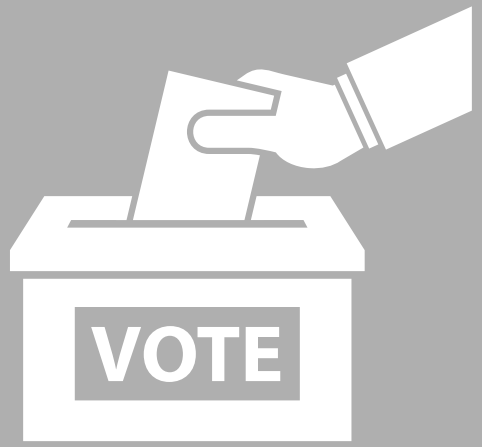
Vote of No Confidence: Second Factor Correctness

November 2018

DARKMATTER

SMART AND SAFE DIGITAL

Content

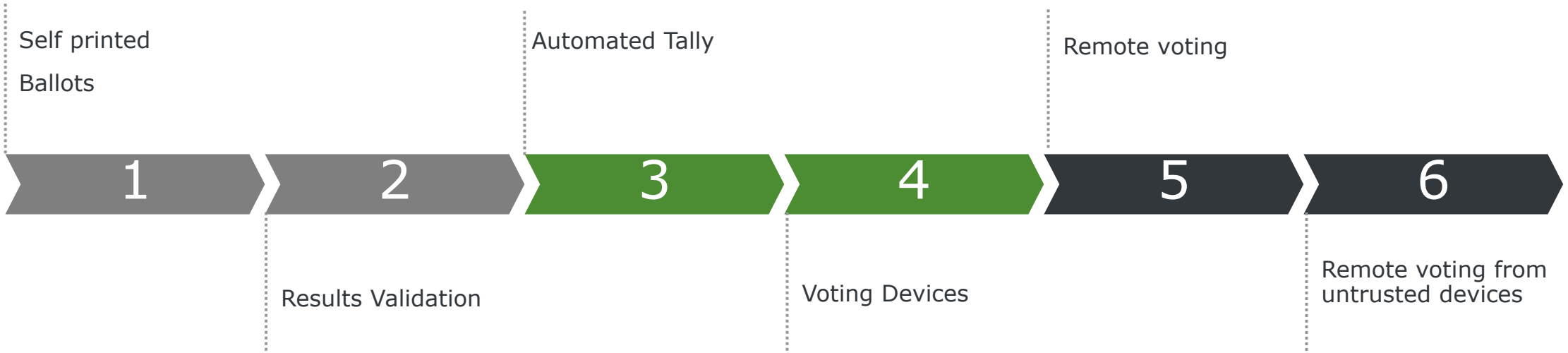


INTRODUCTION TO ELECTRONIC VOTING

INTRODUCTION TO ELECTRONIC VOTING

- Electronic voting could be summarized as a protocol allowing people to vote using **information technologies**.

- There are **many** proposals:



INTRODUCTION TO ELECTRONIC VOTING

- There are **3** main **security requirements** in electronic voting:

Integrity — Election results cannot be modified

Privacy — Ballots cannot be linked to their caster

Verifiability — All the process can be verified by an external authority.

INTRODUCTION TO ELECTRONIC VOTING

- These security requirements are more challenging as the technology **covers more** stages of an **election**:
  - For ballot printing **privacy** and **verifiability** remain the same.
  - With voting devices **all** properties are **required**, but the **environment** is more **controlled**.
    - Do not google Brazilian or US voting machines.
  - **Remote** electronic voting is the **most challenging** approach.

INTRODUCTION TO ELECTRONIC VOTING

Which are these challenges?

Problems arise from dealing with two confronted terms:

- **Privacy** vs **integrity**
  - Electoral roll against anonymity
- **Privacy** vs **verifiability**
  - Guarantee proper behavior without leakages
- **Security** vs **performance**
  - This one is a very old friend.
- **Integrity** vs **usability**
  - Make cryptography easy

INTRODUCTION TO ELECTRONIC VOTING

The 4 **phases** of a remote electronic voting protocol:

1 Setup

- Electoral roll
- Candidates
- Public key infrastructure
- Cryptosystem setup
- Publishes all information

2 Voting phase

- Voter choses a candidate
- Generates a ballot:
  - **Encryption**
  - **Zero** knowledge **proofs**
  - Additional information
- Digitally **signs** the ballot
- Sends ballot signed to server
- Server validates and stores

3 Tallying

The server:

- Validates consistency of DB
- **Anonymize** ballots
- Generates **correctness** proofs
- **Decrypts** all ballots
- Obtain the election results

4 Verification

Auditors (everyone):

- Validate electoral roll
- Validate ballot proofs
- Validate anonymization proofs
- Validate election results

PRIVACY vs INTEGRITY

- **Digital signatures** allow control over electoral roll.
- **Encryption** prevents voting options to be disclosed.
- We need to **anonymize encryptions**: (homomorphic operation)

MIXING

- Chose a random **permutation**
- Chose random **masking** values for each ballot
- Permute the ballots **applying** the masking values.

HOMOMORPHIC TALLYING

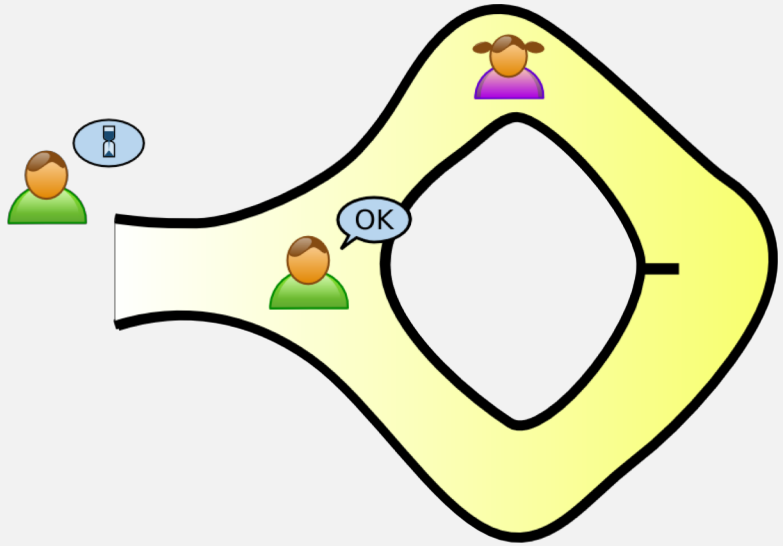
- **Encrypt** in a **specific** way considering the homomorphic operation.
- Use the homomorphic operation with **all** the ballots.
- Decrypt the **resulting** ciphertext.
- **Recover** the messages from the result.

PRIVACY vs VERIFIABILITY

- Data is **anonymized**, but how can correctness be **guaranteed**?

ZERO KNOWLEDGE PROOFS

- A protocol by which a prover can **guarantee** a statement to be **true** without giving information about the a **secret** parameter:



PRIVACY vs VERIFIABILITY

Where are these ZKP's used?

- **Decryption:** Proof to be using the appropriate private key
  - Secret parameter: The key
- **Encryption:** Proof to the plaintext to be a candidate
  - Secret parameter: The candidate
- **Mixing:** Proof a permutation and masking to be properly applied
  - Secret parameter: The permutation and masking values

SECURITY vs PERFORMANCE

- These are the typical requirements:
  - **Require** public key cryptosystems
  - Proofs **require** trapdoor functions
  - **Require** PKI for digital signatures
  - Voters should be able to vote from mobile devices. (**Efficient**)
  - **Minimize traffic** in the network

- Widely known solution:
  - **There is no solution: Do the best with what we have!!!**

INTEGRITY vs USABILITY

- Integrity requires the use of:
  - **Cryptography**
  - Identification protocols
  - **Zero knowledge proofs**

- Voters must use it without knowing how to
  - **All** cryptographic operations should be **transparent** for the voter
  - The protocol must be **easy** to **identify** as a traditional voting protocol

- Solution:
  - Voting device **computes** cryptography, and voter only selects a candidate

INTEGRITY REQUIREMENTS

- This approach requires trust in two components:

SOFTWARE IMPLEMENTATION

- The binaries have not been modified
- The application behaves properly

VOTING DEVICE

- The device is not controlled by an attacker
- The device behaves properly

- The required assumptions are very **hard** to meet in **real world**
- Solution:
  - **Do not trust** anyone!!

Vote with No Confidence

VERIFIABILITY AGAINST UNTRUSTED DEVICES

Three different kinds of **verifiability**

1

CAST AS INTENDED

- Verify the **voting option** is the intended one.
- Verify the device **casts** the ballot

2

RECORDED AS CAST

- Voter is in the electoral **roll**
- Voter has **not cast** a ballot before
- Vote is digitally **signed** by the voter
- **Stored** the ballot with the digital signature in a publicly accessible **bulletin board**

3

COUNTED AS RECORDED

- Ballots **processed** properly
- **Decryption** with the proper private key

CAST-AS-INTENDED VERIFIABILITY

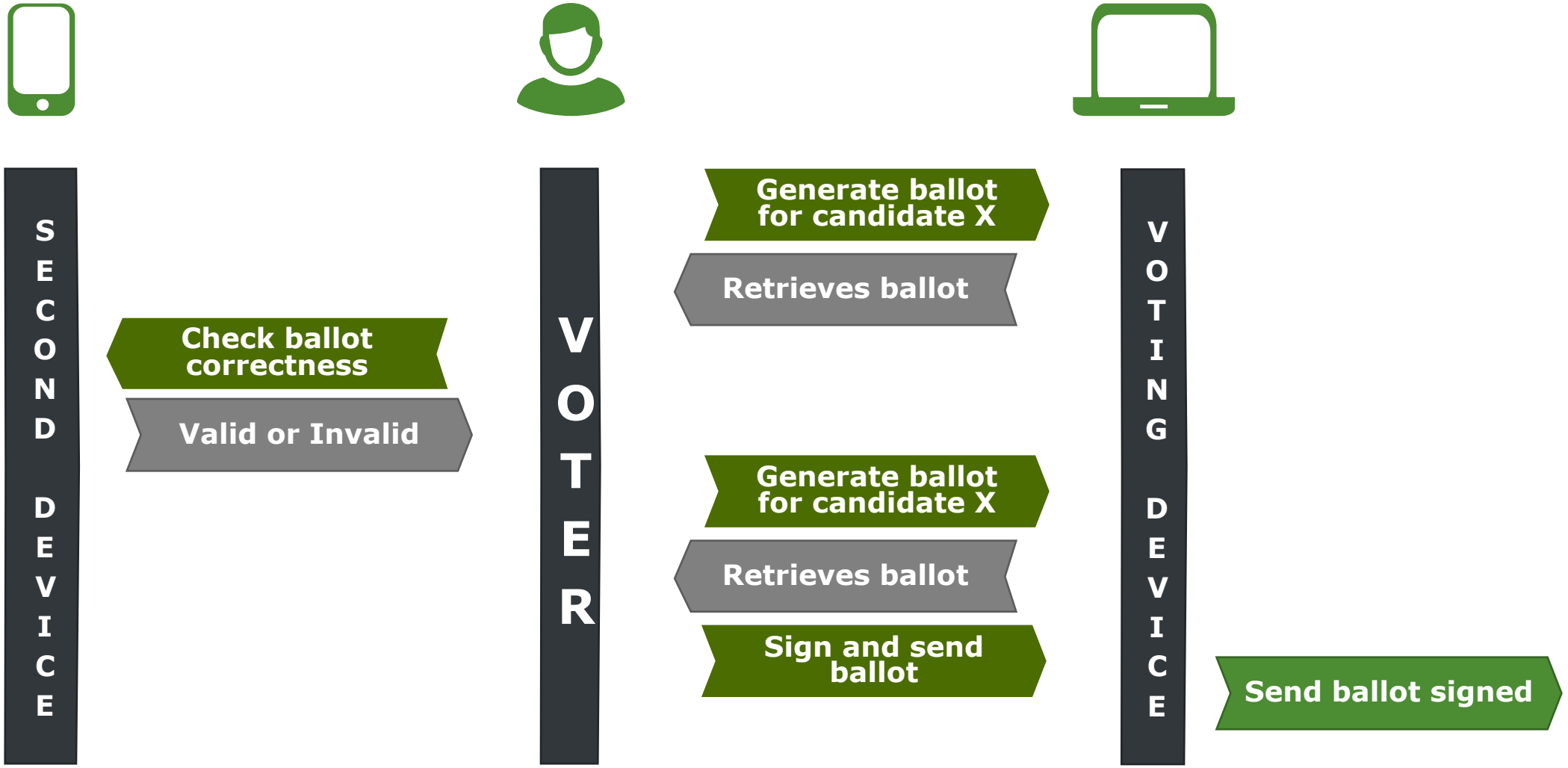
CHALLENGE THE SYSTEM

- The voter **asks** to generate a ballot
- The voter **decides** either to send the ballot or to verify its content
- A **second device** validates the content
- **Repeat** this until the device is trusted
- Then, **send** the vote

PROOF OF CORRECTNESS

- The voter **asks** to generate a ballot
- The voter **asks** for a ZKP for this ballot
- The ballot is **sent**
- The server **validates** the proof and returns a **code**
- Voter knows the **code** is **related** with her voting **option**
- Voter **sends** a validation **code** to **confirm** correctness

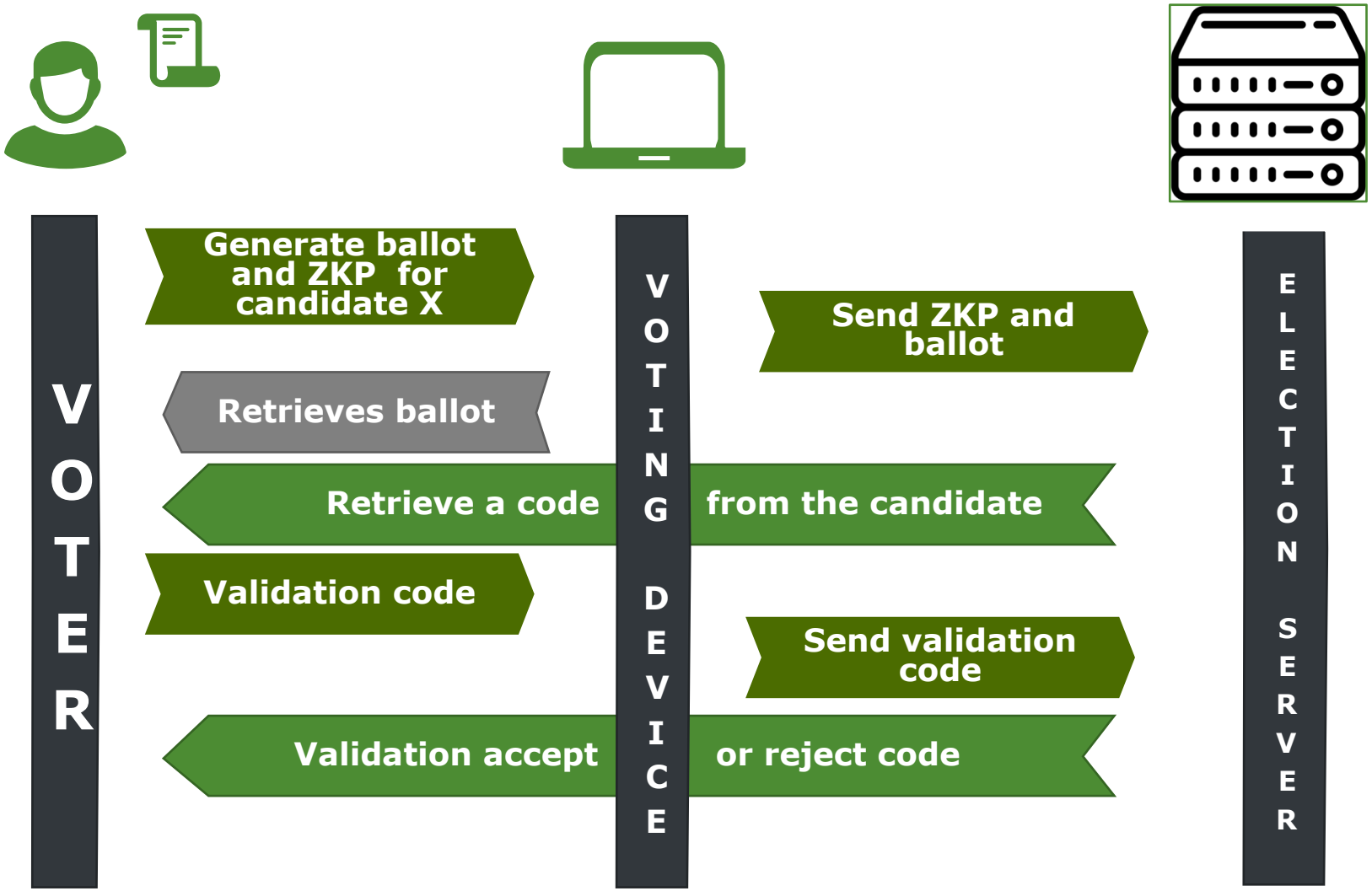
CHALLENGE THE SYSTEM



CHALLENGE THE SYSTEM

ADVANTAGES 	DISADVANTAGES 
Cheating device do **not know** if it is challenged	**Cheating probability** rely on the amount of validations from the **voter**
The validation is sound (**no false positives**)	Requires a **second device** with knowledge of the protocol
A **voter** can generate **trust** on her device	**They do not care** (not easy to use)
	Voter **never checks** the ballot sent
	Social engineering attack might succeed (it is voluntary)

PROOF OF CORRECTNESS



PROOF OF CORRECTNESS

ADVANTAGES	DISADVANTAGES
Easy to validate by **voters**	**Probability** of cheating **increased** (but low)
Part of it might be validated by **auditors**	Entire **proof** can **only** be validated by **voter**
Encourage voter participation	They **do not care**
Proof apply to the **ballot** sent	
Only **voting device** needed (or not)	

Security
Details

VERY SPECIFIC CASE

- We use a protocol to achieve:
  - **Privacy**
  - **Correctness**
  - **Verifiability**

- **Elections** are a very **controversial** topic

- Are there other **transactions** where a device cannot be **trusted**?
  - Fortunately (for us) the response is: **YES!**

OTHER CASES

- Authentication:
  - It is the **first** security **measure**
  - Also faces the restriction of **security** vs **usability**
    - **Short** passwords
    - Many **recovery** mechanisms
  - **Historically** has been **broken** too many times
    - Companies using **bad practices**
    - Users using **bad passwords**

SECOND FACTOR AUTHENTICATION

- The solution proposed is the Second Factor Authentication (SFA):

ADVANTAGES 	DISADVANTAGES 
Prevents **unauthorized** access	**Weak** against corrupted devices
Easy to use (sometimes)	
Offers sort of a physical **protection**	

SECOND FACTOR AUTHENTICATION

- Why is authentication important?
  - To prevent an attacker from impersonating a user

- Does SFA prevents that from happening?
  - It certainly **prevents** authentication when **user** is **not using** the application

- What happens when the user is already using the application?
  - The **attacker** successes on impersonating:
    - Signing, Messaging, Online Banking, Play games,…

SECOND FACTOR AUTHENTICATION

- Used **properly**, SFA can prevent **unintended** authentications

- But once authenticated, the common tools to **prevent attacks** are:
  - Receiving a **confirmation** message for specific **actions** (CODE)
    - Vague (No information about the action)
    - Allows forgery
  - Receiving a very **precise confirmation** message (CODE)
    - Hardly private (contains the amount)
    - Allows forgery

LOOKING FOR A BETTER SOLUTION

- A better solution should:
  - **Prevent** any **unwanted** action to happen
  - Maintain the **privacy** of the action
  - Allow **verification** by user
  - Be **usable** (at least as SFA)

- These properties are the **same** than in **remote** electronic **voting**
  - Can the **proof of correctness** approach be **generalized**?

Second Factor Correctness

INITIAL IDEA

Assuming a case in which **privacy** and **correctness** are **not** required:

- The **information** of the transaction is **sent** to the user
- The user **accepts** the transaction
- The server **proceeds** with the transaction

For **privacy** we could just encrypt the content:

- Requires dealing with **keys** (symmetric apparently)
- The key is still on the device (needs to be trusted)

CORRUPTED DEVICE

- What happens if the **device**, or the software used, is **corrupt**?
  - **PRIVACY**:
    - The device **knows** all the transaction information
    - Fixing this has a big **impact** in **usability**
  - **INTEGRITY**:
    - The **information** shown to the user might be **fake**:
      - The device modifies the **petition** and the **response**
      - The **user** still proceeds by **confirming** the operation
- Additional device is needed to prevent integrity disruptions

ACTORS IN SECOND FACTOR CORRECTNESS

- **Entities** and **devices** involved in the protocol are:
  - User (U) **willing** to perform an **action**
  - Entity (E) **offering** the action as a **service** in the Internet
  - We have the **original** device running the **application** (the corrupted device CD)
  - There is also an **additional** device for **validating** purposes (the validation device VD)

VALIDATION DEVICE

- Requirements for VD:
  - **Non** accessible by **network** connection
  - Able to read **QR** codes
  - **Cryptography** capable (**symmetric** cryptography and **hash** functions)
  - An **output** channel (screen or something)
  - Enough **storage** to store a symmetric **key**

PROTOCOL OVERVIEW

Petition submission phase:

- U uses CD to **send** a petition to E servers
- E servers store the petition and put it on **hold**
- E servers derive a **short-lived** key using the original **key** and a **timestamp**
- E **encrypts** a summary of the transaction information + timestamp and sends it back to CD
- E computes the **validation code** using the short-lived key, the generated ciphertext, and the timestamp
- E computes the **confirmation code** using the short-lived key, the generated ciphertext, and the timestamp
- CD **presents** this information as a **QR code**

PROTOCOL OVERVIEW

Petition validation phase:

- U uses VD to **scan** the code and obtains a timestamp and an encrypted ciphertext
- VD **validates** the timestamp to be close to the actual time
- VD **derives** a short-lived key using the timestamp and its secret key
- VD computes a **validation code** using the ciphertext, the short-lived key and the timestamp
- VD computes a **confirmation code** using the ciphertext, the short-lived key and the timestamp
- VD **decrypts** the ciphertext and **shows** the information to U
- VD shows the **validation code** to U

PROTOCOL OVERVIEW

Confirmation phase:

- If the information shown by the VD is **not correct**, U does **nothing**
- Otherwise, U inputs the **validation code** to the CD
- CD **sends** the validation **code** received to E servers
- If the validation code received matches the previously generated one:
  - The operation is valid and E can **proceed** with it
  - E also sends **confirmation** code to the CD
  - CD shows the confirmation code to U
- Otherwise, the CD is assumed to be misbehaving U must contact E using another device

Use Cases

ONLINE BANKING

- **Transactions** with banks require both:
  - Correctness
  - Privacy
  - Verifiability

- **Users** accept **extra security** measures when dealing with **money**

- Banks are already providing **similar** solutions
  - Usually involving **SMS** with codes
  - Also phone **calls**
  - Security numbers from a **second device**

ALREADY EXISTING SOLUTIONS

Mobile Banking already have some solutions:

SMS + CODE

Please use PIN Number 548914 to complete your transaction for AED 681.45 with [redacted] card ending [redacted] If you did not request a PIN please call us. T&Cs apply

RANDOM CODE

ONLINE BANKING

- The proposal consists on a second device (SFC device):
  - With a secret **seed** (key) known by the bank
  - With a small **display**
  - Able to scan **QR** codes
  - Able to run **Key Derivation** Functions (KDF)
  - Able to run **symmetric key** cryptography
  - Able to run SHA3 hash function.

ONLINE BANKING

- Security **relies** on the SFC **device**

- To prevent attacks:
  - It is more **conservative** to not allow internet connection
  - Capabilities must be very **restricted**
  - **QR** scanning is the main **threat** vector

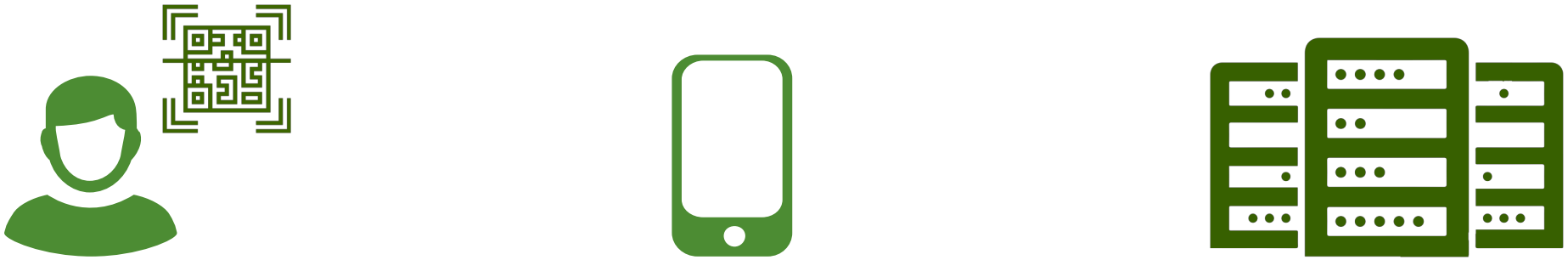
- **Physical** attacks can be **prevented** in different ways but increasing the **cost** of the device.
  - The **seed** could be protected
  - The rest is **not** as **critical** as the seed

QR CODE GENERATION

QR CODE GENERATION

- When the user performs an action:
  - The bank **process** the petition
  - **Derives** a short-lived key (s) = KDF(seed, timestamp (ts))
  - Generates a QR code with the **encryption (e)**, using the users **symmetric** key (s), of the following information:
    - **Type** of transaction (t)
    - **Date** of petition (d)
    - **Amount** involved (a)
    - Additional **data** (x)
  - Generates the validation **code** (c) = encode(SHA3(t,d,a,x,s,"validate"))
  - Generates the confirmation **code** (c) = encode(SHA3(t,d,a,x,s,"confirm"))
  - Sends the QR code containing (ts) and (e) to the application

SFC DEVICE PROTOCOL



SFC DEVICE PROTOCOL

- The user validates by:
  - **Scanning** the QR code with the SFC device
  - The SFC checks if the **(ts)** is in an appropriate time **frame** considering **current** time
  - The SFC derives the short-lived key s = KDF(seed,ts).
  - The SFC **decrypts** (e), using (s), and gets:
    - Type (t), date(d), amount(a), and additional data (x)
  - The device generates the validation **code** as Encode(SHA3(t,d,a,x,s,”validate”))
  - The device generates the confirmation **code** as Encode(SHA3(t,d,a,x,s,”confirm”))
  - The device shows the **information** of the transaction and both **codes**.
  - The user **inputs** the **validation** code to the application

SFC CONFIRMATION

- Once the bank receives the **validation** code:
  - Validates the received code is the same as the **previously computed** one
  - Sends the **confirmation code** to the client's device
  - Waits an **acknowledged** amount of time
  - **Proceeds** with the transaction.

  - If the validation code was **not correct**, it would mark the **transaction** as **invalid**

IDEALISTIC ALTERNATIVE USING ZKP

- The **requirements** in the communication does **not** seem to require a **ZKP**
- However, they **can** be **used** for:
  - Allowing **verification** from an **external** entity without **leaking** operational information

- This advantage does not seem important for the purpose, but the disadvantages:
  - ZKP are **slower**
  - ZKP require additional **arithmetic** in the device
  - The device would require more **storage** and **memory**
  - Basically….. It would make the SFC more **EXPENSIVE**.

Final Remarks

DEVIL IS IN THE DETAILS

- The **solution** proposed is **not** unbreakable
- It **drastically** reduces the success probabilities from an **attacker** with control of the user's device
- Unfortunately, this is not the only **attack vector:**
  - Getting access to the **entity's server** would still allow an attacker to cheat, using **validation** devices **keys**
  - **Physical access** to the validation **device** could also be enough to obtain information about the **key**
- It is important to **know** the **limitations** of the solution so that proper **mitigation** techniques can be implemented

FUTURE WORK

- **MAKE IT REAL:** Theory is always beautiful, but its time to **get** the hands **dirty**

- **Key rotation** on the device
  - Keeping always the same **seed** in the device is **dangerous**
  - Creating a mechanism to change the key using **QR** reader seems **risky**

- **QR** code readers are too **old-fashioned**
  - Why not something like google **glasses**?
  - Does secure **watch** sound as cool as smart watch?

THANK YOU!