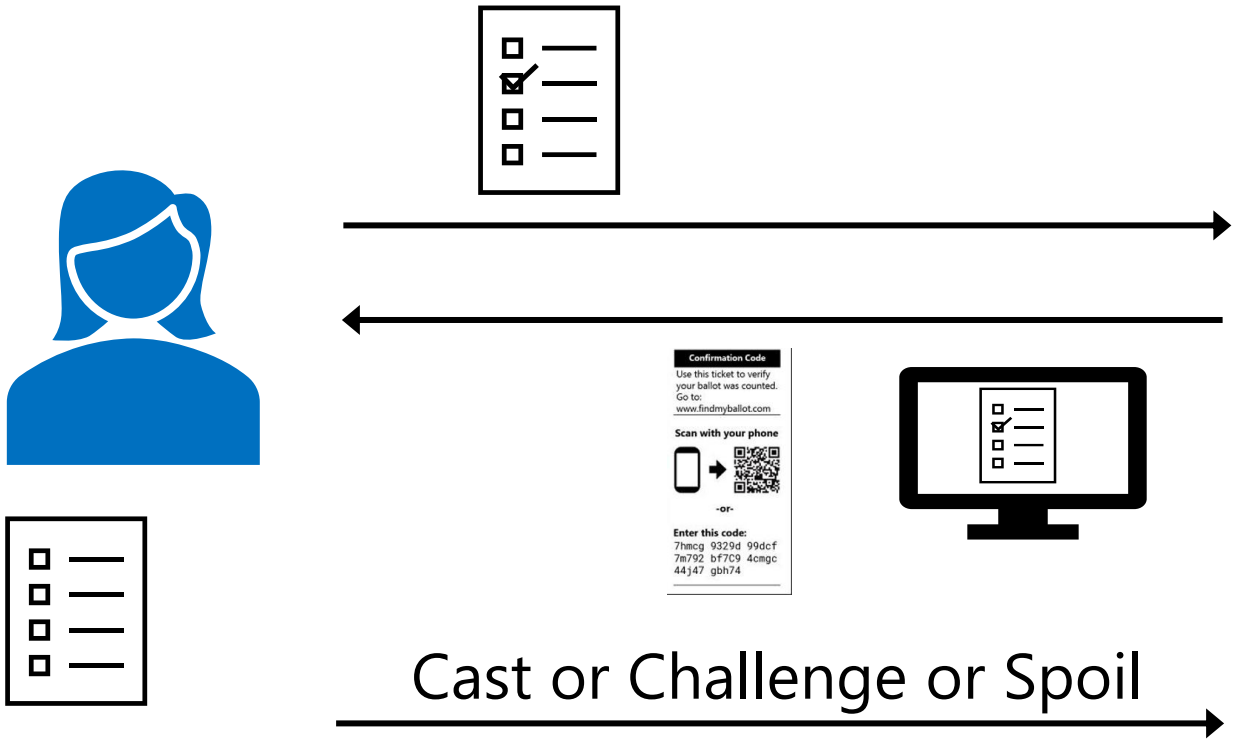
Other Elections (Remote)

- 2020 – U.S. House Democratic Caucus
  with Markup
- 2023 – Neuilly-sur-Seine, FRANCE
  with Electis
- 2023 – Utah absentee voters
  with Enhanced Voting
- 2024 – internal election
  with Concordium

Voter experience – an example



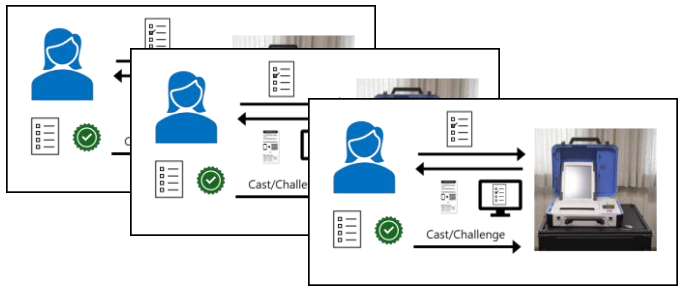
Voter experience – an example



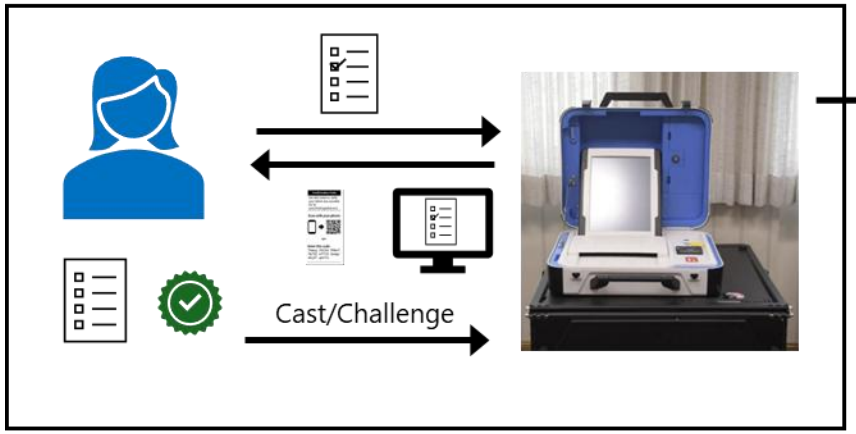
Protocol sketch

Roles

- Voters
- Election administrator
- Trustees (guardians)
- Verifiers



Ballots, proofs

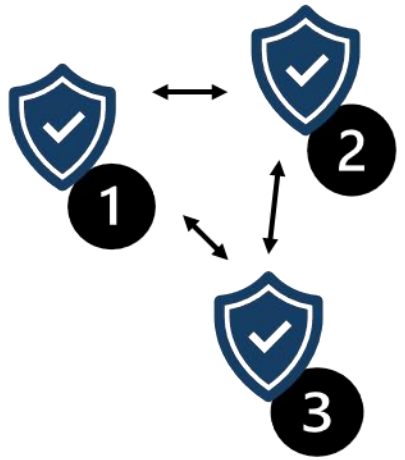


Ballots, proofs

Public key, proofs

Encrypted tallies

Decrypted tallies, proofs



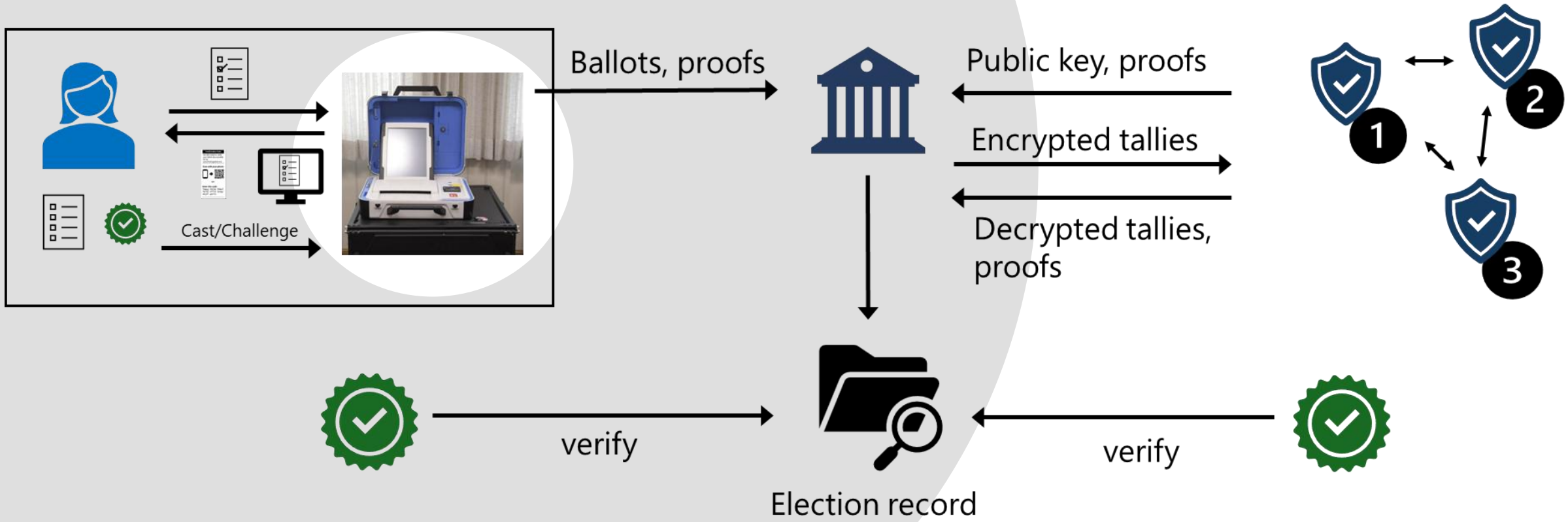
verify

verify

Election record

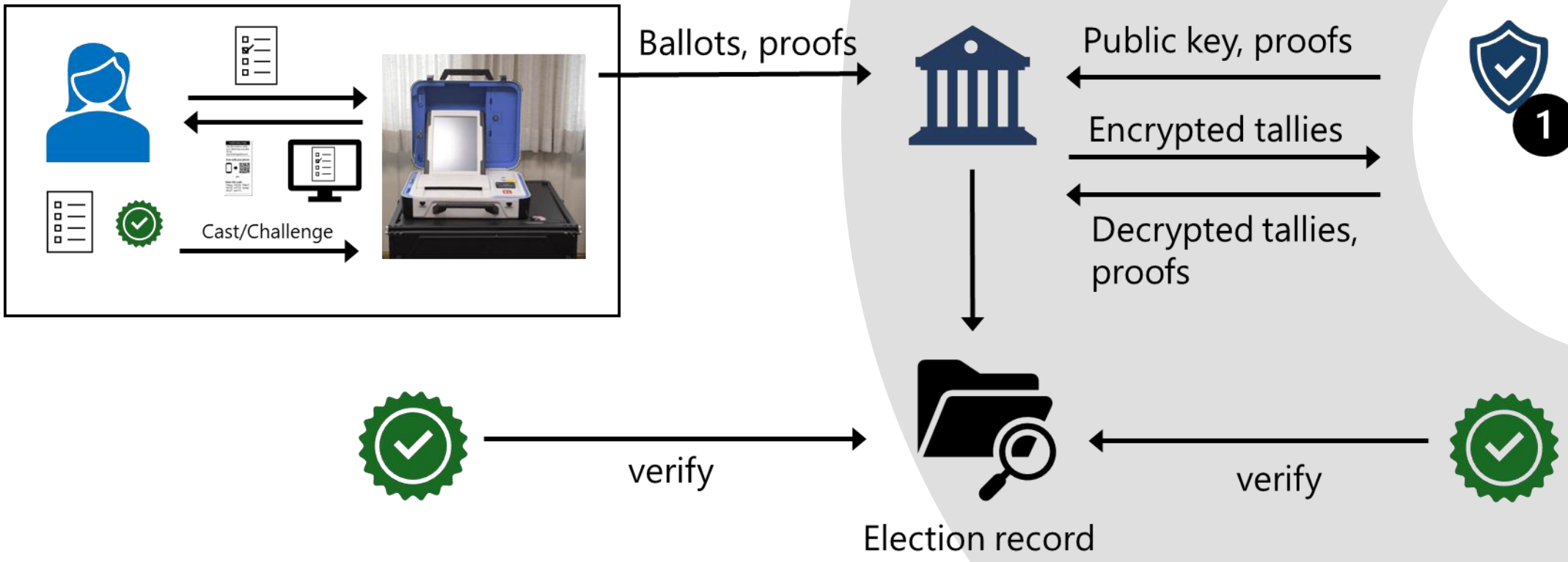
Cryptographic components

- Ballot encryption
- Proofs of ballot well-formedness
- Confirmation code



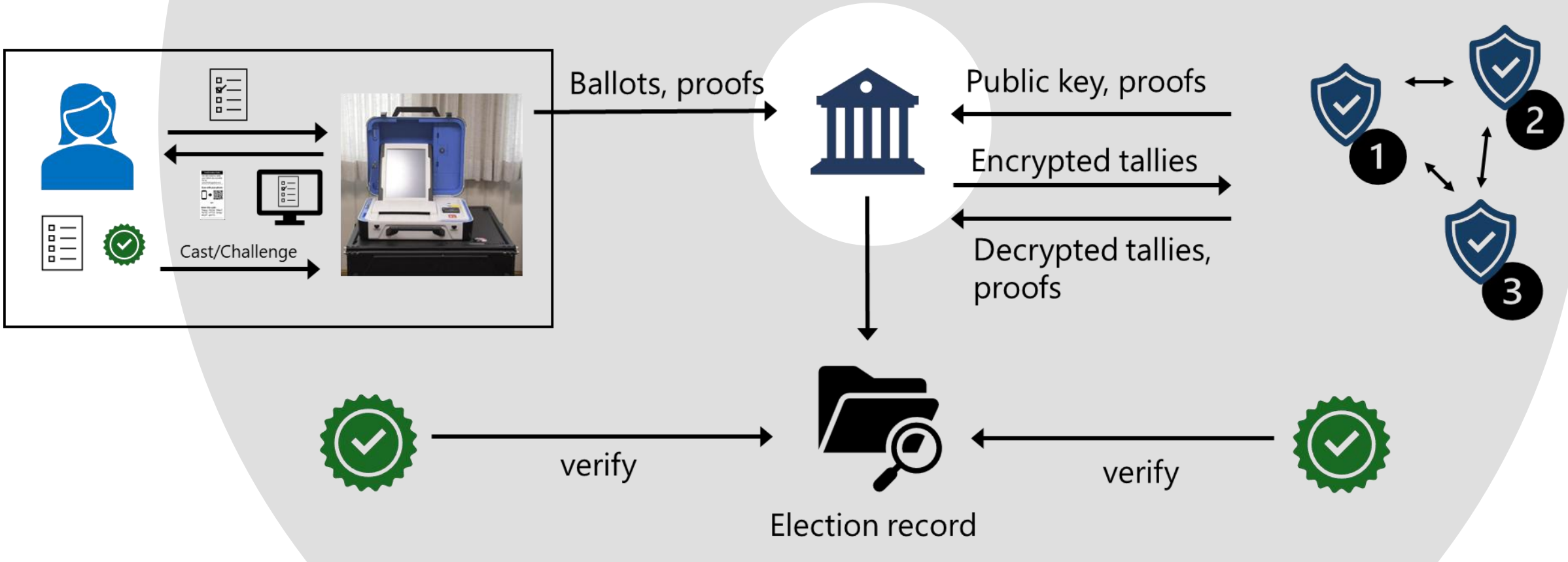
Cryptographic components

- Distributed key generation
- Verifiable decryption of tallies and challenged ballots

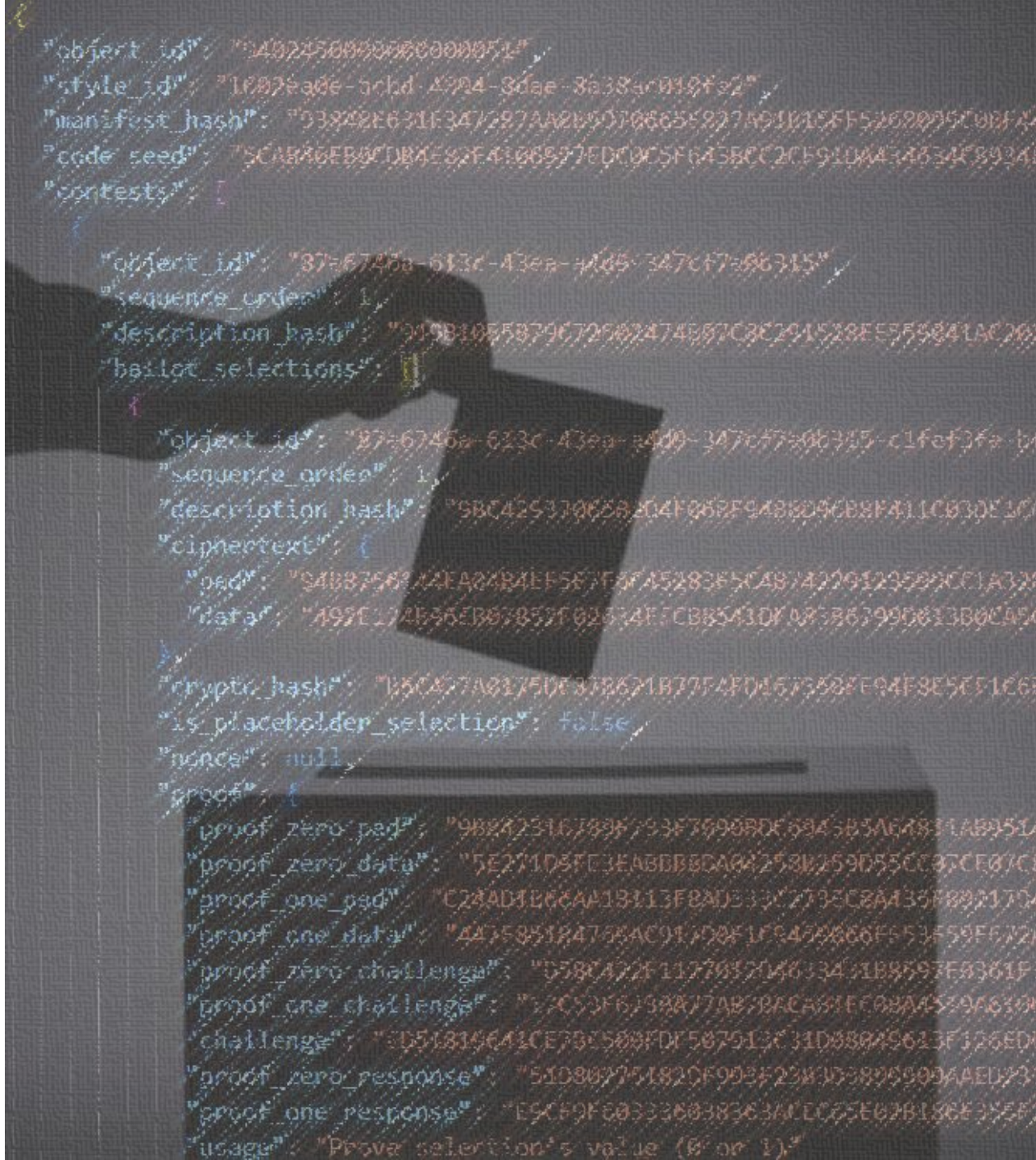


Cryptographic components

- Homomorphic tallying
- Facilitate decryption protocols
- Verify proofs



Cryptographic Components



Cryptographic parameters

Cyclic group of order $q = 2^{256} - 189$

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF43

32B

Multiplicative group in a finite field of size p (4096 bits)

6A3D90A5EA888A04AA367E9F24CB70F7BDE7409452451A92E3D39D98CB4CFCC755572F15AE879FCE930FBBF7C08F37FCF4B42D7C58C6BDADEF
3FABEFD7BDF32FA8BF6017FC292C730FE8D66F21B33A146F9D591F8CE9E0F8CEAA71894F32223245258214FB3C1FB17CB2F57C02F6B6B07C5A
86ACEAAA554EDE87A5A1A54E2EC3CD87E1034D23F824204692893C86C17BCA291D0896D9B50D755451BA6C8A3BC1A2A3ECBB10D6BC293A05BA
C675F60E2615F532646E26FD14C7F83643D4CEDCC1380F020B52B912C907E11F4B23DCAAA19F63F49DE22538130A36F40E4906C8BAD24413BF
C2BFE02B85240FFB8BFD679E007993335879743E6101020681A7879057C44CAA3EFA6B40AADE742F0F5009DEF03EBC12B7142AEB48BA1C06AC
430195928F889E7BFBFD5BC1D3990CE037A75CC0C6002A8617BAB3A493AC040CE81378976D9B1E4ECA99533E099011D888250DEEBEBA5F0900
183847D48C45195049FF89A37A4D209179D3FA09839D18F6AD0C48F79595DE968931302CA8E6CE7FF5F5B30AE9C7C95E0C502EA80FADC42418
278312982E44D9994980EE1DB8841C30AF79837238B5C87716460B9A4739C4B544DD8B4B0232D63FF92CA13B4C5CF2D088132B741FBD4376DA
6C17B89541D759EA0B5F17D7E11C8EAB3386EC20BBF1FF0BF2EEA83CDC550279DBF18809771B303A7BAFF3B029D905754932F776A5C25FDB

Cryptographic parameters

Cyclic group of order $q = 2^{256} - 189$

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF43

32B

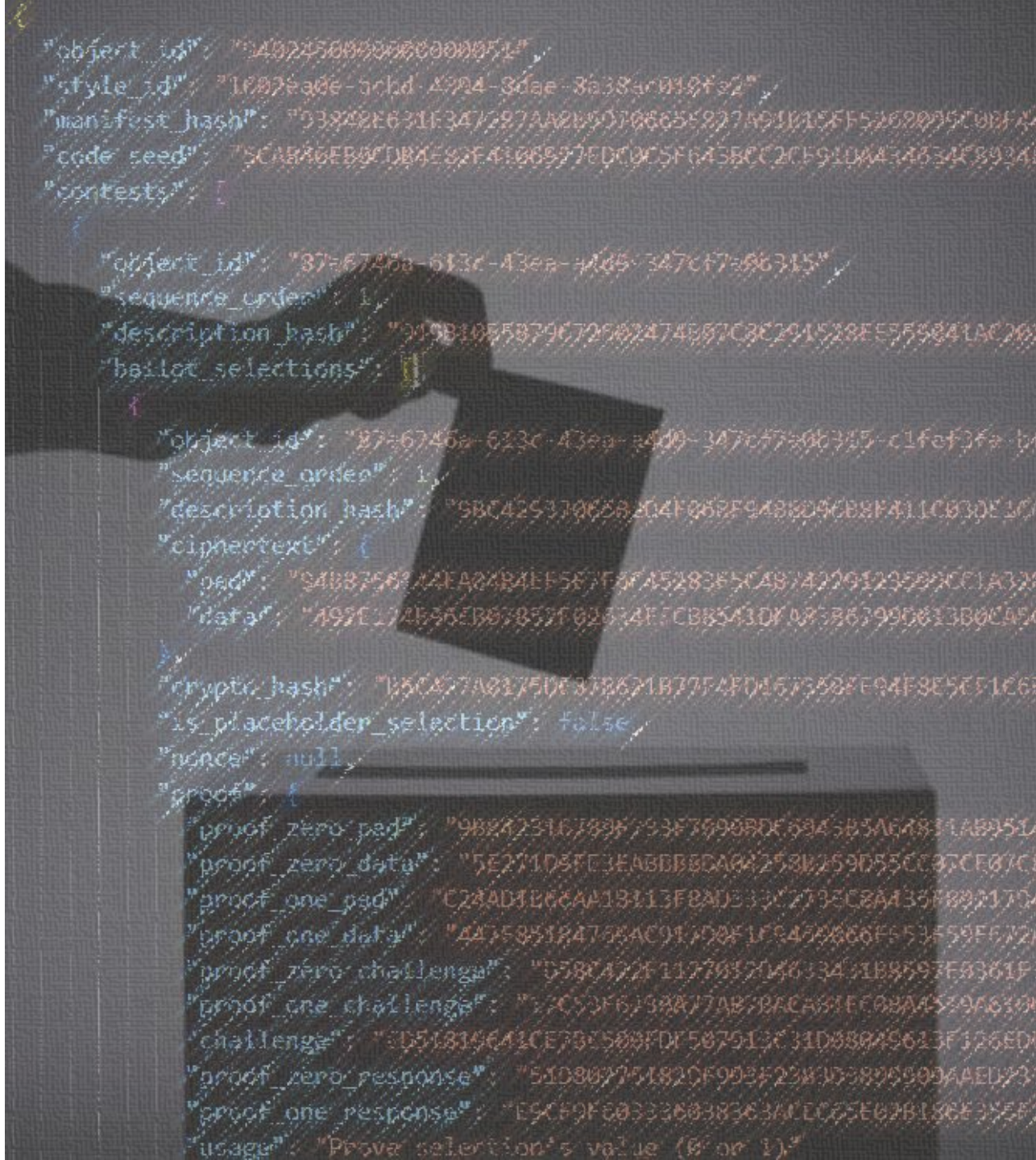
Multiplicative group in a finite field of size p (4096 bits)

512B

Generator g of order q $\quad g^q \equiv 1 \bmod p$

Operations: multiplication, exponentiation $\bmod p$

How are votes encrypted?



Exponential ElGamal public key encryption

Public key

Secret key

$$K = g^s$$

Encryption

$$E(m, r) = (g^r, g^m \cdot K^r)$$

Decryption with secret key

$$(g^r)^s = K^r$$

$$g^m = (g^m \cdot K^r)/(g^r)^s$$

$$m = \mathrm{DL}_g(g^m)$$

Homomorphism

$$\left(g^{r_1} \cdot g^{r_2}, (g^{m_1} \cdot K^{r_1}) \cdot (g^{m_2} \cdot K^{r_2})\right) = (g^{r_1+r_2}, g^{m_1+m_2} \cdot K^{r_1+r_2})$$

Exponential ElGamal public key encryption

Public key

Secret key

$$K = g^s$$

Encryption

$$E(m, r) = (g^r, K^m \cdot K^r)$$

Decryption with secret key

$$(g^r)^s = K^r$$

$$K^m = (K^m \cdot K^r)/(g^r)^s$$

$$m = \mathrm{DL}_K(K^m)$$

Homomorphism

$$\left(g^{r_1} \cdot g^{r_2}, (K^{m_1} \cdot K^{r_1}) \cdot (K^{m_2} \cdot K^{r_2})\right) = (g^{r_1+r_2}, K^{m_1+m_2} \cdot K^{r_1+r_2})$$

Exponential ElGamal public key encryption

Public key

Secret key

$$K = g^s$$

Encryption

$$E(m, r) = (g^r, K^{m+r})$$

Decryption with secret key

$$(g^r)^s = K^r$$

$$K^m = (K^{m+r})/(g^r)^s$$

$$m = \mathrm{DL}_K(K^m)$$

Homomorphism

$$\left(g^{r_1} \cdot g^{r_2}, (K^{m_1+r_1}) \cdot (K^{m_2+r_2})\right) = (g^{r_1+r_2}, K^{(m_1+m_2)+(r_1+r_2)})$$

Exponential ElGamal public key encryption

Public key

Secret key

$$K = g^s$$

Encryption

$$E(m, r) = (g^r, K^{m+r})$$

Decryption with secret key s

$$(g^r)^s = K^r$$

$$K^m = (K^{m+r})/(g^r)^s$$

$$m = \mathrm{DL}_K(K^m)$$

Decryption with random value r

$$K^r$$

$$K^m = (K^{m+r})/K^r$$

$$m = \mathrm{DL}_K(K^m)$$

Homomorphism

$$\left(g^{r_1} \cdot g^{r_2}, (K^{m_1+r_1}) \cdot (K^{m_2+r_2})\right) = (g^{r_1+r_2}, K^{(m_1+m_2)+(r_1+r_2)})$$

Vote encryption

Encrypts a vote $m \in \{0,1\}$

$$E(m,r) = (\alpha,\beta) = (g^r, K^{m+r})$$

Option not selected: $E(0,r) = (\alpha,\beta) = (g^r, K^r)$

Option selected: $E(1,r) = (\alpha,\beta) = (g^r, K^{1+r}) = (g^r, K \cdot K^r)$

Homomorphic tallies

Multiply all encrypted votes per selection for all selections in all contests across all ballots

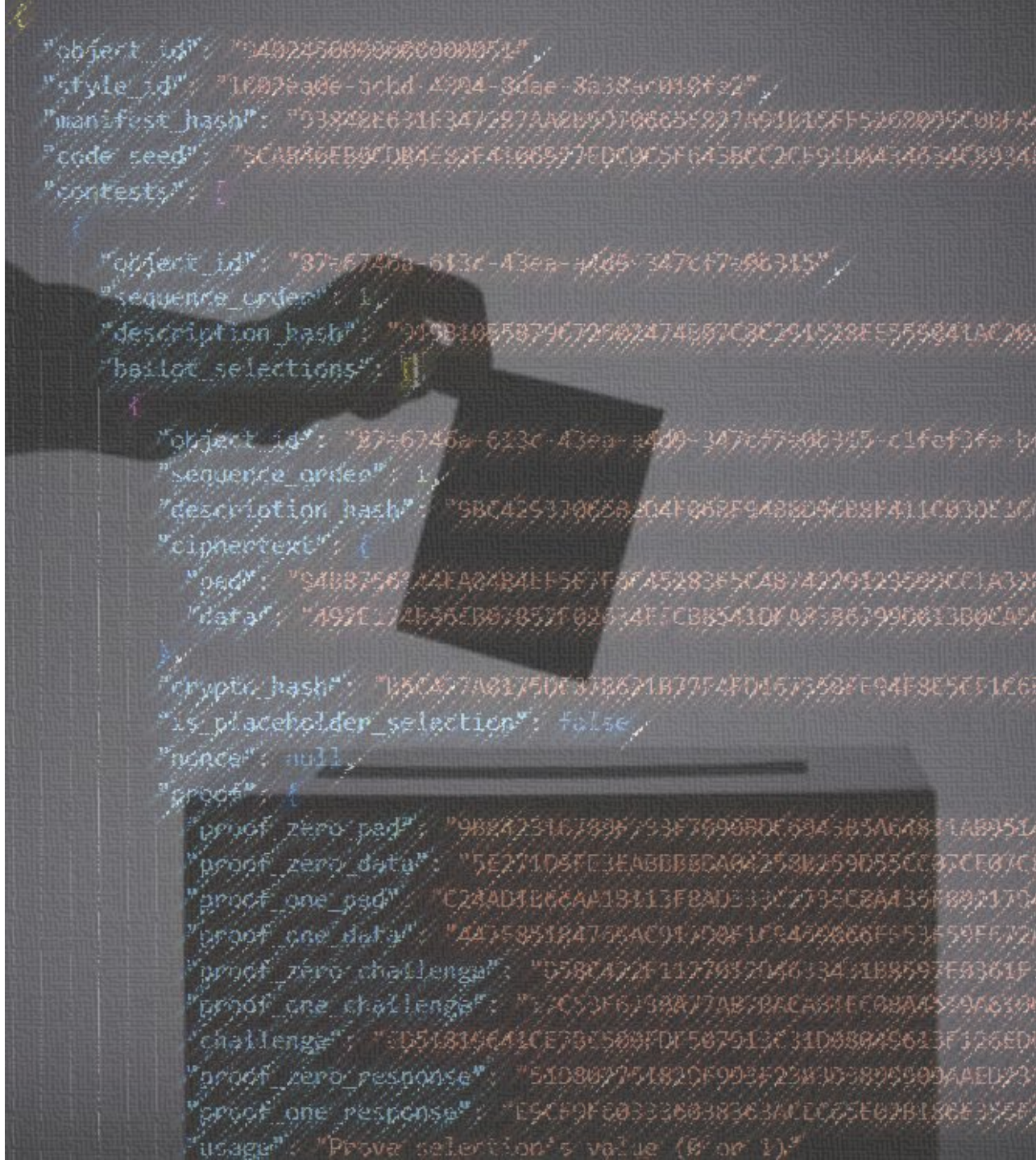
Encrypted votes for a single selection in a contest across all ballots

$$E(m_i, r_i) = (\alpha_i, \beta_i) = (g^{r_i}, K^{m_i + r_i})$$

Encrypted tally for that selection

$$(A, B) = (\prod \alpha_i, \prod \beta_i) = (g^{\sum r_i}, K^{\sum m_i + \sum r_i})$$

How to ensure a ballot is well-formed?



Proof of well-formedness

Encrypts a vote $m \in \{0,1\}$

$$E(m, r) = (\alpha, \beta) = (g^r, K^{m+r})$$

Prove that $(\alpha, \beta) = (g^r, K^r)$ is an encryption of 0 or 1 with an or-proof

	Commitment	Challenge	Response
0	$(a_0, b_0) = (g^{u_0}, K^{u_0})$	$c_0 = c - c_1$	$v_0 = u_0 - c_0 \cdot r$
		$c = H(K, \alpha, \beta, a_0, b_0, a_1, b_1)$	
1	$(a_1, b_1) = (g^{u_1}, K^{u_1 - c_1})$	c_1	$v_1 = u_1 - c_1 \cdot r$

Total size of encrypted option: (1024 + 128)B = 1152B

Proof

$$\pi = (c_0, v_0, c_1, v_1)$$

0/1 128B

Contest encryption

Contest			ElectionGuard
Alice		0	
Bob		0	
Carol	✓	1	
Dave		0	

(4*1152)B = 4608B

Selection limits

Prove that there are not more votes in a contest than allowed

Encrypted votes for all selections in contest

$$E(m_i, r_i) = (\alpha_i, \beta_i) = (g^{r_i}, K^{m_i + r_i})$$

Encrypted vote total in contest

$$(A, B) = (\prod \alpha_i, \prod \beta_i) = (g^{\sum r_i}, K^{\sum m_i + \sum r_i})$$

If voters must select exactly $L = 1$ option in the contest,

prove that the ciphertext encrypting the sum is an encryption of 1.

Selection limits

Contest			ElectionGuard
Alice		0	0/1
Bob		0	0/1
Carol	✓	1	0/1
Dave		0	0/1
Total		**1**	L=1

Range proofs

Contest			ElectionGuard
Alice		0	0..1
Bob		0	0..1
Carol	✓	1	0..1
Dave		0	0..1
Total		**1**	0..2

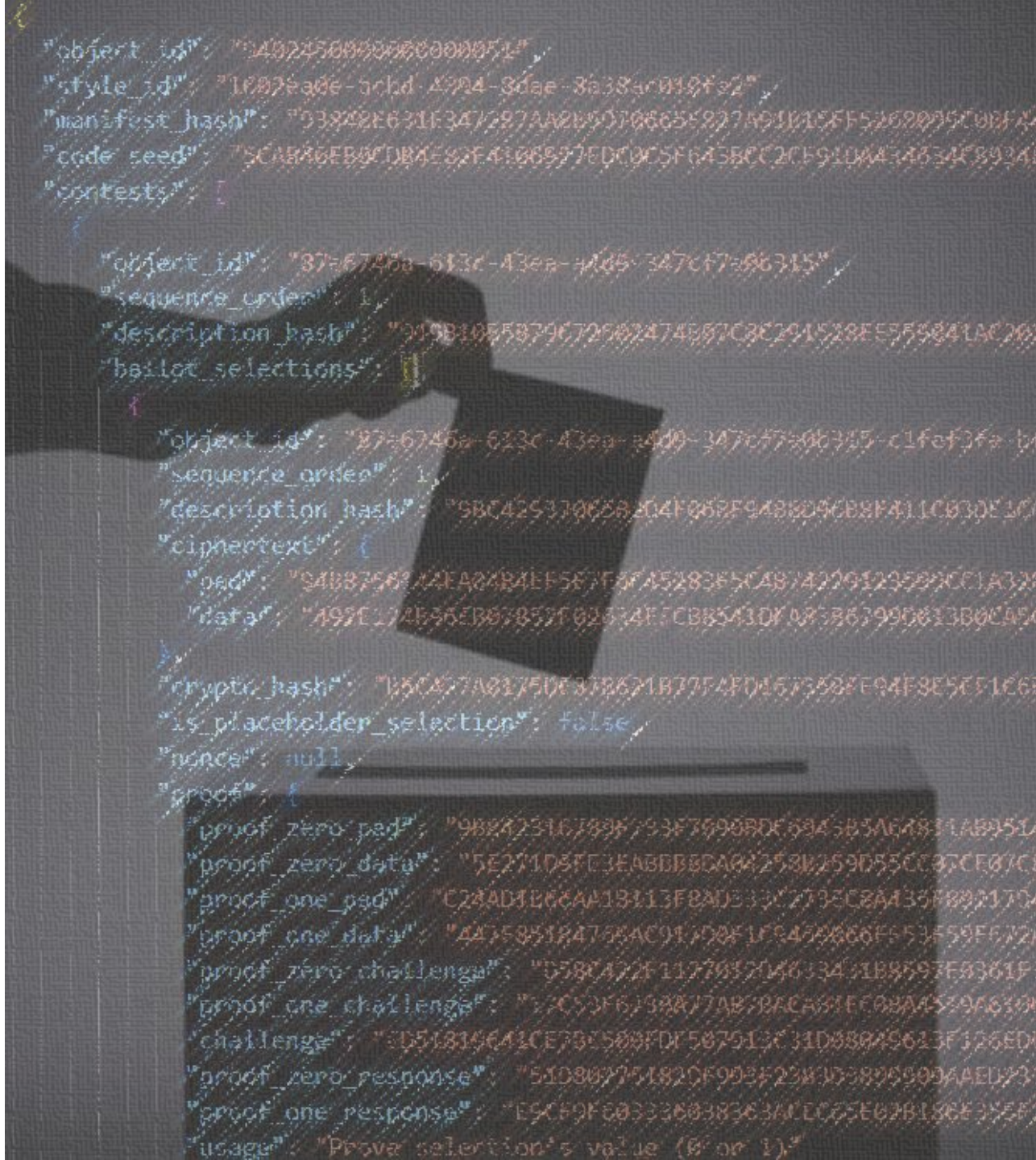
Range proofs

Contest			ElectionGuard
Alice	✓	1	0..1
Bob		0	0..1
Carol	✓	1	0..1
Dave		0	0..1
Total		**2**	0..2

Range proofs

Contest			ElectionGuard
Alice	✓	1	0..3
Bob		0	0..3
Carol	✓✓	2	0..3
Dave		0	0..3
Total		**3**	0..3

Confirmation Codes



Confirmation codes

Voters receive a confirmation code while submitting their ballot

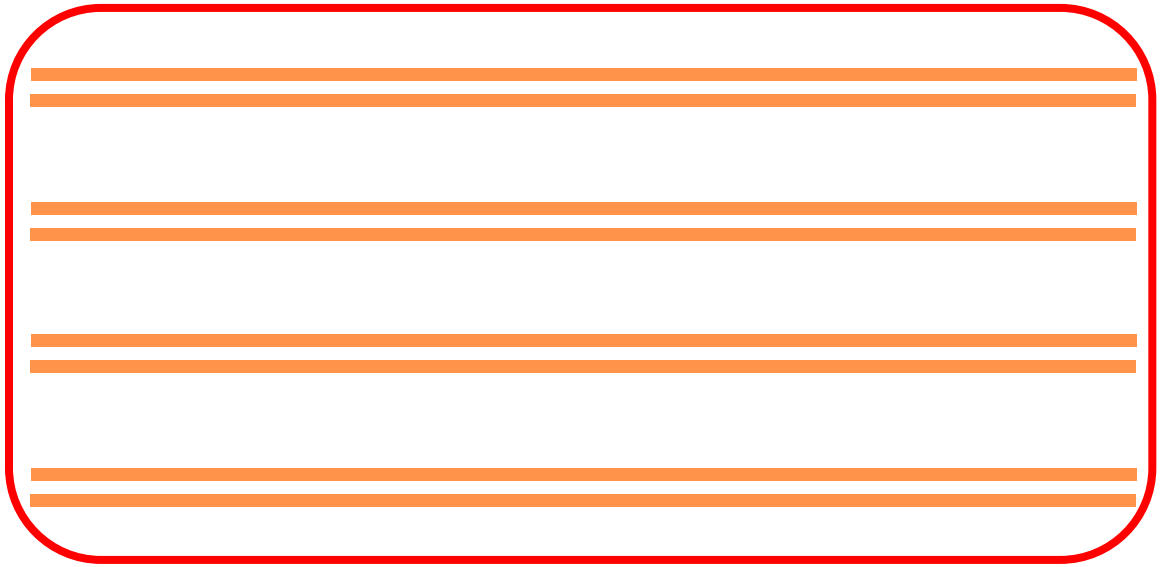
Hash of all vote encryptions over all contests on the ballot

Contest			ElectionGuard
Alice		0	
Bob		0	
Carol	✓	1	
Dave		0	
Total		**1**	

Confirmation codes

Voters receive a confirmation code while submitting their ballot

Hash of all vote encryptions over all contests on the ballot



Confirmation codes

Voters receive a confirmation code while submitting their ballot

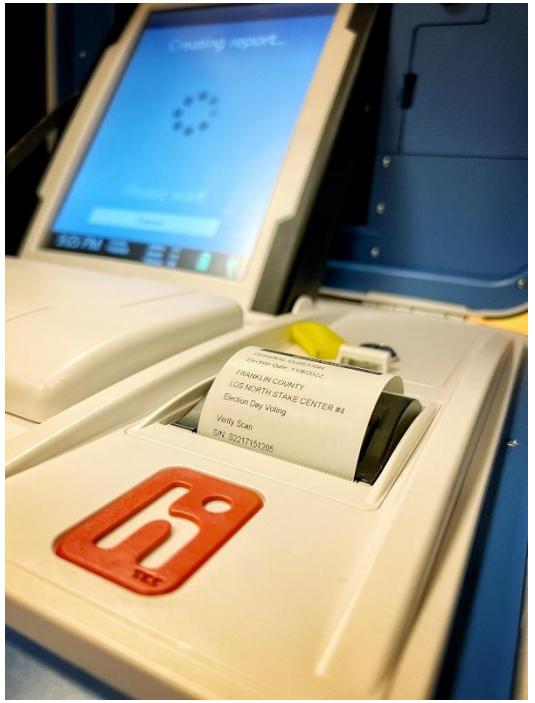
Hash of all vote encryptions over all contests on the ballot

Confirmation codes

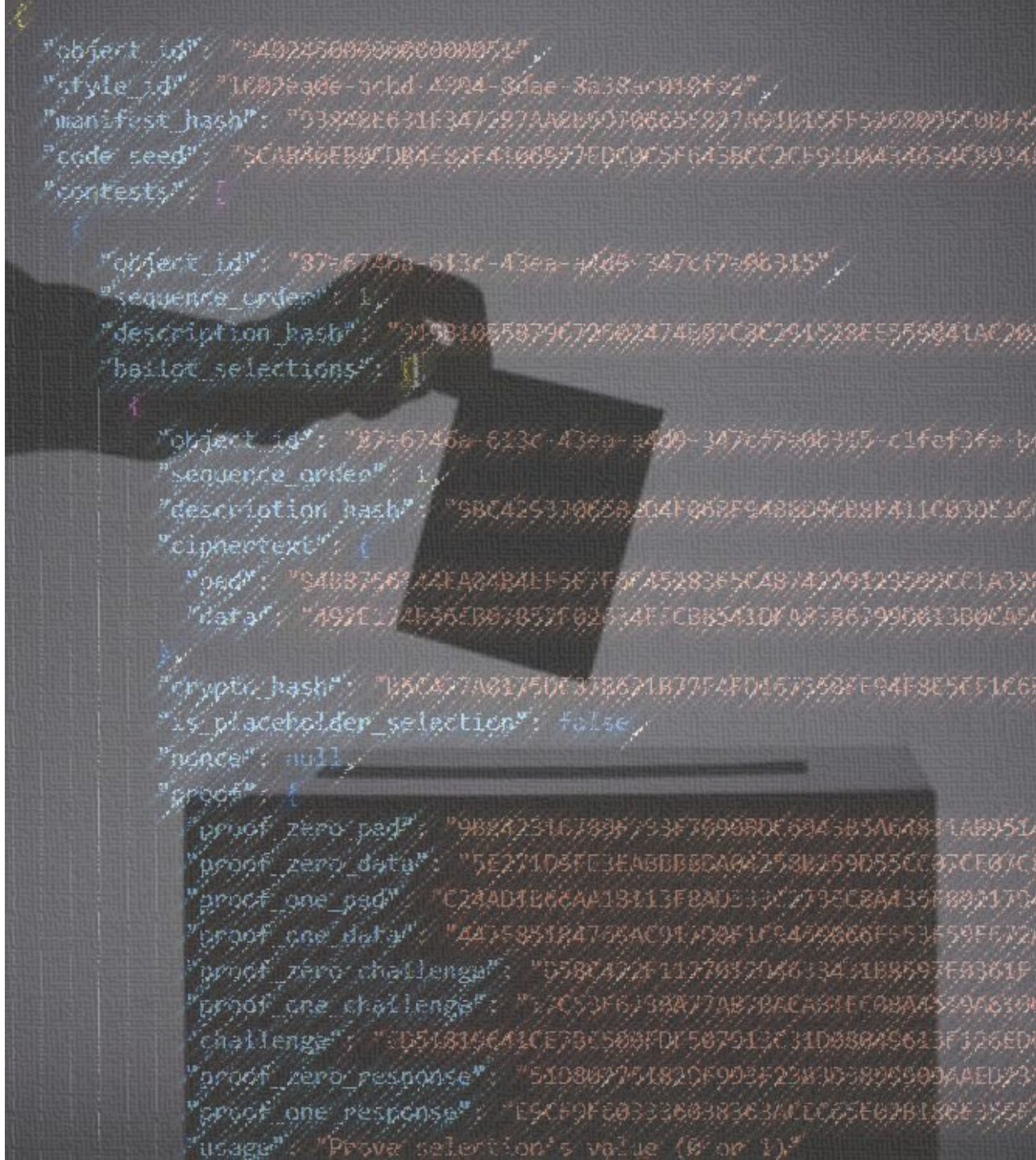
Voters receive a confirmation code while submitting their ballot

Hash of all vote encryptions over all contests on the ballot

H(▭ ▬ ▭)

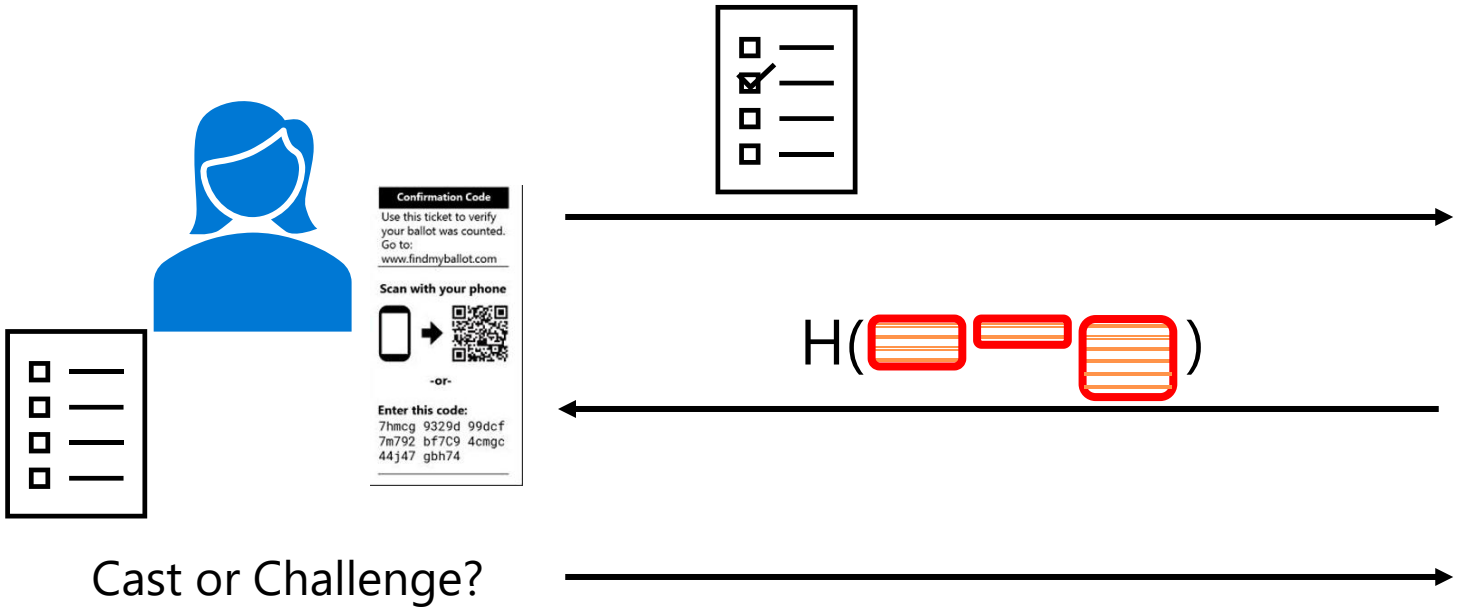


How can voters verify that their votes have been correctly encrypted?



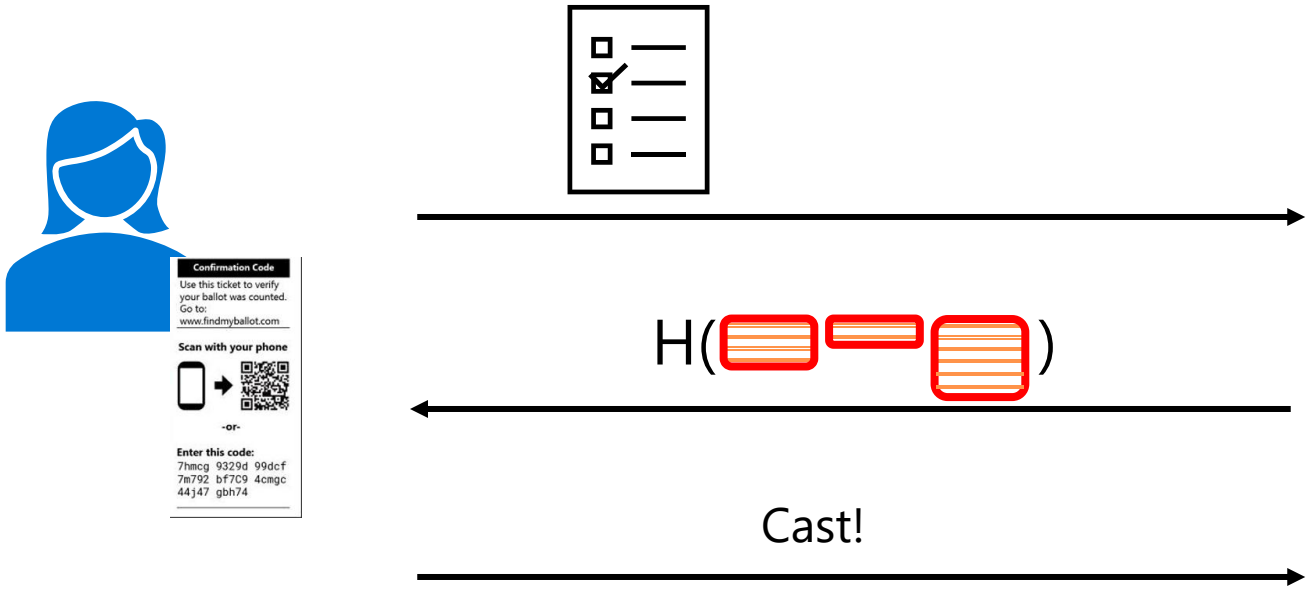
The Benaloh challenge

Challenge the encryption device in an unpredictable way.



The Benaloh challenge

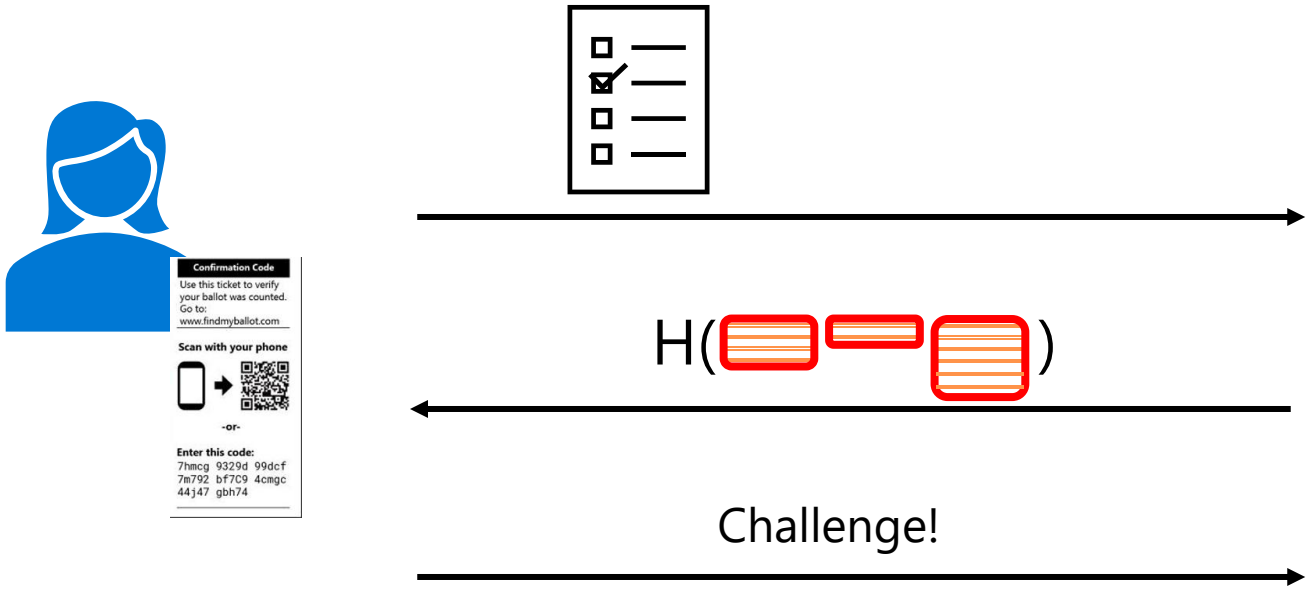
Challenge the encryption device in an unpredictable way.



If the voter decides to cast, the ballot is cast and the voting device records the encrypted ballot.

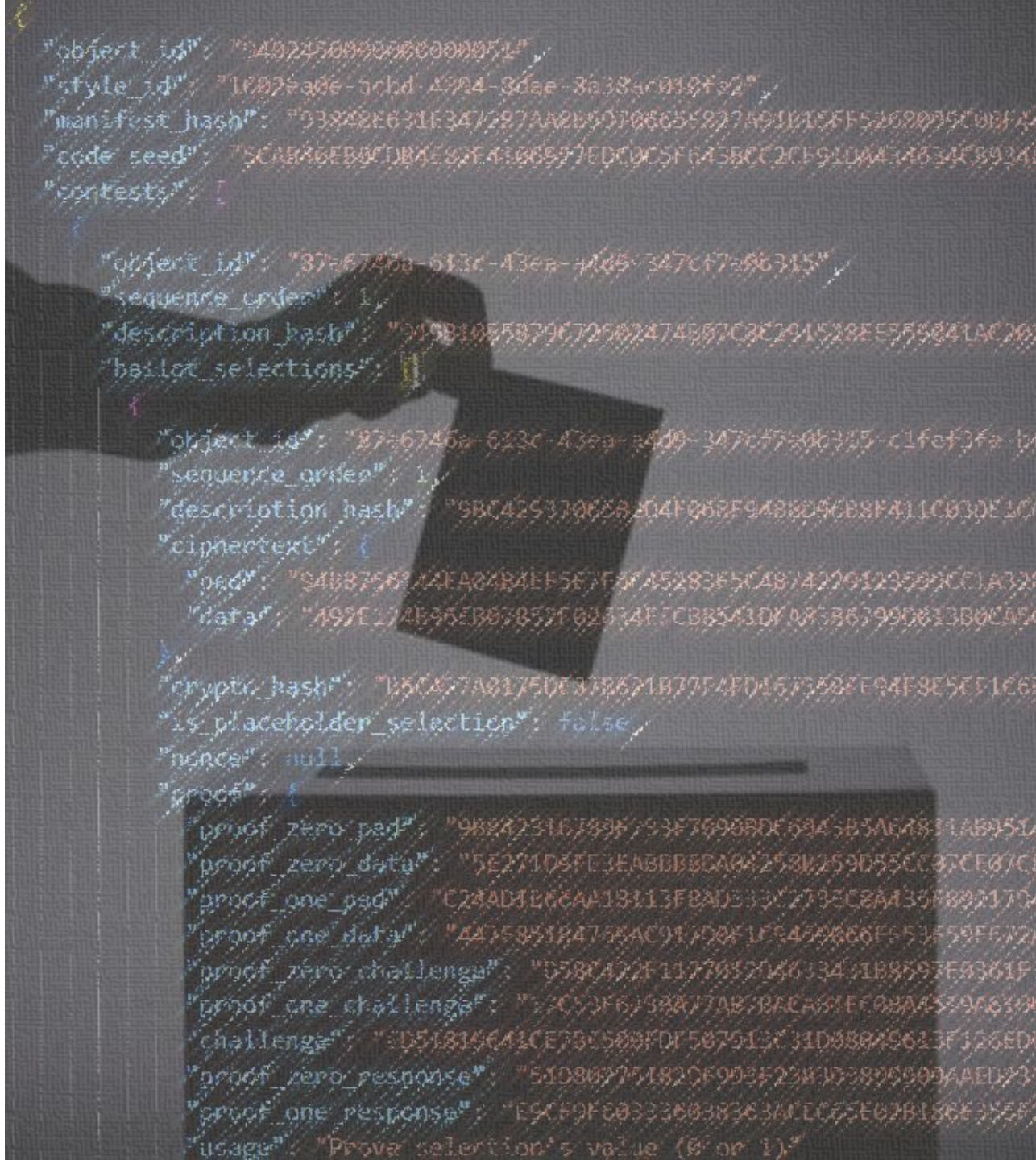
The Benaloh challenge

Challenge the encryption device in an unpredictable way.



If the voter decides to challenge, the ballot is opened and can be checked by the voter.

Who can decrypt tallies?



Election trustees and secret sharing

Election public key corresponds to a secret election key that is shared between n trustees (or guardians).

$K_1 = g^{s_1}$

Election public key

$$K = K_1 \cdot K_2 \cdot K_3 = g^{s_1+s_2+s_3}$$

$K_3 = g^{s_3}$

$K_2 = g^{s_2}$

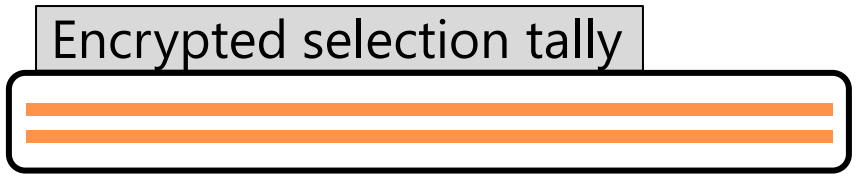
Election secret key

$(s = s_1 + s_2 + s_3)$

Each guardian secret key s_i is shared via Shamir secret sharing to allow threshold decryption with k available guardians.

Verifiable decryption

Trustees jointly decrypt the tallies



$$(A, B) = (g^R, K^{\sum m_i + R})$$

$$M_1 = A^{s_1}$$

$$M_2 = A^{s_2}$$

$$M_3 = A^{s_3}$$

$$\bar{M} = M_1 \cdot M_2 \cdot M_3 = A^s$$

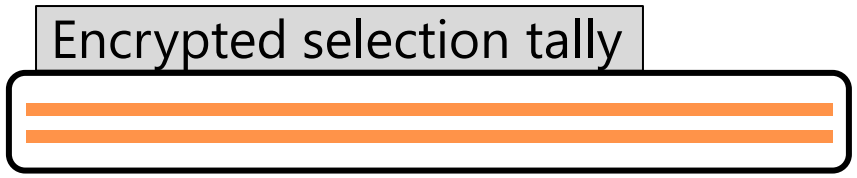
$$M = B / \bar{M} = K^{\sum m_i}$$

$$\sum m_i = \mathrm{DL}_K(M)$$

Solve a small discrete logarithm problem

Verifiable decryption

Trustees jointly decrypt the tallies and prove correct decryption



$$(A, B) = (g^R, K^{\sum m_i + R})$$

Commitments

$$M_1 = A^{s_1} \qquad (a_1, b_1) = (g^{u_1}, A^{u_1})$$

$$M_2 = A^{s_2} \qquad (a_2, b_2) = (g^{u_2}, A^{u_2})$$

$$M_3 = A^{s_3} \qquad (a_2, b_2) = (g^{u_2}, A^{u_2})$$

$$a = a_1 \cdot a_2 \cdot a_3 = g^{u_1 + u_2 + u_3}$$

$$b = b_1 \cdot b_2 \cdot b_3 = A^{u_1 + u_2 + u_3}$$

Verifiable decryption

Trustees jointly decrypt the tallies and prove correct decryption

Encrypted selection tally

$$(A, B) = (g^R, K^{\sum m_i + R})$$

Commitments

1 $M_1 = A^{s_1}$ $(a_1, b_1) = (g^{u_1}, A^{u_1})$

2 $M_2 = A^{s_2}$ $(a_2, b_2) = (g^{u_2}, A^{u_2})$

3 $M_3 = A^{s_3}$ $(a_2, b_2) = (g^{u_2}, A^{u_2})$

$$a = a_1 \cdot a_2 \cdot a_3 = g^{u_1+u_2+u_3}$$

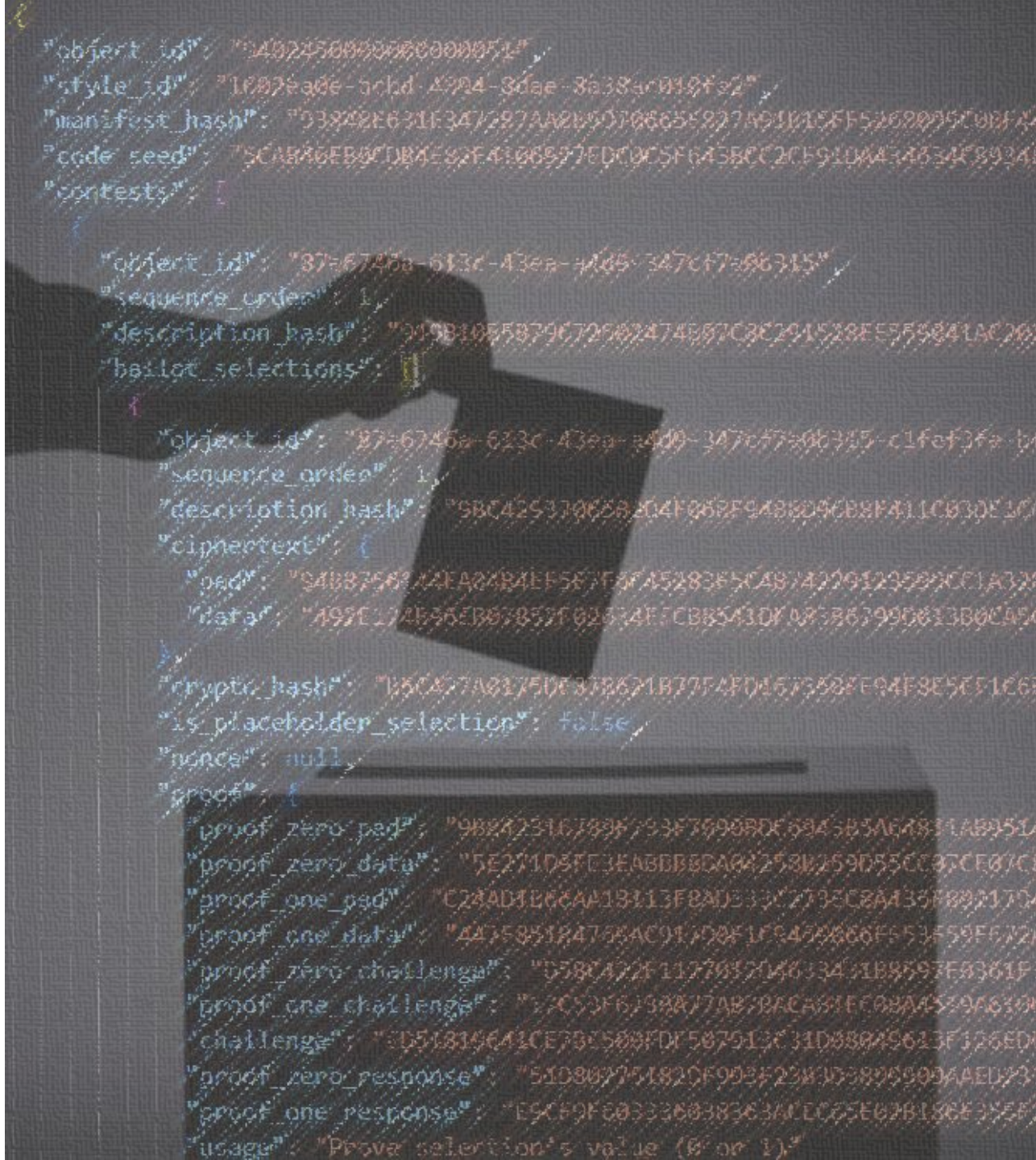
$$b = b_1 \cdot b_2 \cdot b_3 = A^{u_1+u_2+u_3}$$

$$v = v_1 + v_2 + v_3$$

$$\pi = (c, v)$$

Proof

Public evidence



The election record

The public record that provides the evidence to verify the election.

- Election manifest
- Encrypted submitted ballots
- Challenged ballots
- Encrypted and decrypted tallies
- Cryptographic proofs that the above are well-formed and correct

The election record is large:

Example ballot: 17 contests, 47 selectable options total

Ciphertexts + proofs:

47 * 1152B + 17 * 1152B
= 73728B > 70KB

~10000 ballots: > 700MB

What have we learned from pilots?

Challenges:

- Performance, interplay between parameters and devices

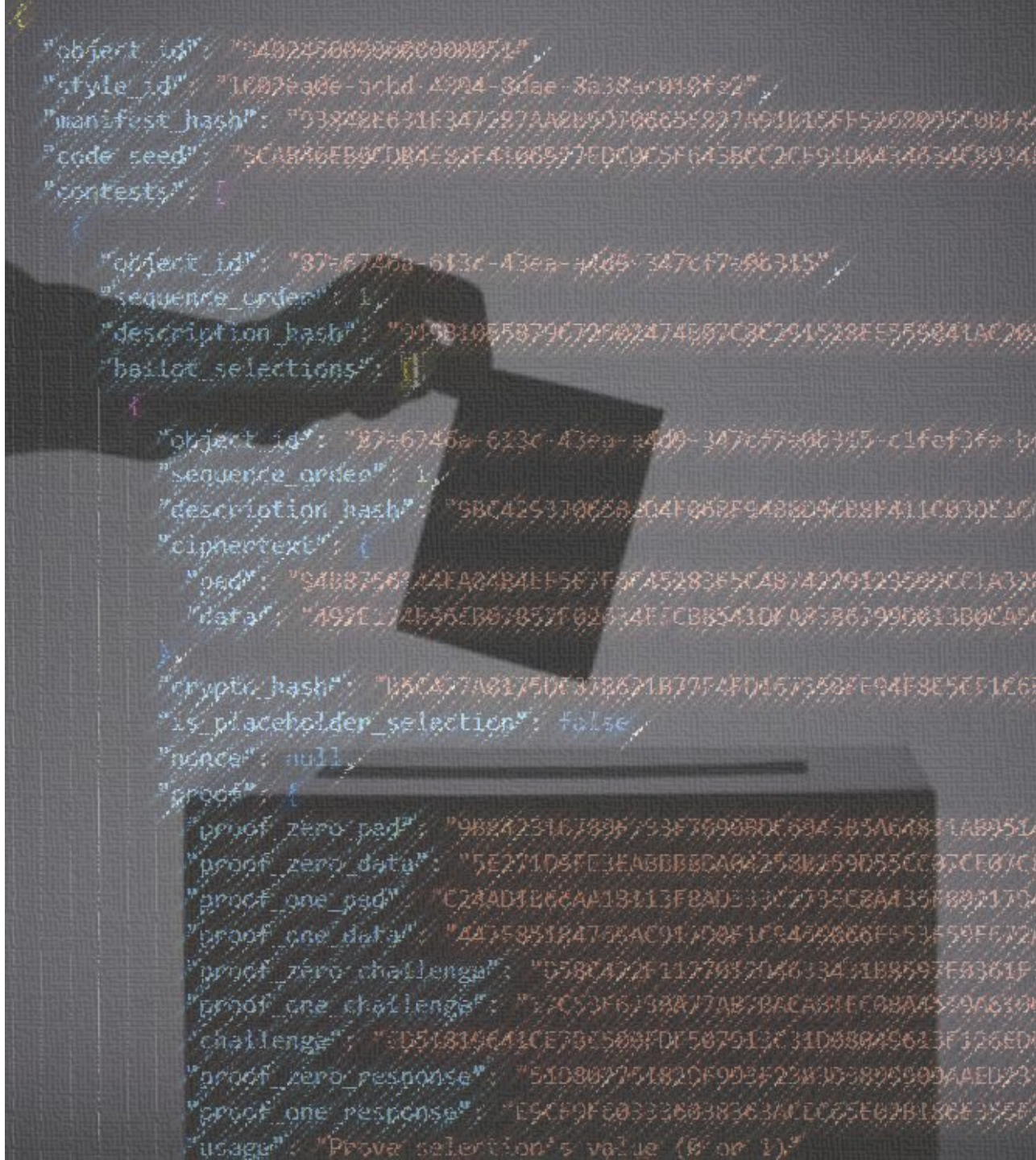
- Specifying how data is hashed is important!

- Size of the election record

- Challenge process in real time?

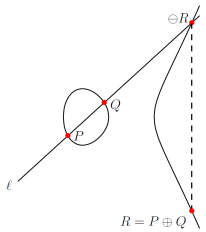
- Trustee process is complicated and requires interaction.
  For it to be meaningful, trustees must understand their role

What’s next?



Possible changes and extensions

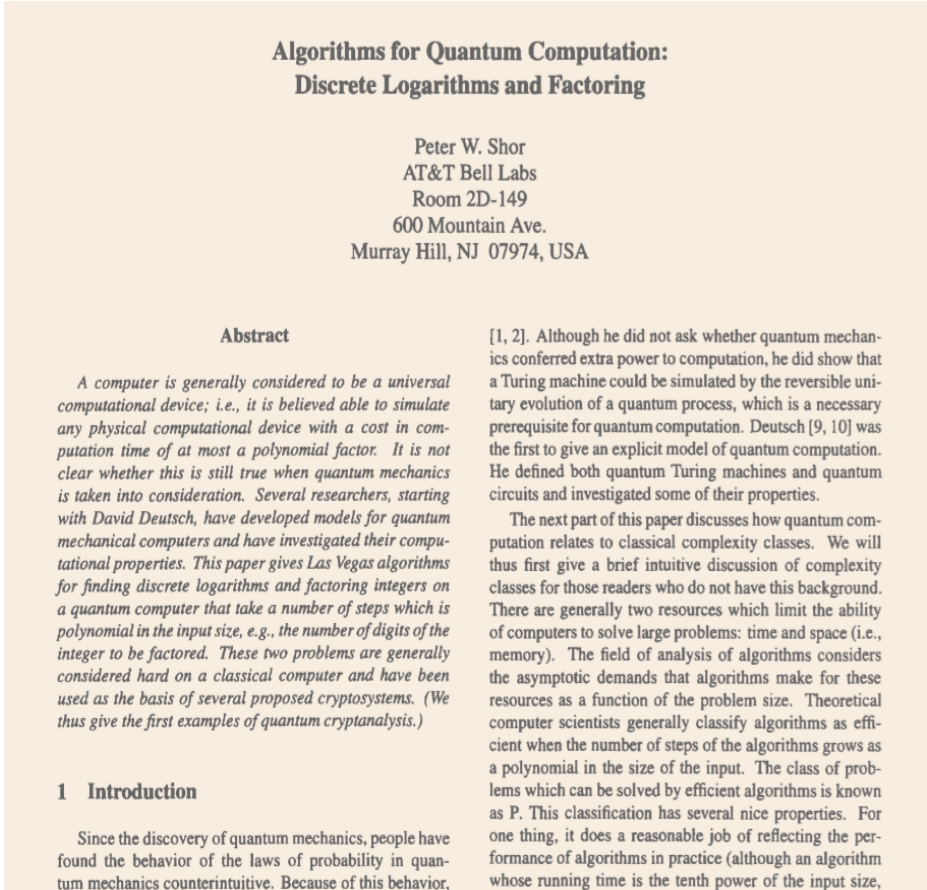
- Add functionality for verifiable mixnets.
- Replace human trustees with secure hardware?



- Use elliptic curve group instead of finite field group.
- Make EG post-quantum.

Post-quantum cryptography

- A large-scale quantum computer would be able to compute discrete logarithms in polynomial time
- Breaks ElGamal encryption and thus privacy of the votes
- Use post-quantum primitives, e.g. lattice-based cryptography



Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We thus give the first examples of quantum cryptanalysis.)

1 Introduction

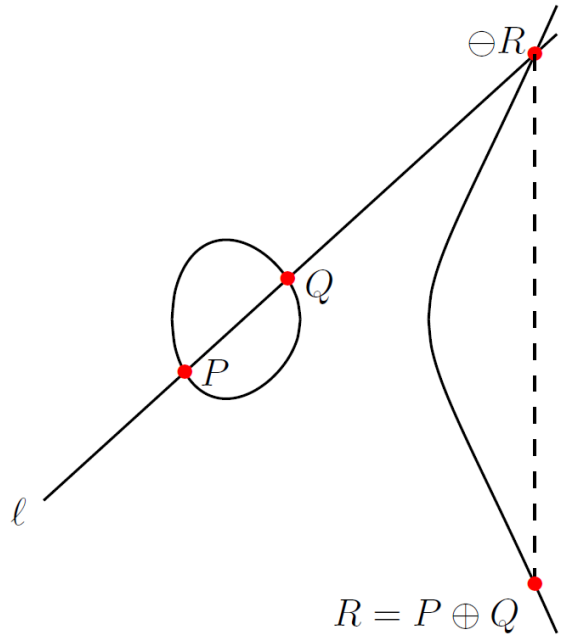
Since the discovery of quantum mechanics, people have found the behavior of the laws of probability in quantum mechanics counterintuitive. Because of this behavior,

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical computer scientists generally classify algorithms as efficient when the number of steps of the algorithms grows as a polynomial in the size of the input. The class of problems which can be solved by efficient algorithms is known as P. This classification has several nice properties. For one thing, it does a reasonable job of reflecting the performance of algorithms in practice (although an algorithm whose running time is the tenth power of the input size,

Elliptic Curves

Replace finite field group with elliptic curve group



$$E/\mathbb{F}_p\colon y^2 = x^3 - 3x + b$$

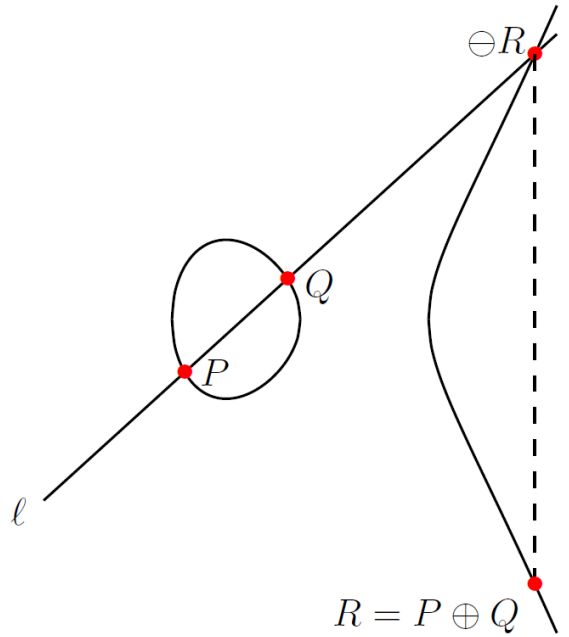
P-384

$$p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$$

p = 39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319

39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319

Elliptic Curves



Replace finite field group with elliptic curve group

$$E/\mathbb{F}_p\colon y^2 = x^3 - 3x + b$$

P-384

$$p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$$

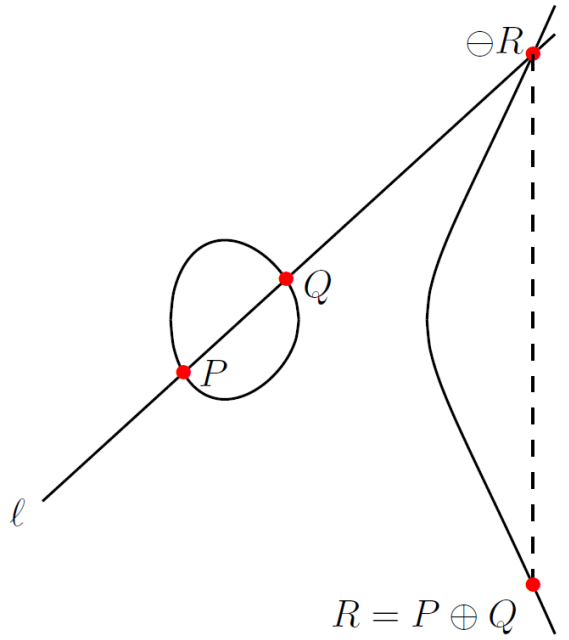
p = 39402006196394479212279040100143613805079739270465446667948293404245721771496870329047266088258938001861606973112319

48B

G = (26247035095799689268623156744566981891852923491109213387815615900925518854738050089022388053975719786650872476732087, 8325710961489029985546751289520108179287853048861315594709205902480503199884419224438643760392947333078086511627871)

$G = ($, $)$ → $G =$

Elliptic Curves



- Group elements are smaller for similar security
- Ciphertext can be represented by 2 elements

2*48B = 96B

- Proofs become slightly larger

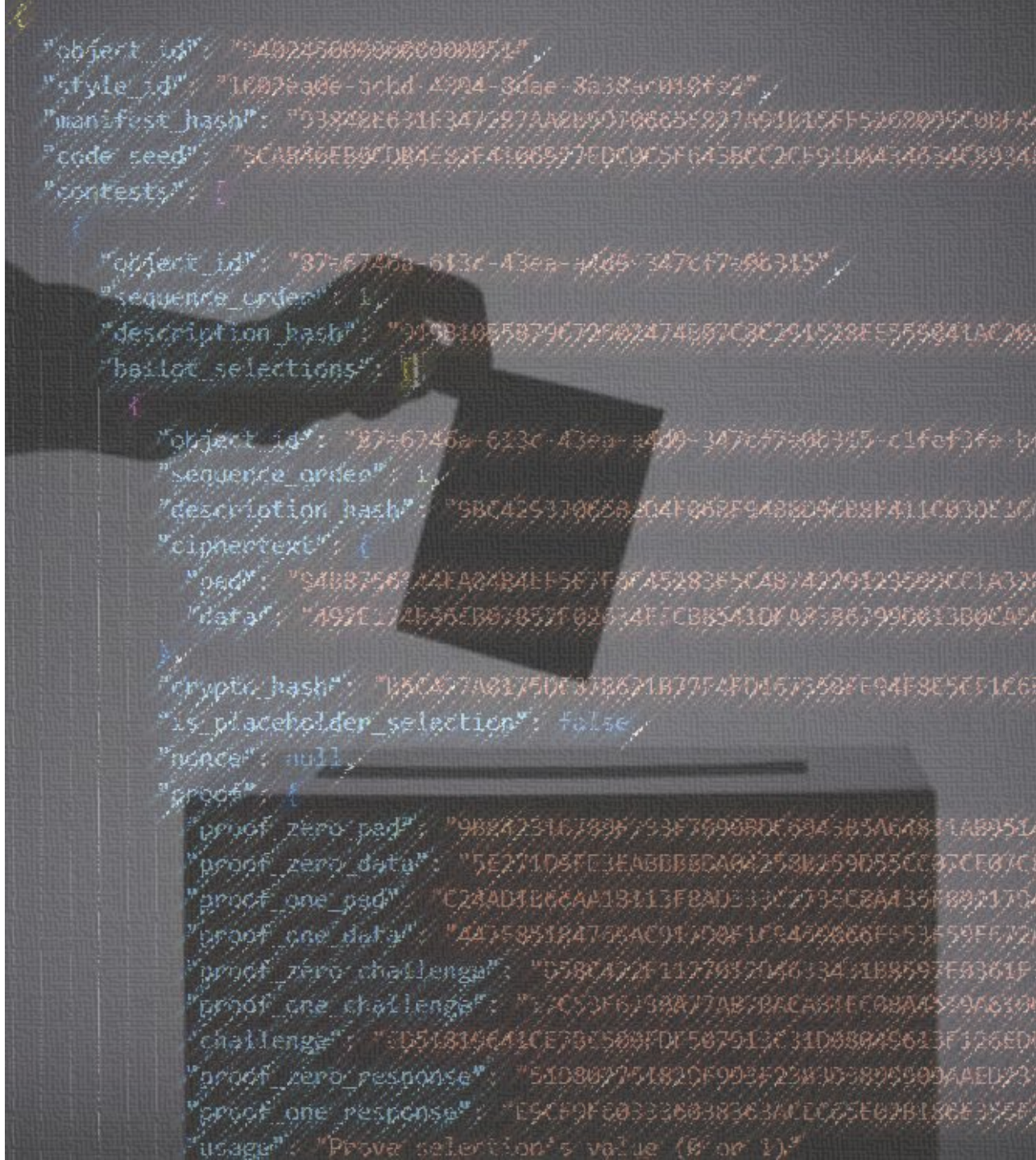
4*48B = 192B

0/1

- Example ballot from earlier:

64*192B = 12288B > 12KB
10000 ballots > 120MB

New directions



Commitments

Replace encryption with commitments

Encryption

$$E(m,r) = (g^r, K^{m+r})$$

$$K = g^s$$

Commitment

$$C(m,r) = g^r \cdot g_1^m$$

- Generators g and g_1 are generated publicly without knowledge of discrete logarithm between them.
- There are no keys anymore!
- Statistically hiding, computationally binding.

Everlasting privacy!

Commitments

Replace encryption with commitments

Encryption

$$E(m, r) = (g^r, K^{m+r})$$

Commitment

$$C(m, r) = g^r \cdot g_1^m$$

Encryption homomorphism

$$\left(g^{r_1} \cdot g^{r_2}, (K^{m_1+r_1}) \cdot (K^{m_2+r_2})\right)$$
$$= (g^{r_1+r_2}, K^{(m_1+m_2)+(r_1+r_2)})$$

Commitment homomorphism

$$\left(g^{r_1} \cdot g_1^{m_1}\right) \cdot \left(g^{r_2} \cdot g_1^{m_2}\right)$$
$$= (g^{r_1+r_2} \cdot g_1^{m_1+m_2})$$

Commitments

Replace encryption with commitments

Encryption

$$E(m, r) = (g^r, K^{m+r})$$

Commitment

$$C(m, r) = g^r \cdot g_1^m$$

$$E(m_1, r_1) = (g^{r_1}, K^{m_1+r_1}) \qquad C(m_1, r_1) = g^{r_1} \cdot g_1^{m_1}$$

$$E(m_2, r_2) = (g^{r_2}, K^{m_2+r_2}) \qquad C(m_2, r_2) = g^{r_2} \cdot g_1^{m_2}$$

$$E(m_3, r_3) = (g^{r_3}, K^{m_3+r_3}) \qquad C(m_3, r_3) = g^{r_3} \cdot g_1^{m_3}$$

$$E(m_4, r_4) = (g^{r_4}, K^{m_4+r_4}) \qquad C(m_4, r_4) = g^{r_4} \cdot g_1^{m_4}$$

Commitments

Replace encryption with commitments

Encryption

$$E(m, r) = (g^r, K^{m+r})$$

Commitment

$$C(m, r) = g^r \cdot g_1^m$$

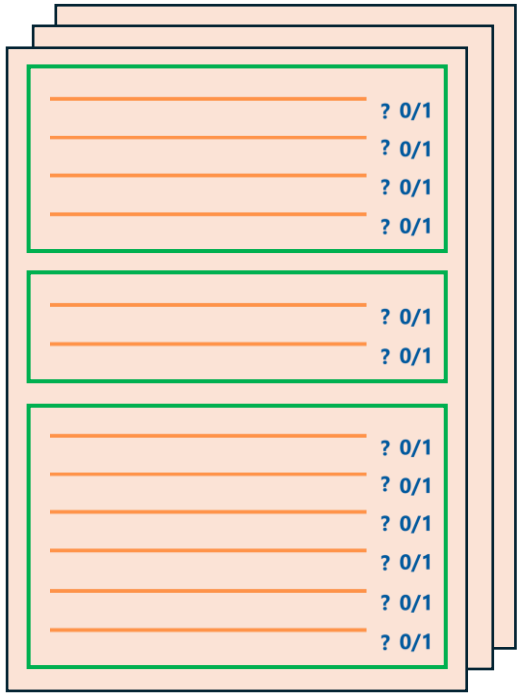
Contest			Encryption
Alice		0	0/1
Bob		0	0/1
Carol	✓	1	0/1
Dave		0	0/1

Contest			Commitment
Alice		0	? 0/1
Bob		0	? 0/1
Carol	✓	1	? 0/1
Dave		0	? 0/1

Tallying committed votes

Commitments to votes

$$C(m_i, r_i) = g^{r_i} \cdot g_1^{m_i}$$



Device keeps

Sum of random values, running tally, commitments

$$\sum m_i, \sum r_i, C(m_i, r_i)$$

Contest	$\sum m_i$	$\sum r_i$
Alice	314	—
Bob	47	—
Carol	376	—
Dave	12	—

Commitment to the tally for that selection

$$C = \prod C(m_i, r_i) = g^{\sum r_i} \cdot g_1^{\sum m_i}$$

Vector commitments

$$C(\vec{m}, r) = g^{r} \cdot g_1^{m_1} \cdot g_2^{m_2} \cdot g_3^{m_3} \cdot g_4^{m_4} \cdot \ldots \cdot g_n^{m_n}$$

- Per contest:
  A single $C(\vec{m}, r)$ = ——————
- Proof (with n selections):
  —————— + (2* n +6) —
- Per selection: 2 —
  Add 2 —————— + 6 — per contest

Example ballot:
47*2*32B + 17*(2*512B + 6*32B)
= 23680B > 23KB

Elliptic curve P-384:
47*2*48B + 17* (2*48B + 6*48B)
= 11040B > 11KB

Elliptic curve P-256:
47*2*32B + 17* (2*32B + 6*32B)
7360B > 7KB

Committing the whole ballot:
47*2*32B + 17*2*32B + 2*32B +
6*32B = 4352B > 4KB

ElectionGuard

https://www.electionguard.vote/
https://github.com/Election-Tech-Initiative/electionguard
https://electiontechnology.org/

Design specification (Josh Benaloh, M. N., Olivier Pereira):
https://www.electionguard.vote/spec/
Academic paper (Josh Benaloh, M. N., Olivier Pereira, Dan S. Wallach):
https://dl.acm.org/doi/10.5555/3698900.3699207

v2.0 implementations (John Caron):
https://github.com/JohnLCaron/egk-ec

The MITRE verifier:
https://electionintegrity.mitre.org/verifier/

ElectionGuard

Thank you:

Eion Blanchard, Ales Bizjak, Nicholas Boucher, John Caron, Henri Devillez, Joey Dodds, Felix Doerre, Gerald Doussot, Aleks Essex, Keith Fung, Pierrick Gaudry, Rainbow Huang, Chris Jeuell, Anunay Kulshrestha, Moses Liskov, Shreyas Minocha, Arash Mirzaei, Luke Myers, Karan Newatia, Thomas Peters, John Ramsdell, Marsh Ray, Dan Shumow, Vanessa Teague, Aaron Tomb, Daniel Tschudi, Daniel Wagner, Jake Waksbaum, Dan Wallach, Matt Wilhelm, Greg Zaverucha

The Microsoft Democracy Forward Team, RC Carter, Inferno Red, Hart Intercivic, Enhanced Voting, MITRE, Center for Civic Design, the jurisdictions that ran pilot elections

Jiwon Kim for working out the commitment-based scheme during her internship this summer