ElectionGuard - Open Source Election Security

Ethan Chumley, Senior Security Strategist, Microsoft, Defending Democracy Program

Matt Wilhelm, Senior Software Engineer, InfernoRed Technology, @addressXception

Cyber-enabled Threats to Democracy Continue

Microsoft

The 2020 Election Won't Look Like Any We've Seen Before

Bots and Trolls Elbow Into Mexico's Crowded Electoral Field

What's being done to stop Russia's election interference?

Mar 15, 2020 5:08 PM EDT

Democracy at risk due to fake news and data misuse, MPs conclude

Parliamentary inquiry to demand urgent action to combat 'relentless targeting of hyper-partisan views'

Twitter admits far more Russian bots posted on election than it had disclosed

Company says it removed more than 50,000 accounts and reported them to investigators, marking latest upward revision of figures

SECURITY

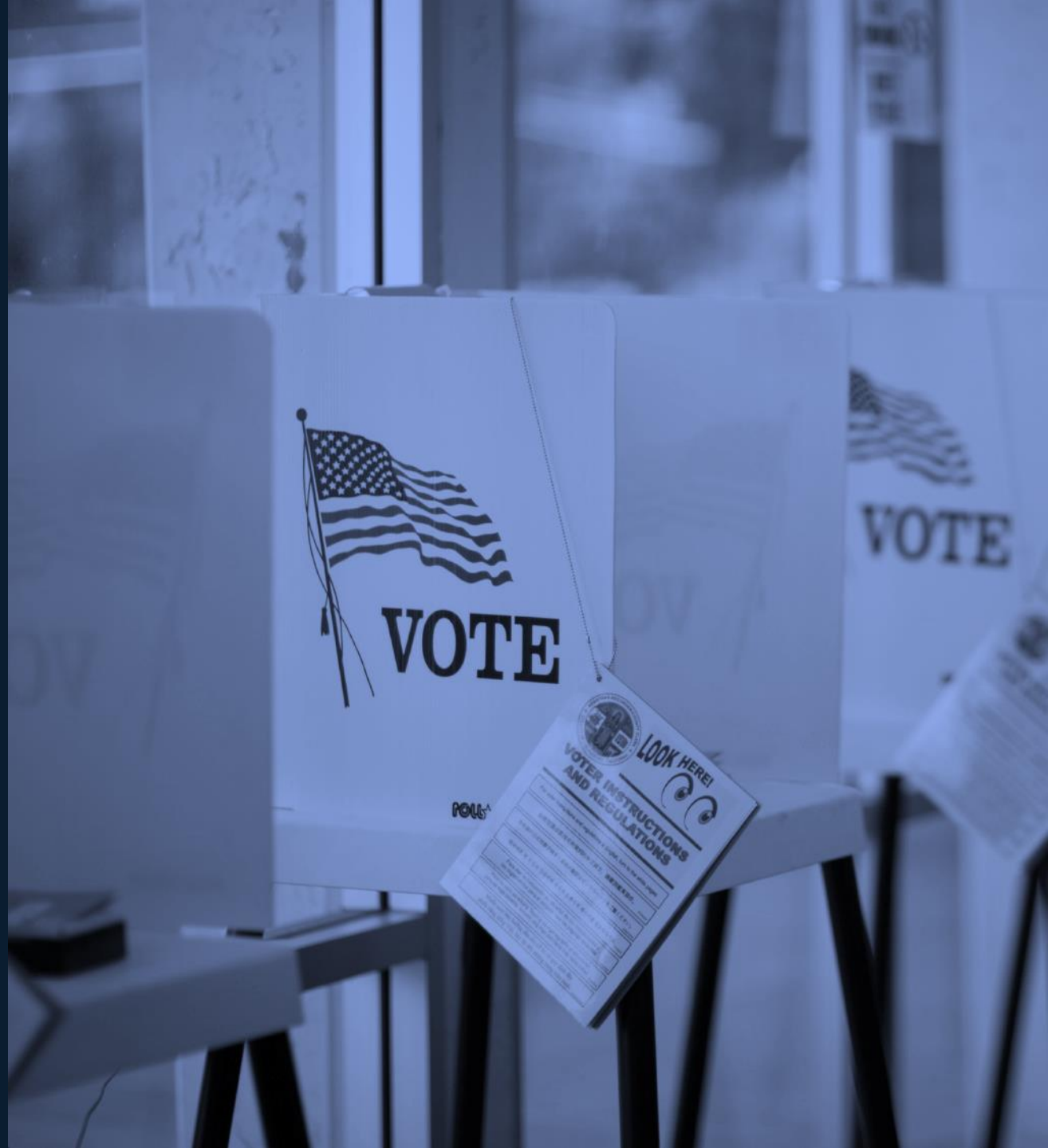
Iran-linked hackers tried to compromise presidential campaign, Microsoft says

The company said that it had seen "significant cyberactivity" from a group of hackers that it believes "originates from Iran and is linked to the Iranian government."

Our Challenge: Protect the Integrity of Elections

- Paper vs. Electronic
- Security vs. Accessibility
- Recently: In-person vs. Mail-In

- Ensuring that the integrity of the vote – that the official tally represents each voter's intent – is paramount
- Our design principles:
  - Highly secure, highly vetted
  - Respect voter privacy and secrecy
  - End-to-end verifiable
  - Auditable



What is End-to-End Verifiability?

Microsoft

End-to-End (E2E) Verifiability aims to answer the question:

How can I trust the accuracy of an election outcome …

if I think there could be a compromise in the software, hardware, or personnel responsible for conducting the election?

What is End-to-End Verifiability?

Microsoft

An election is *end-to-end verifiable* if:

1. Voters can verify that their own selections have been correctly recorded
2. Anyone can verify that all the recorded votes have been correctly tallied.

An election is secret-ballot if nobody is able to know the ballot selections of a specific voter

Introducing: ElectionGuard

Open Source

End-to-End Verifiable Election SDK

Supports any election style
(so long as there's an electronic component)

ElectionGuard

Microsoft

Key Ceremony

- Unique encryption keys generated for each election
- Multiple election “guardians” split and hold parts of the keys offline (eg: hardware key, Smartcard)

Homomorphic Encryption

- Each ballot is homomorphically encrypted.
- Unencrypted individual ballots are never stored or processed.

Tally

- Homomorphic tally allows final results to be calculated without violating voter secrecy or decrypting individual ballots
- Quorum of election officials must be present to do decryption

Public Verification

- Each voter given a unique Tracking ID
- All encrypted election records are publicly published following an election- allows watchdogs to verify the tally and verify no tampering using mathematical proofs

Homomorphic Encryption 101

Microsoft

In “Traditional” Static Encryption (eg: AES), the only thing you can do with encrypted data is decrypt it.

oiyotwfSLrZmLOTa6LmP5Q → *SANS Hackfest*

However, some modern encryption methods allow for useful computations on encrypted data, such as basic addition, without the need for decryption.

FYckqVmHGv + icmybfT5U = NBPdHAo5o

NBPdHAo5o → [Adams: 2, Jefferson: 0]

Using homomorphic encryption in an elections context allows us to tally (add) individual encrypted votes to get an encrypted sum total. We can then decrypt only the encrypted total without the need to violate ballot secrecy and ever decrypt any one particular ballot.

Code Sample

Microsoft

```python
def encrypt_selection(
    selection: PlaintextSelection,          # The plaintext selection (e.g. "True" or "False")
    metadata: SelectionDescription,         # The Metadata for the selection (used for hashing)
    public_key: ElementModP,                # The public key used to encrypt the election
    nonce_seed: ElementModQ                 # A random number derived for the ballot
) -> CiphertextSelection:

    if selection.is_valid_for(metadata):
        selection_nonce = sha256(metadata.hash, nonce_seed)         # derive a secret value

        encryption = elgamal_encrypt(
            selection, public_key, selection_nonce                  # encrypt the plaintext
        )
        disjunctive_cp_proof = make_disjunctive_proof(
            selection, encryption, public_key, selection_nonce      # create a proof
        )

        if disjunctive_cp_proof.is_valid(encryption, public_key)
            return CiphertextSelection(
                selection.id, encryption, disjunctive_cp_proof      # return the encrypted value
            )
```

Encrypted Ballot Output

Microsoft


```
"ballotSelections": [
    {
        "objectId": "john-adams-selection",
        "message": {
            "public_key": "XTdnWdwTRlSvIyGwfeqy3...",
            "ciphertext": "AacWdwzV4hqL9v6HGFDr5..."
        },
        "extended_data": "Y8FRUlFpoU9MNzHPh7lgD...",
        "proof": {
            "zero_proof": "EQVwcWdwTzV14hqLXNSo...",
            "one_proof": "sTINrnX1RmDYxUXbTPos..."
        }
    },
    ...
    {
        "objectId": "undervote-placeholder-selection-1"
    },
    {
        "objectId": "undervote-placeholder-selection-2"
    }
],

"proof": "AQABAA0M51gdSdcDjj+aP..."
```


The output format is the same structure as the input format, but now encrypted

The Proofs are key:
"Zero-Knowledge Proofs" are a cryptographic assurance that math was done correctly, without revealing secret values

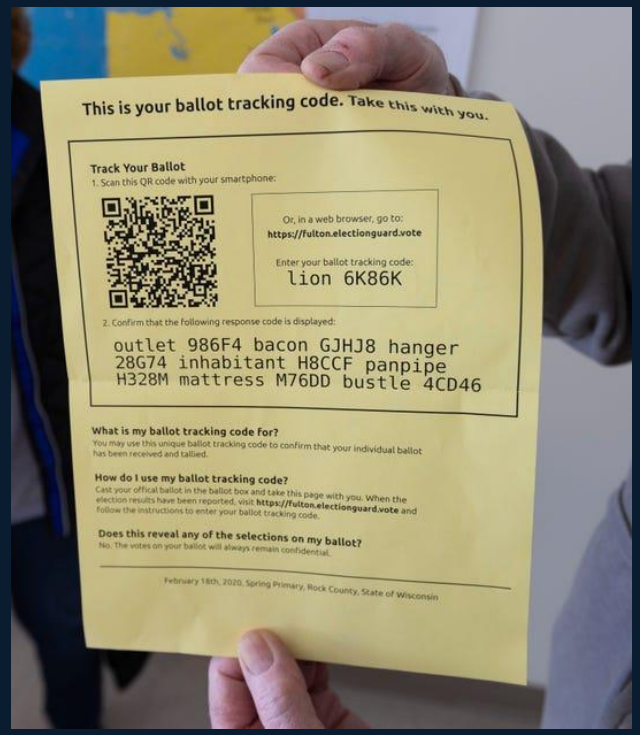
A proof is generated for each contest that ensures:

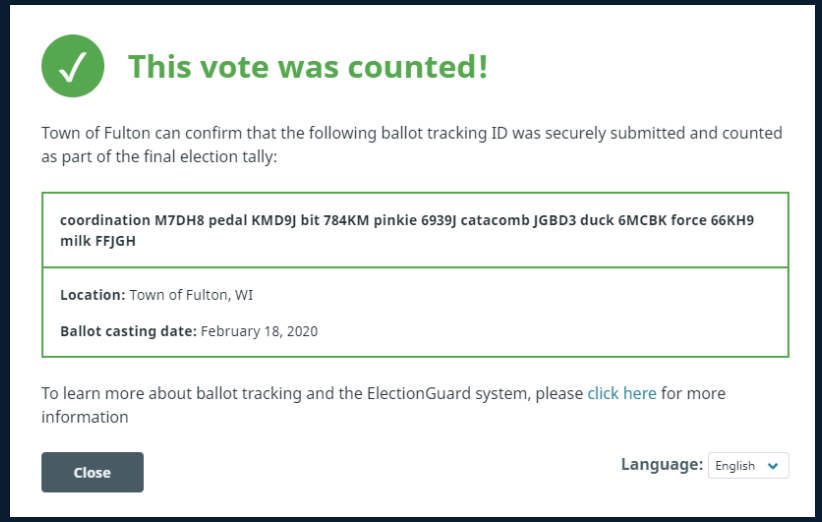
- A "1" (a single vote) or "0" (no selection) was entered for each selection
- A voter made no more than the maximum number of allowed selections

Independent Verifiers

Microsoft

When results are published, anyone can review the encrypted data, verify the proofs for the entire election, and determine if the data is mathematically correct and was not tampered with



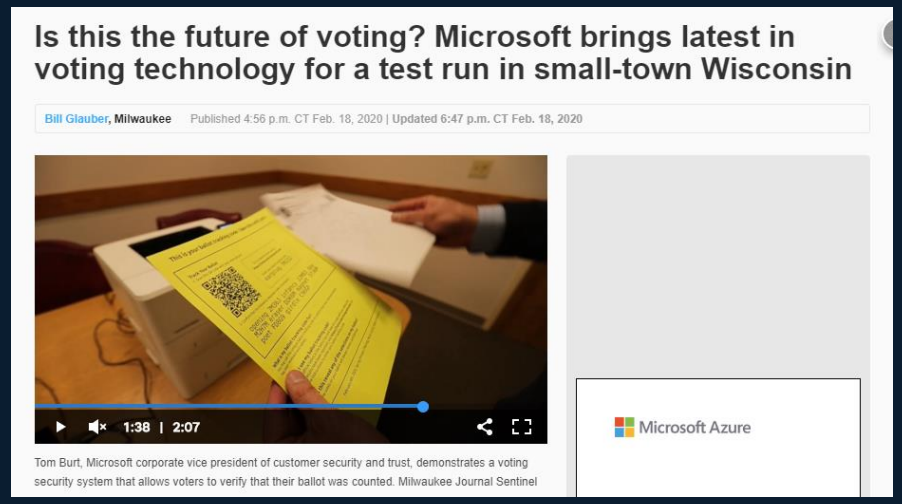


Individuals can check their unique Tracking ID (a hash representation of their encrypted vote) to ensure their vote is included in the final tally once and only once, without revealing their selections

Pilot Election in Fulton, WI

Microsoft to deploy ElectionGuard voting software in first real-world test

Residents in Fulton, Wisconsin will elect representatives for the Wisconsin Supreme Court via voting machines running Microsoft's ElectionGuard voting software.



VotingWorks

- Successful Pilot in February Primary election in Fulton, Wisonsin
- In partnership with VotingWorks, a nonprofit elections system vendor
- ElectionGuard acted as a parallel backup tally system to a hand count

You Can Help Validate ElectionGuard's Security!

…but we suggest kindly waiting until next week

Bug Bounty Program

- Security Researchers welcome!
- $100 - $15,000 bounties paid out for security and cryptography bugs
- Learn more at **aka.ms/EGbugbounty**

Open Source

- All on Github – **https://github.com/Microsoft/electionguard**
- Core cryptography components & mathematical specs
- Demo reference implementations

GA Release Coming Soon

- June 15th Release of full Python reference
- ElectionGuard Core (C++) coming later this summer for low-powered hardware
- Interested in early access? **Sign up for notices at aka.ms/EGnotify**

Thank you!

github.com/microsoft/electionguard

Questions?
Email: ElectionGuard@Microsoft.com

Matt Wilhelm – InfernoRed Technology - @addressXception
Ethan Chumley – Microsoft - @EthanChumley1