How to Measure Usable Security: Natural Strategies in Voting Protocols

Damian Kurpiewski, Mateusz Kamiński
(Wojtek Jamroga, Vadim Malvone)

FOLCO 03.12.2024

Motivation

Analyzing voting protocols

Voting / e-voting protocol

- Cryptography
- Procedures
- Attacker/intruder
- Security
- Human factor

Human factor

Makes a mistake

Ignores instructions

Skips the procedure, because it’s too complex, time-consuming, hard to understand…

Can affect the security of the system

Analyzing the voting system

01

Create the (simplified) model of the system

02

Focus on the voter's behavior and her point of view

03

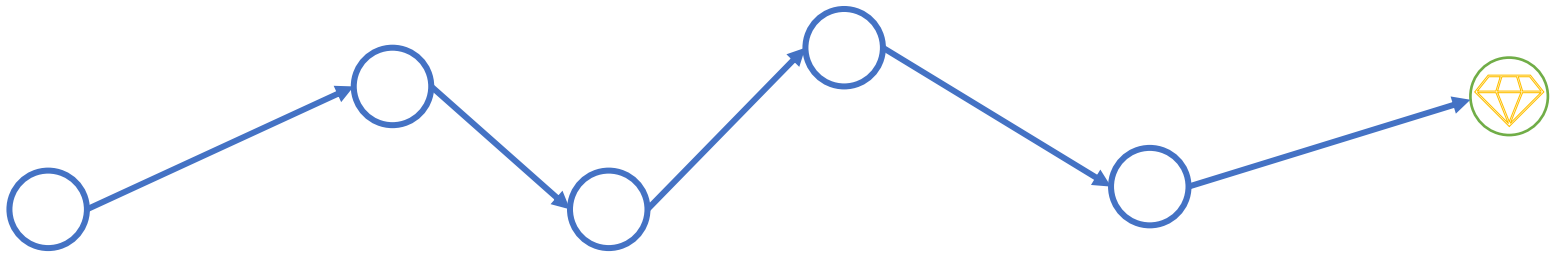
Describe requirements using ATL/NatATL formulae

04

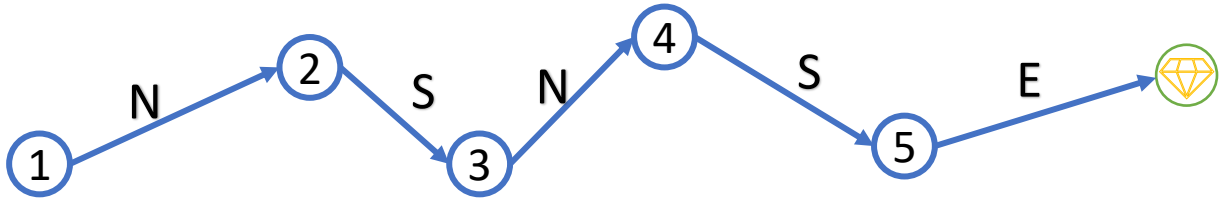
Create natural strategies for the voter (and other agents)

Strategy

- A plan
- A path in the model



Strategy description



1: N

2: S

3: N

4: S

5: E

Classic strategy

Complex

Long

Easy for the computer

Hard for the human

Natural Strategy

Conditional plan

Decisions are based on some observations

Based on the human behavior

Natural Strategy for the Voter

1. Out of the polling station -> go to the polling station

2. Empty ballot -> fill your ballot

3. Filled ballot -> cast your vote

Strategy in reality

- Understand the rules of the voting procedure
- Check if your vote is correct
- Verify that your vote has been counted correctly
- Sign-in to the e-voting system
- And much more …

Background

Logics and strategies

ATL: What Agents Can Achieve

- ATL: Alternating-time Temporal Logic [Alur et al. 1997-2002]
- Temporal logic meets game theory
- Main idea: cooperation modalities
- $\langle\langle A \rangle\rangle \phi$: **coalition A has a collective strategy to enforce ϕ**
- ϕ can include temporal operators: X (next), F (sometime in the future), G (always in the future), U (strong until)

Example Formula

- $\langle\langle Client \rangle\rangle F\ ticket$
- Client can eventually buy a ticket

Strategy

A *strategy* of agent $a \in \mathbb{A}\mathrm{gt}$ is a conditional plan that specifies what a is going to do in every possible situation.

Formally, a perfect information memoryless strategy for a can be represented by a function $s_a : St \to Act$ satisfying $s_a(q) \in d_a(q)$ for each $q \in St$.

Strategy

A *strategy* of agent $a \in \mathbb{A}\mathrm{gt}$ is a conditional plan that specifies what a is going to do in every possible situation.

Formally, a perfect information memoryless strategy for a can be represented by a function $s_a : St \to Act$ satisfying $s_a(q) \in d_a(q)$ for each $q \in St$.

An *imperfect information memoryless strategy* additionally satisfies $s_a(q) = s_a(q')$ whenever $q \sim_a q'$

Natural ATL

- Strategies in a form of a set of simple conditions: guarded actions
- Strategy complexity represented as the total lengths of guards in the strategy
- $\langle\langle A \rangle\rangle^{\leq k} \phi$: **coalition A has a collective strategy of size less or equal than k to enforce ϕ**
- $\langle\langle Client \rangle\rangle^{\leq 10} F\ ticket$
- Client can buy a ticket by a strategy of complexity at most 10

Example Strategy

1. $\neg ticket \wedge \neg selected \wedge \neg paid \wedge \neg error \rightarrow \boldsymbol{select}$
2. $selected \rightarrow \boldsymbol{pay}$
3. $\top \rightarrow \boldsymbol{idle}$

Example Strategy Complexity

1. $\neg ticket \wedge \neg selected \wedge \neg paid \wedge \neg error \rightarrow \boldsymbol{select}$

$$\boldsymbol{cost = 11}$$

2. $selected \rightarrow \boldsymbol{pay}$

$$\boldsymbol{cost = 1}$$

3. $\top \rightarrow \boldsymbol{idle}$

$$\boldsymbol{cost = 1}$$

Complexity: $\mathbf{11 + 1 + 1 = 13}$

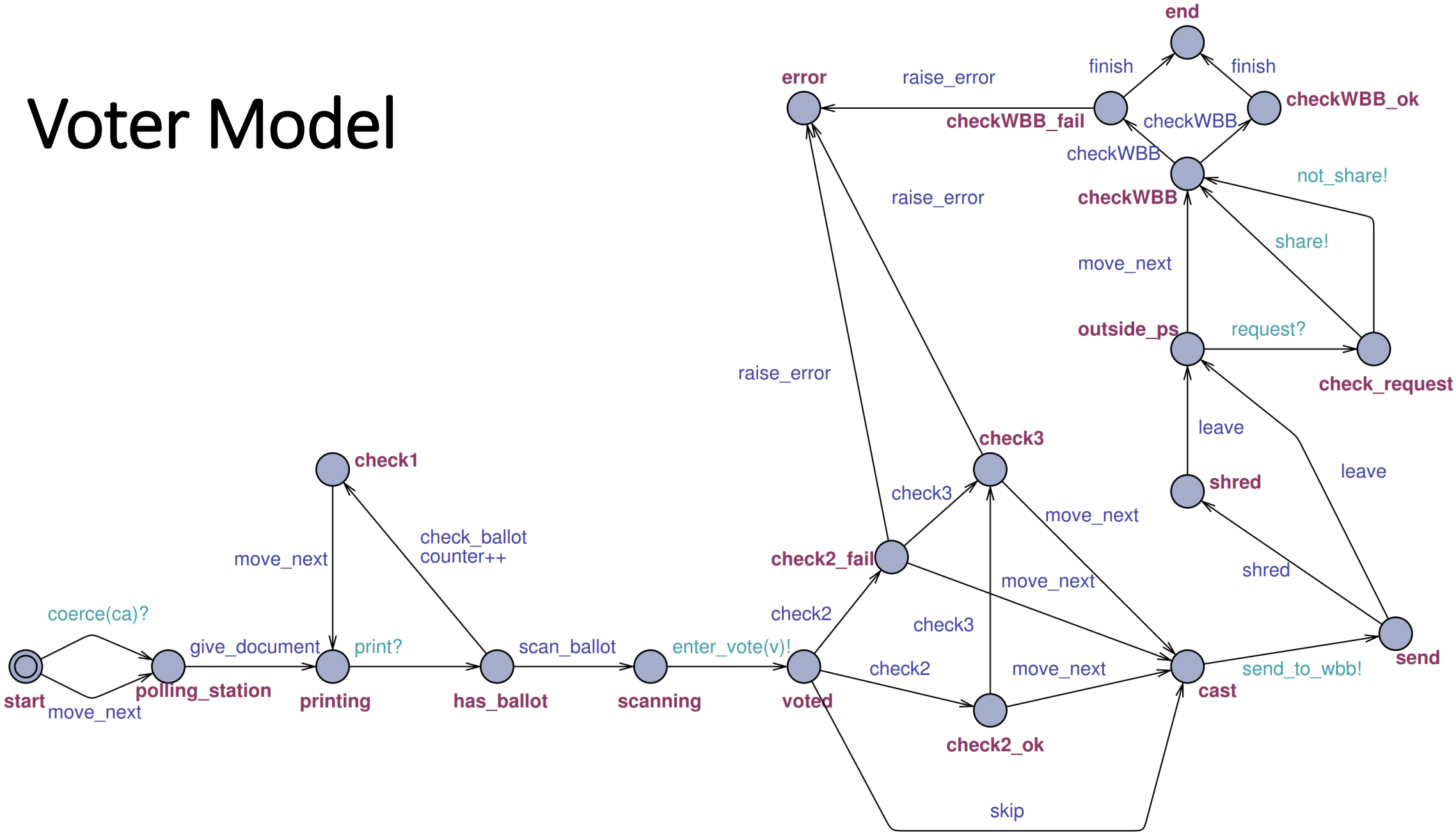
Case Study

vVote voting system

Example case study: vVote

- Implementation of $Prêt\ á\ Voter$ protocol
- Used for remote voting and voting of handicapped persons in the Australian state of Victoria elections in November 2014
- **Main idea**: encoding the vote using a randomized candidate list

Voter Model


start
coerce(ca)?
move_next
polling_station
give_document
printing
print?
move_next
check1
check_ballot
counter++
has_ballot
scan_ballot
scanning
enter_vote(v)!
voted
error
raise_error
checkWBB_fail
finish
end
finish
checkWBB_ok
checkWBB
checkWBB
checkWBB
not_share!
raise_error
share!
move_next
outside_ps
request?
check_request
raise_error
leave
shred
leave
check3
check3
move_next
check2_fail
shred
move_next
check2
check3
check2
move_next
send_to_wbb!
send
cast
check2_ok
skip

$$\varphi_1 = \langle\langle voter \rangle\rangle^{\leqslant k} \mathrm{F}(\mathsf{checkWBB\_ok} \vee \mathsf{checkWBB\_fail})$$

(1) $\mathsf{start} \vee \mathsf{check2\_ok} \vee \mathsf{check2\_fail} \vee \mathsf{outside\_ps} \rightsquigarrow move\_next$
(2) $\mathsf{polling\_station} \rightsquigarrow give\_document$
(3) $\mathsf{has\_ballot} \rightsquigarrow scan\_ballot$
(4) $\mathsf{scanning} \rightsquigarrow enter\_vote(v)$
(5) $\mathsf{voted} \rightsquigarrow check2$
(6) $\mathsf{cast} \rightsquigarrow send\_to\_wbb$
(7) $\mathsf{send} \rightsquigarrow shred$
(8) $\mathsf{shred} \rightsquigarrow leave$
(9) $\mathsf{check\_request} \rightsquigarrow not\_share$
(10) $\mathsf{checkWBB} \rightsquigarrow checkWBB$
(11) $\top \rightsquigarrow \star$

Complexity

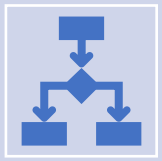
- 11 guarded commands
- (1) start ∨ $check2\_ok$ ∨ $check2\_fail$ ∨ $outside\_ps$: cost 7
- Other guarded commands cost 1
- Total complexity: $1 * 10 + 7 * 1 = \mathbf{17}$
- The formula φ_1 is true with any k of 17 and more

Example construction of the strategy for φ_1

(1) has_ballot $\rightsquigarrow$ *scan_ballot*
(2) $\neg$has_ballot $\wedge$ scanning $\rightsquigarrow$ *enter_vote*
(3) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ voted $\rightsquigarrow$ *check2*
(4) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ (check2_ok $\vee$ check2_fail) $\rightsquigarrow$ *move_next*
(5) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ $\neg$(check2_ok $\vee$ check2_fail) $\wedge$ cast $\rightsquigarrow$ *send_to_wbb*
(6) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ $\neg$(check2_ok $\vee$ check2_fail) $\wedge$ $\neg$cast $\wedge$ send $\rightsquigarrow$ *shred*
(7) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ $\neg$(check2_ok $\vee$ check2_fail) $\wedge$ $\neg$cast $\wedge$ $\neg$send $\wedge$ shred $\rightsquigarrow$ *leave*
(8) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ $\neg$(check2_ok $\vee$ check2_fail) $\wedge$ $\neg$cast $\wedge$ $\neg$send $\wedge$ $\neg$shred $\wedge$ check_request $\rightsquigarrow$ *not_share*
(9) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ $\neg$(check2_ok $\vee$ check2_fail) $\wedge$ $\neg$cast $\wedge$ $\neg$send $\wedge$ $\neg$shred $\wedge$ $\neg$check_request $\wedge$ checkWBB $\rightsquigarrow$ *checkWBB*
(10) $\neg$has_ballot $\wedge$ $\neg$scanning $\wedge$ $\neg$voted $\wedge$ $\neg$(check2_ok $\vee$ check2_fail) $\wedge$ $\neg$cast $\wedge$ $\neg$send $\wedge$ $\neg$shred $\wedge$ $\neg$check_request $\wedge$ $\neg$checkWBB $\rightsquigarrow$ $\star$

Problems

Problems to solve



Finding (one of possibly many) natural strategy for the given formulae (if the strategy exists)

Minimazing the representation/complexity of the found strategy

Problems to solve

Finding (one of possibly many) natural strategy for the given formulae (if the strategy exists)

Minimazing the representation/complexity of the found strategy

Strategy representation example 1

q1	q2	q3	q4	act
1	0	0	0	A
0	1	1	0	B
0	1	0	0	C

Strategy representation example 1

q1	q2	q3	q4	act
1	0	0	0	A
0	1	1	0	B
0	1	0	0	C

After reduction:

q1	q3	act
1		A
	1	B
		C

Strategy representation example 1

q1	q2	q3	q4	act
1	0	0	0	A
0	1	1	0	B
0	1	0	0	C

After reduction:

q1	q3	act
1		A
	1	B
		C

Natural strategy:

1. $q1 \rightarrow \boldsymbol{A}$
2. $q3 \rightarrow \boldsymbol{B}$
3. $\top \rightarrow \boldsymbol{C}$

Strategy representation example 2

q1	q2	q3	q4	act
1	0	0	0	A
0	1	0	1	A
1	1	0	0	B
0	1	1	0	B

Strategy representation example 2

q1	q2	q3	q4	act
1	0	0	0	A
0	1	0	1	A
1	1	0	0	B
0	1	1	0	B

After reduction:

q1	q2	q4	act
1	1		B
1			A
		1	A
			B

Strategy representation example 2

q1	q2	q3	q4	act
1	0	0	0	A
0	1	0	1	A
1	1	0	0	B
0	1	1	0	B

After reduction:

q1	q2	q4	act
1	1		B
1			A
		1	A
			B

Natural strategy:

1. $q1 \wedge q2 \rightarrow \boldsymbol{B}$
2. $q1 \vee q4 \rightarrow \boldsymbol{A}$
3. $\top \rightarrow \boldsymbol{B}$

Conclusions

- It’s not enough that a voter has a strategy – complexity is important
- Natural Strategy complexity helps to estimate the mental difficulty
- Other important factors exists: time, money, etc.
- Some parts of the voting procedure require more detailed models
- The presented methodology can be applied outside the e-voting domain

Thank You