Proving the security of blockchain protocols

Aggelos Kiayias
aggelos.kiayias@ed.ac.uk

Based on joint work with Juan Garay, Nikos Leonardos

Gratefully acknowledging research and curriculum development support

project CODAMODA

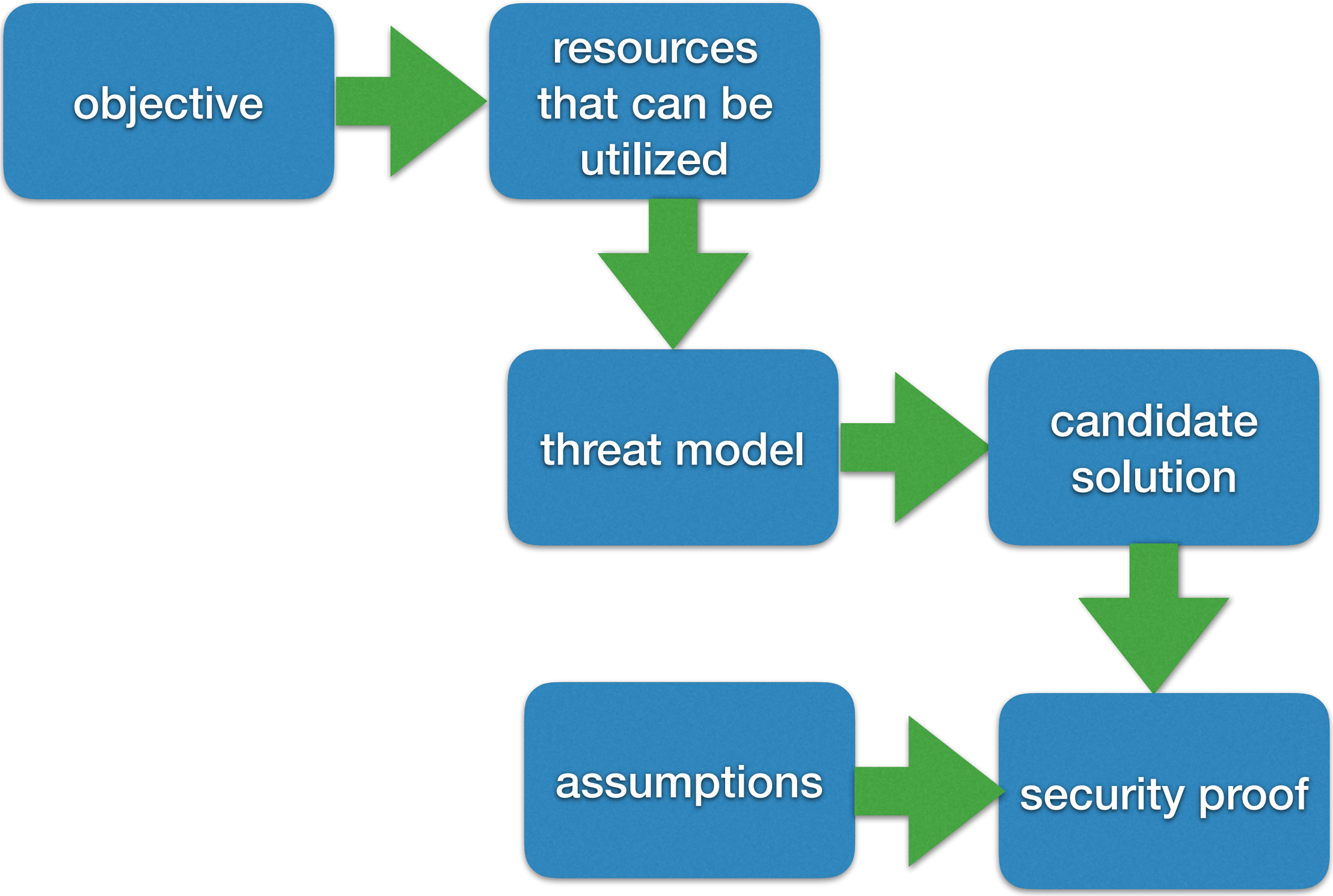
project PANORAMIX

for supporting lecture material development

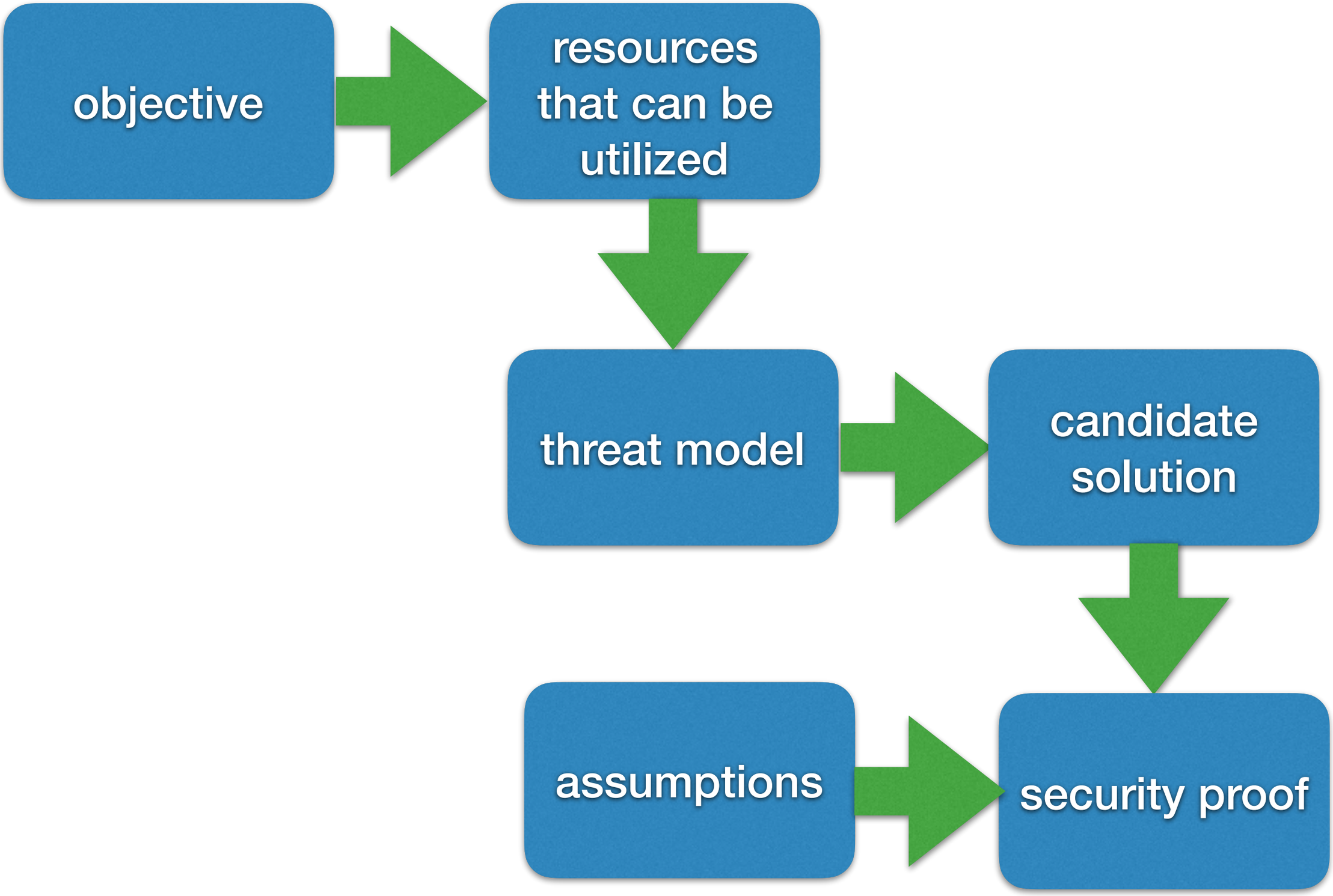
Foundations of Blockchain Protocols

- Understand the **fundamental security properties** of these protocols and obtain **proofs of security** in formal adversarial models.

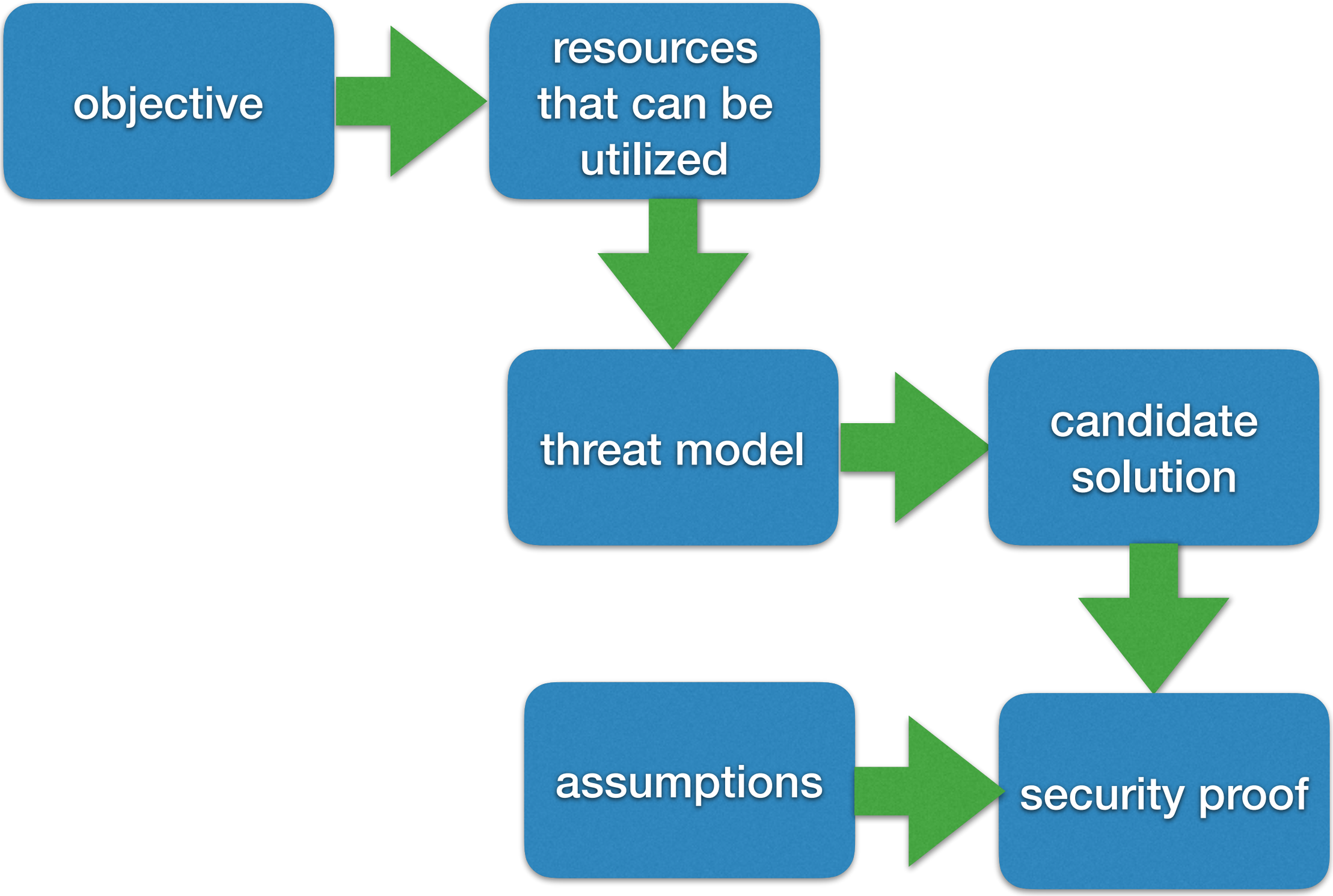
Building Secure Systems



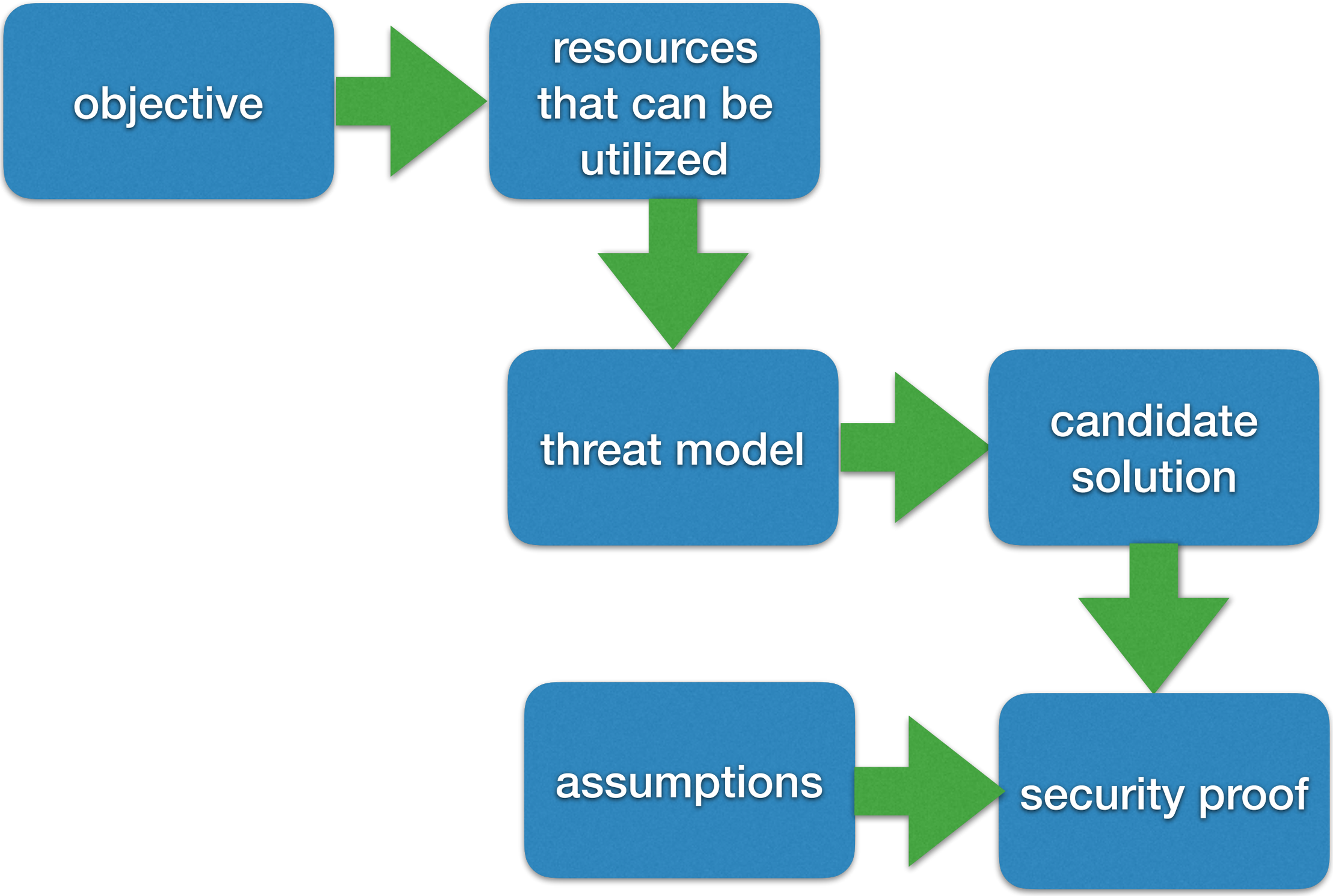
Building Secure Systems



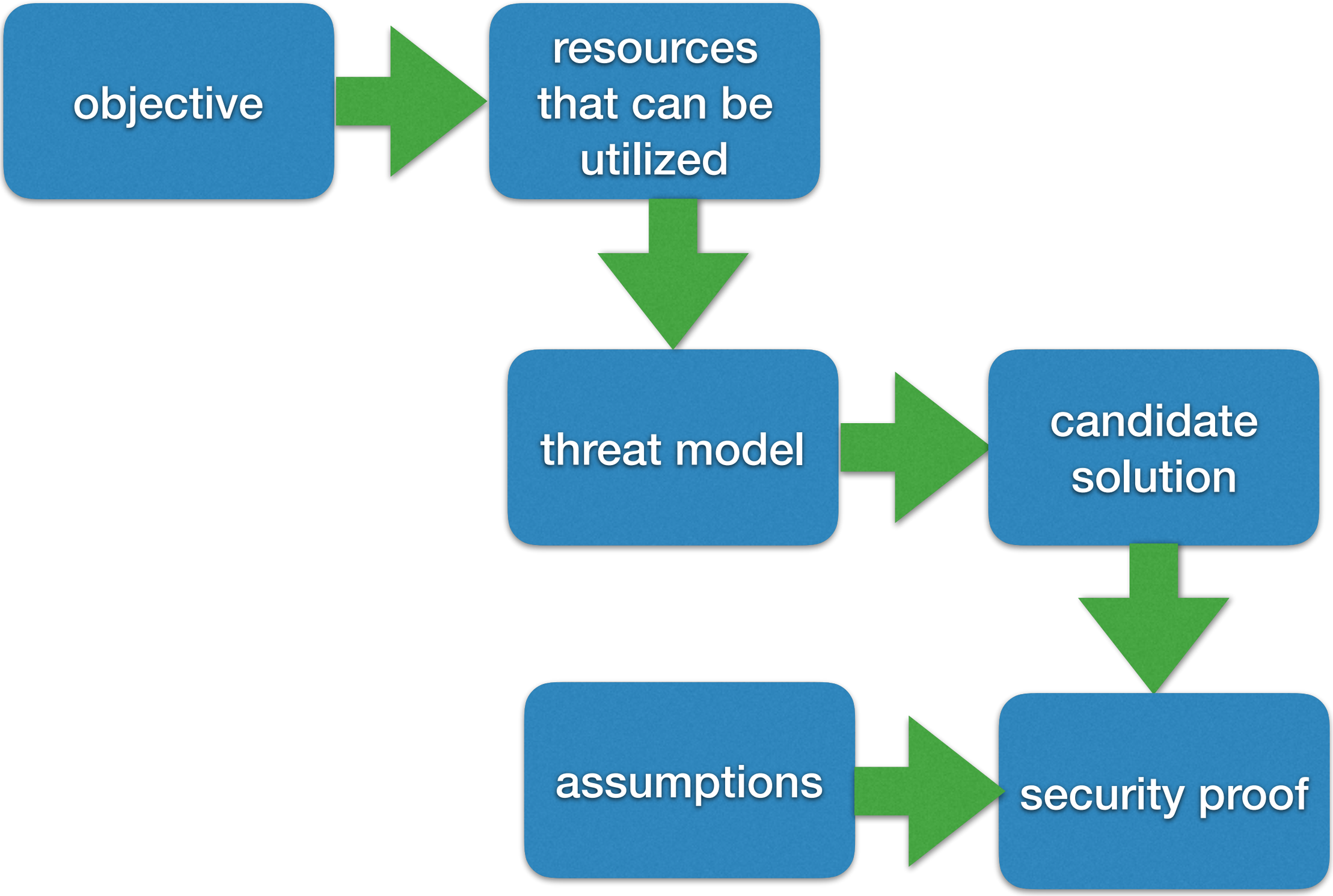
Building Secure Systems



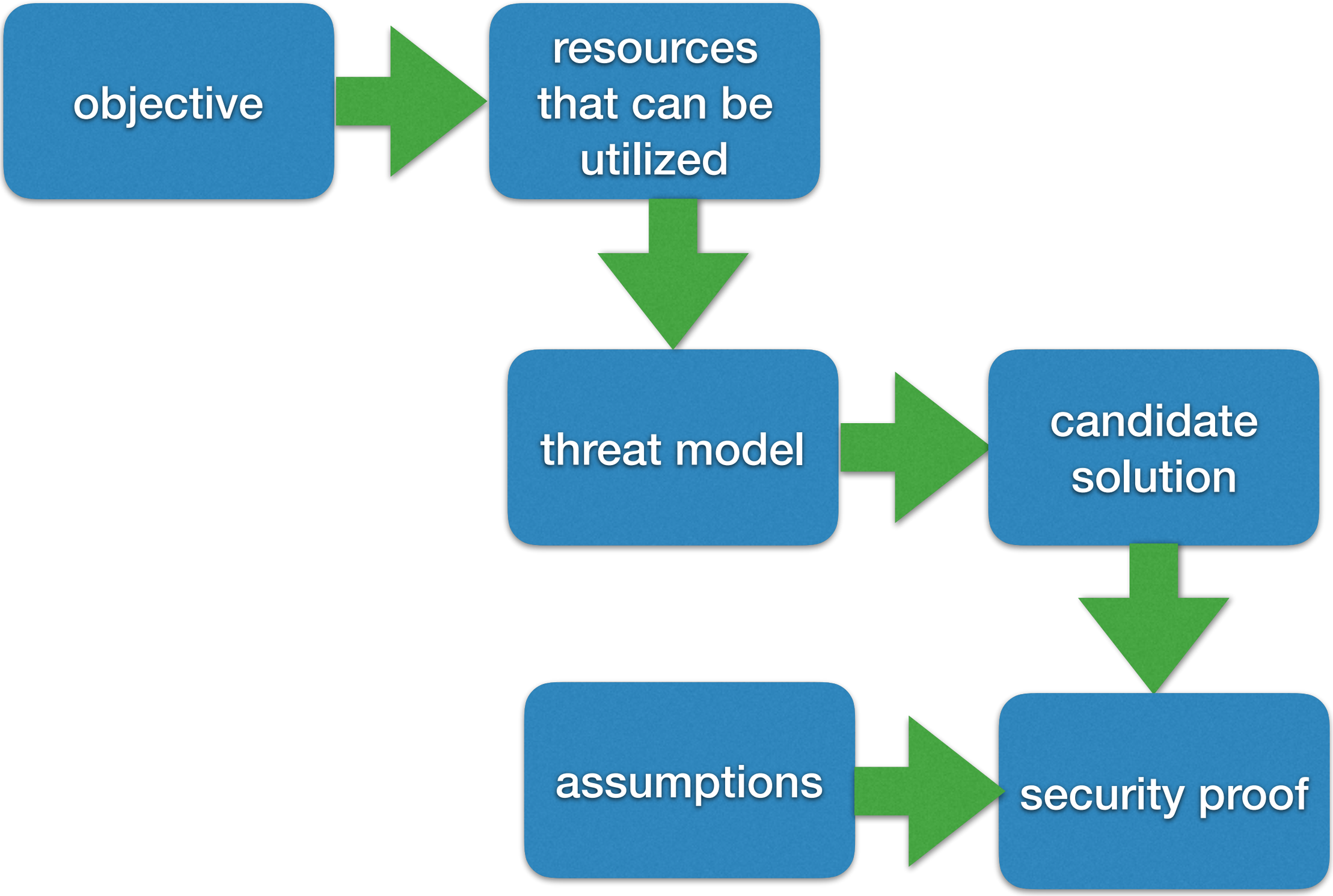
Building Secure Systems



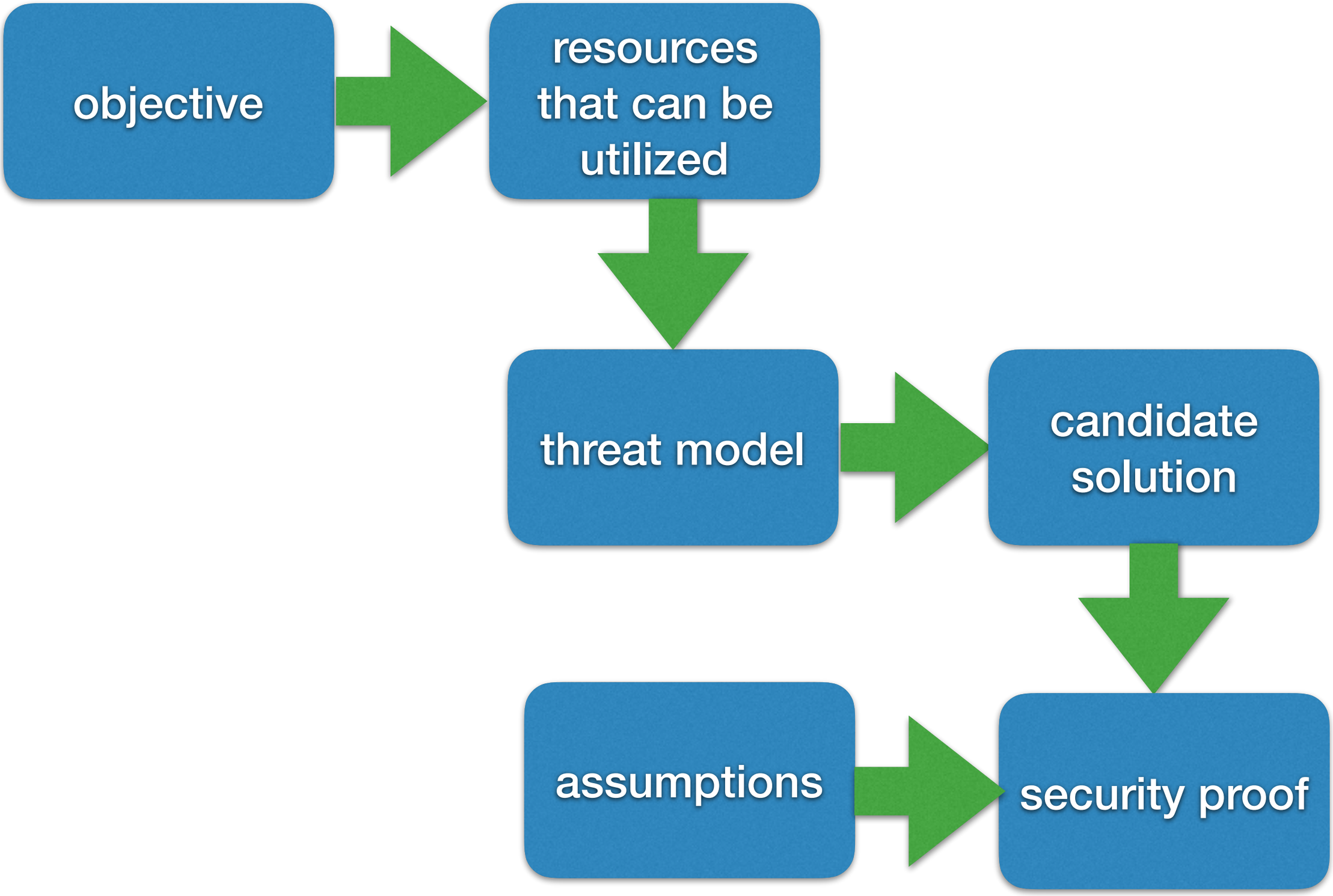
Building Secure Systems



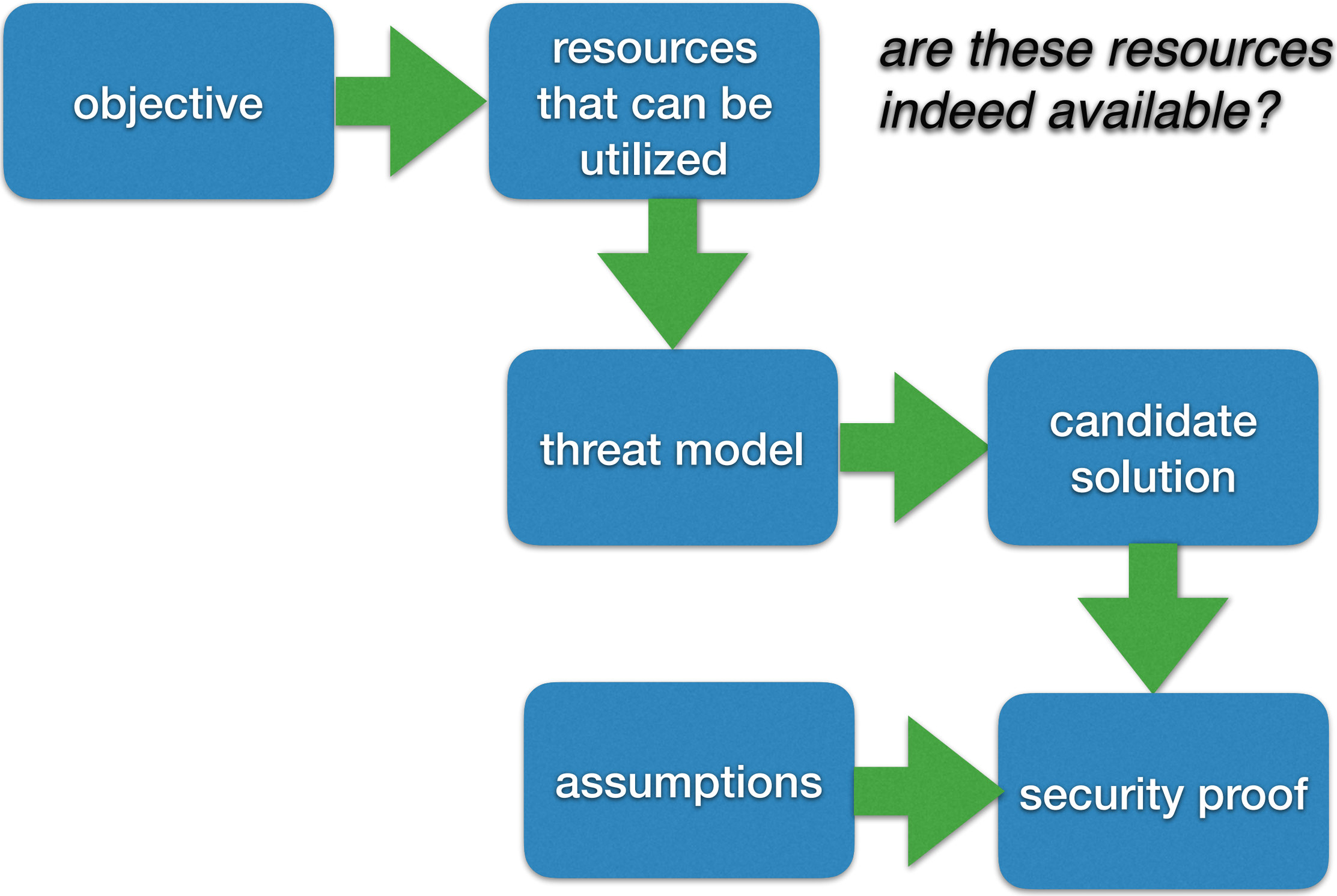
Building Secure Systems



Building Secure Systems

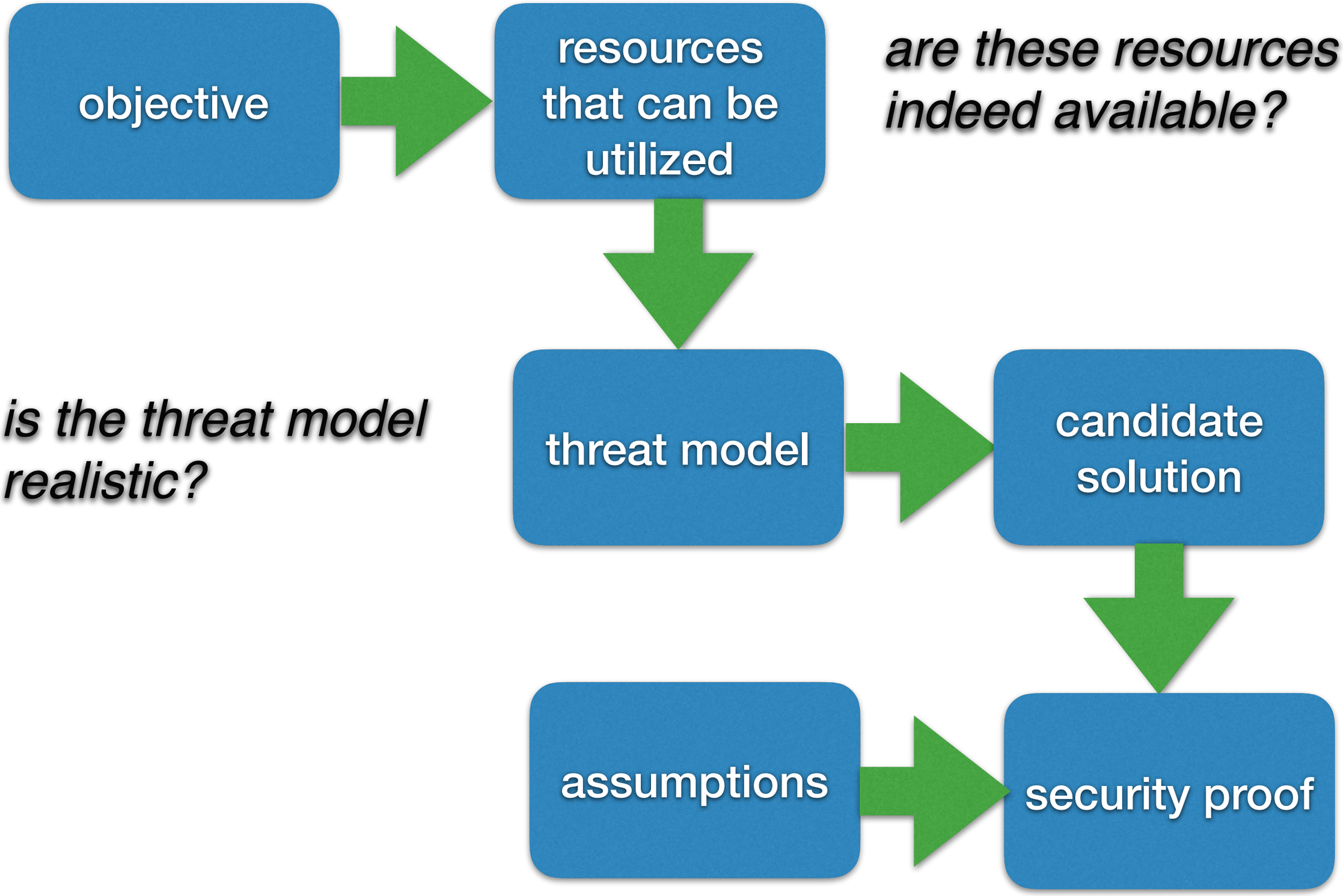


Building Secure Systems



are these resources indeed available?

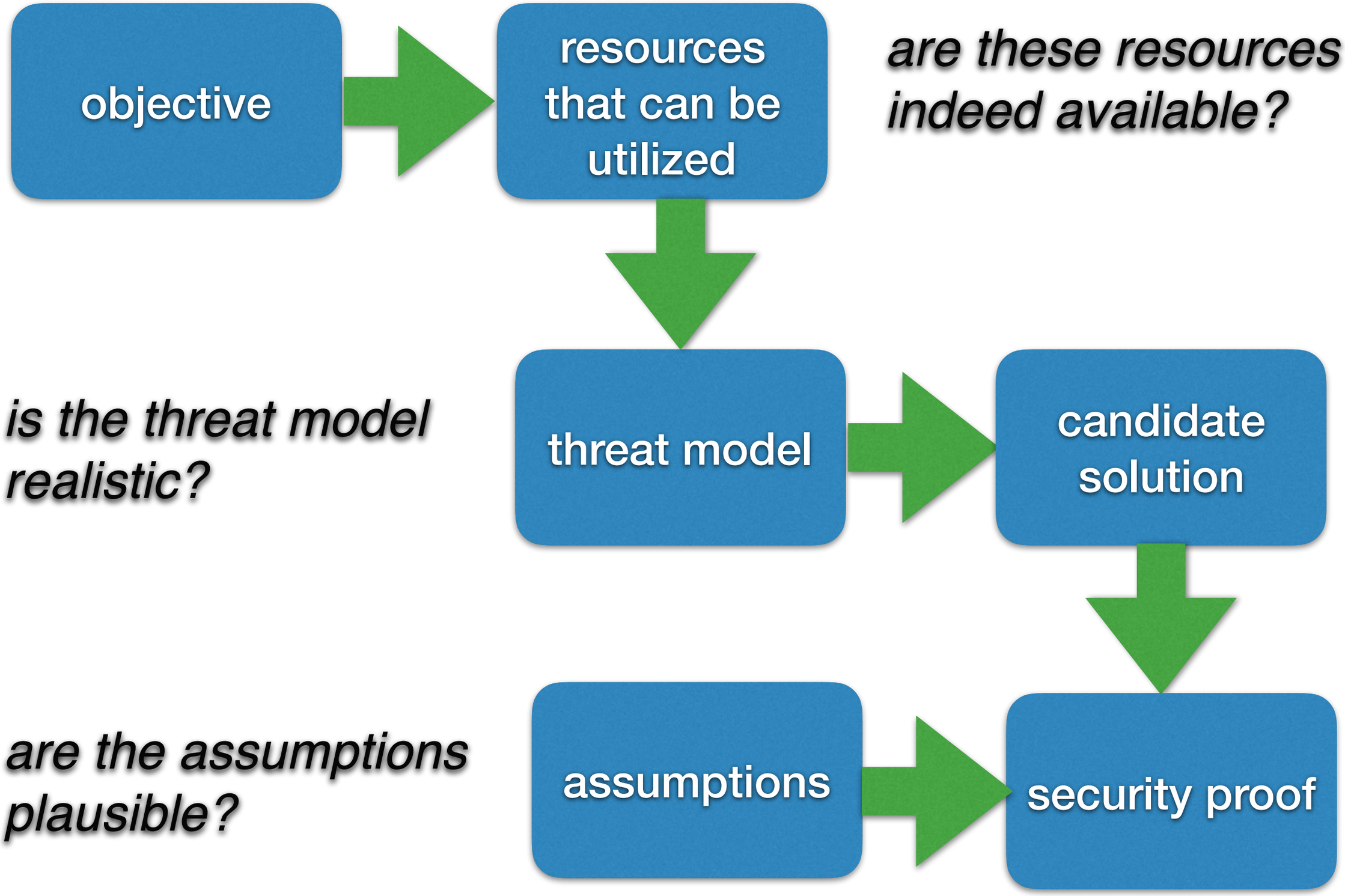
Building Secure Systems



are these resources indeed available?

is the threat model realistic?

Building Secure Systems

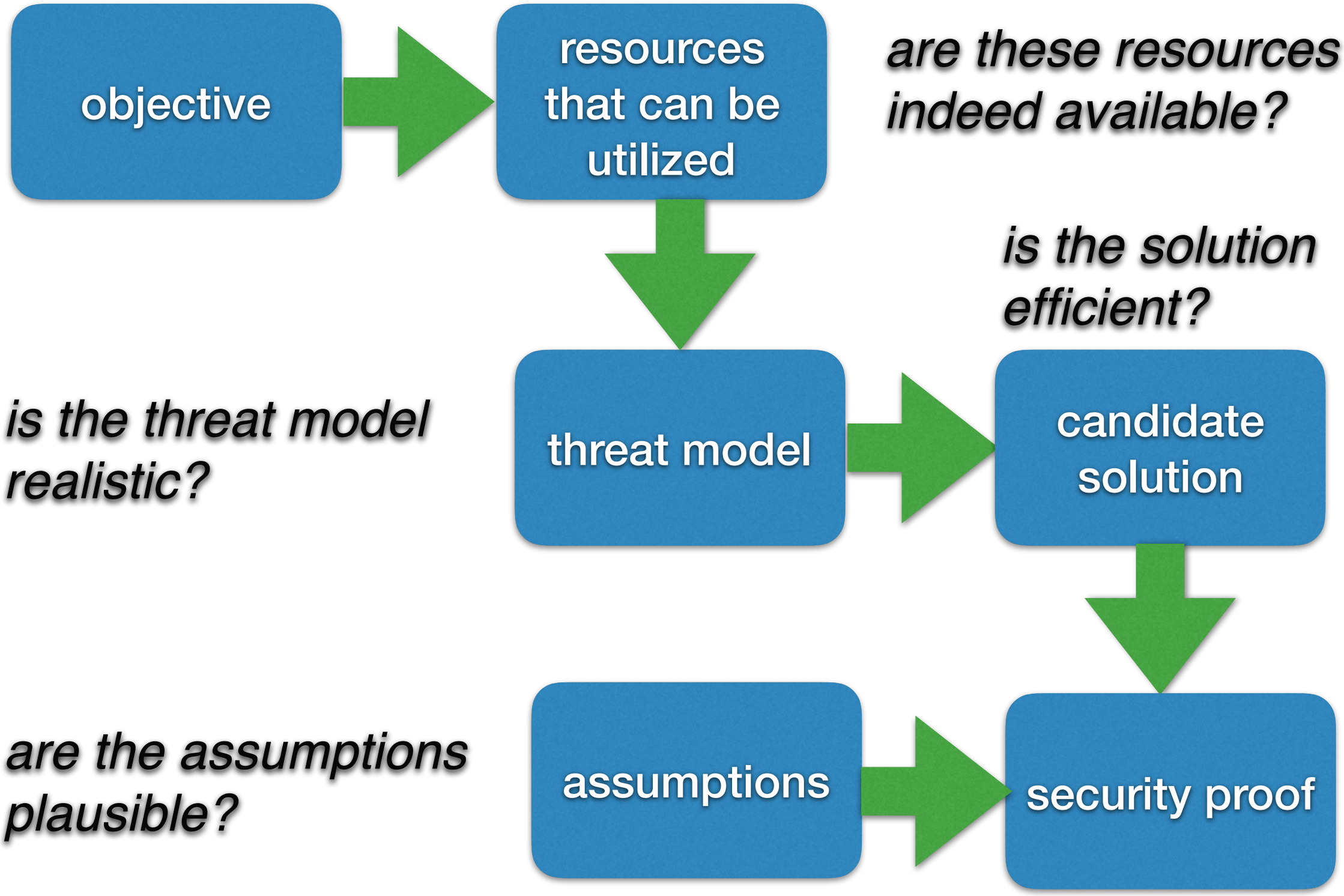


are these resources indeed available?

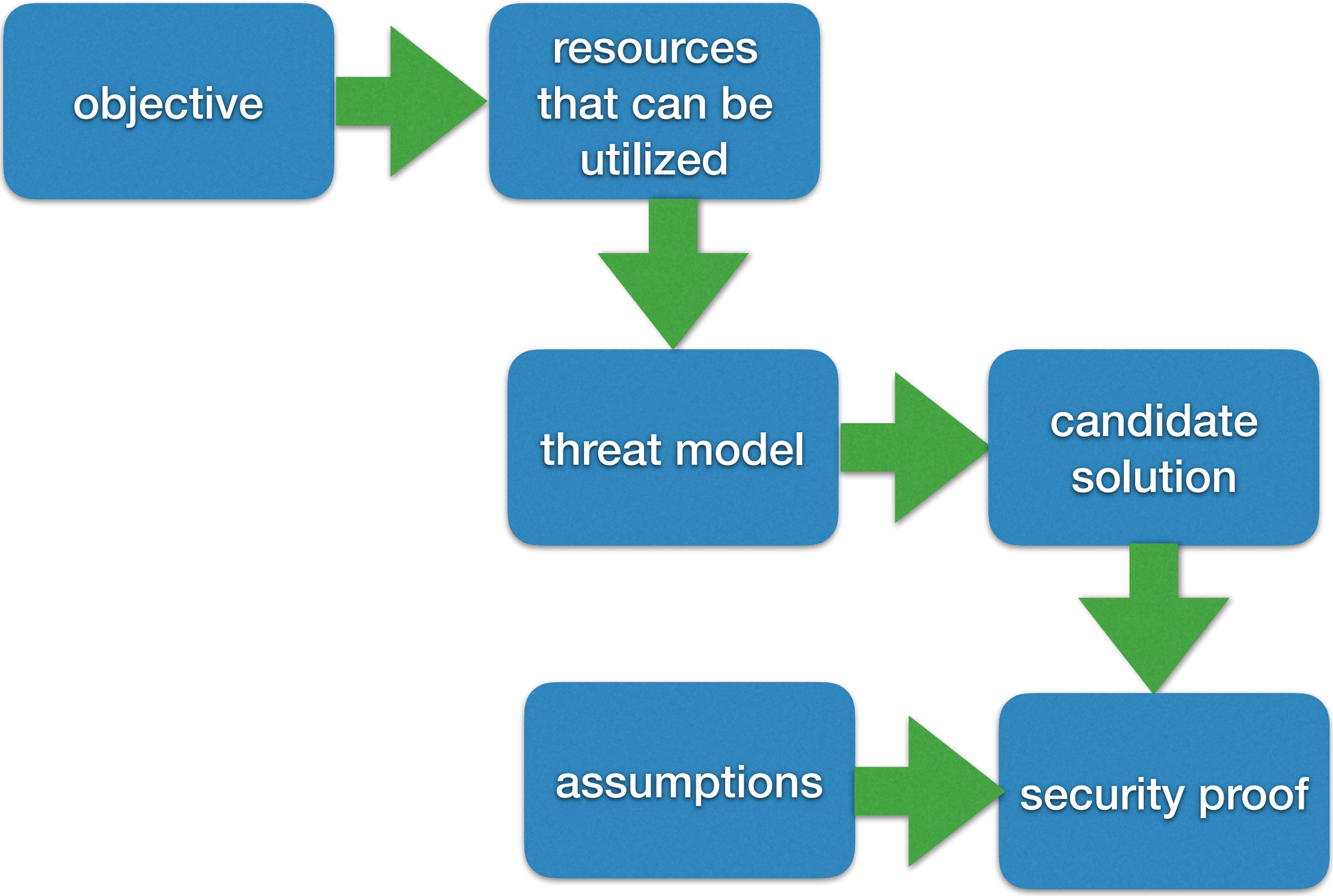
is the threat model realistic?

are the assumptions plausible?

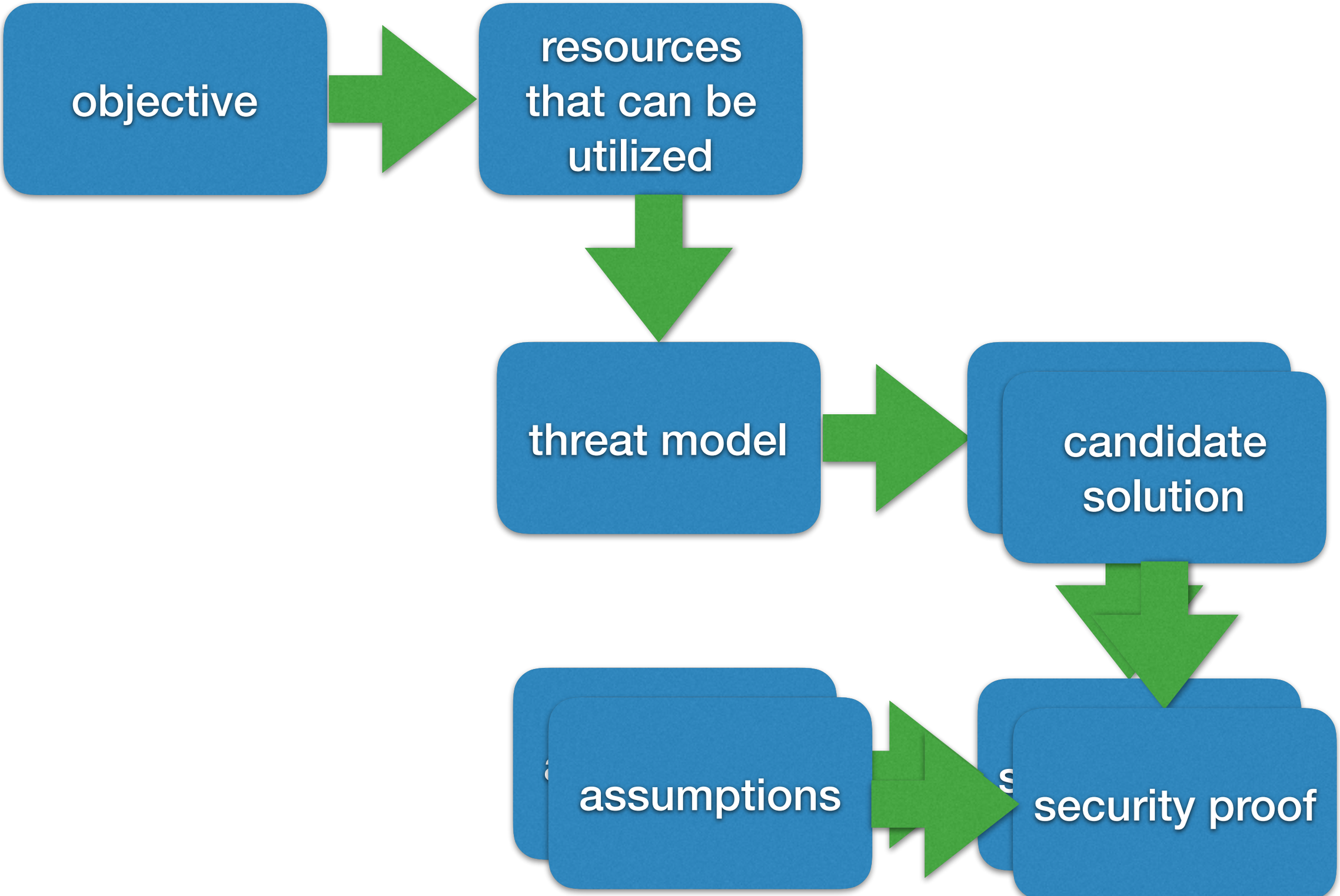
Building Secure Systems



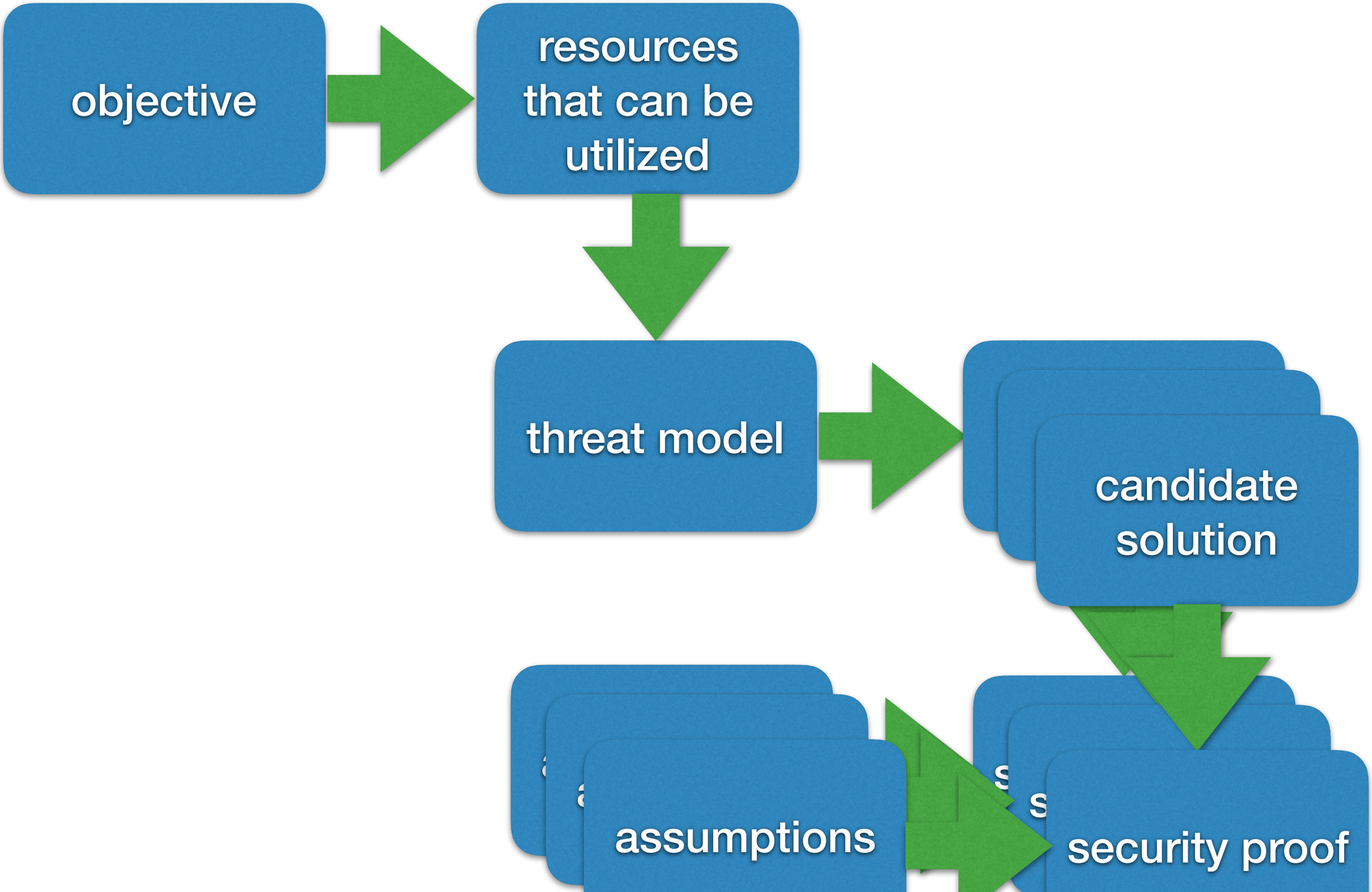
One objective, many solutions



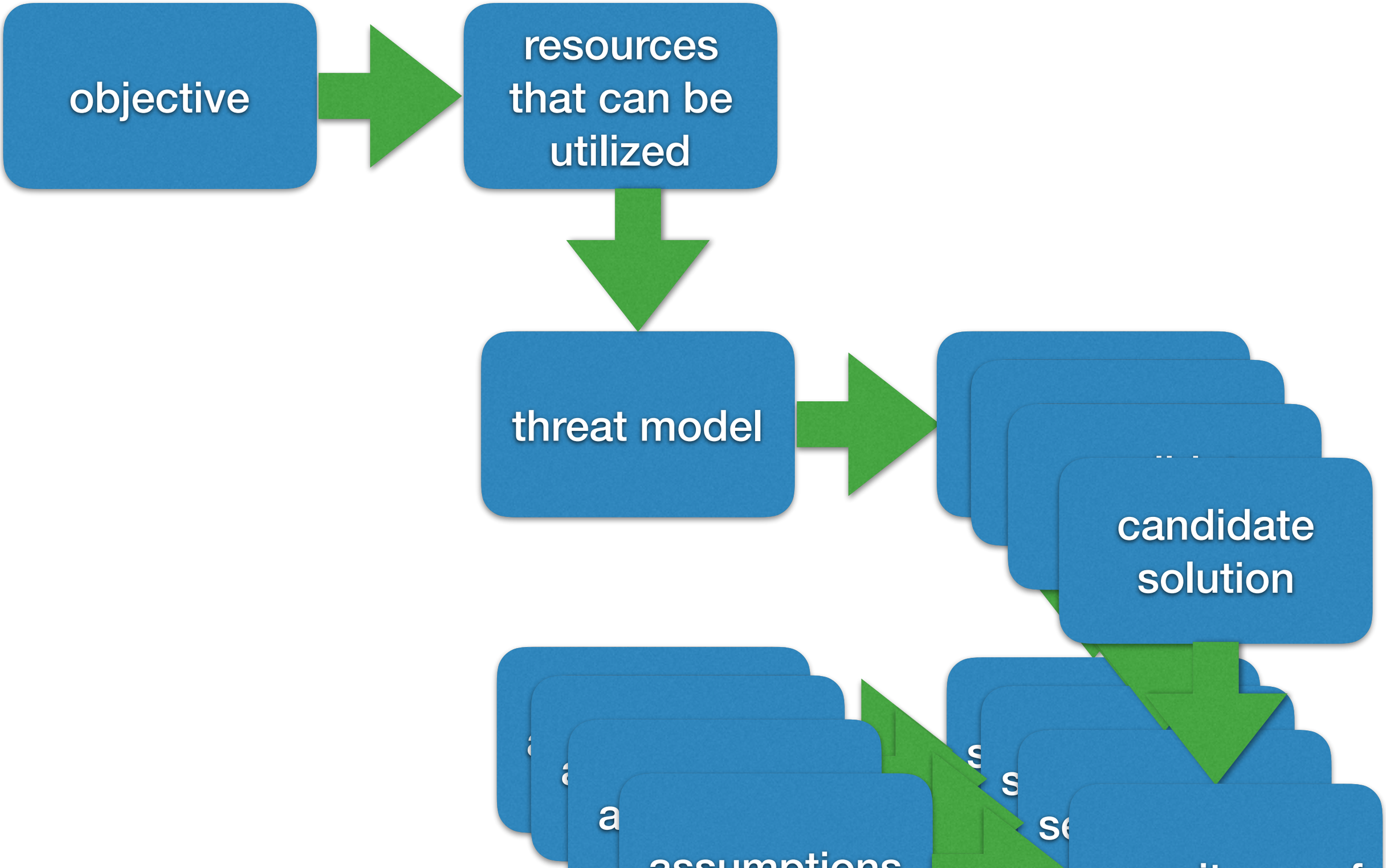
One objective, many solutions



One objective, many solutions



One objective, many solutions



…What is an objective?

- Compare:

Alice and Bob want to communicate securely

vs.

Functionality $\mathcal{F}_{\text{cert-ke}}$

The functionality $\mathcal{F}_{\text{cert-ke}}$ parametrized by a domain D proceeds as follows:

(a) Upon receiving message of the form $(\text{keyexchange}, sid, S, R)$ from some ITI S, if this is the first activation, set $k = \perp$ and send $(\text{keyexchange}, sid, S, R)$ to $\mathcal{S}$. (Otherwise, ignore the message).

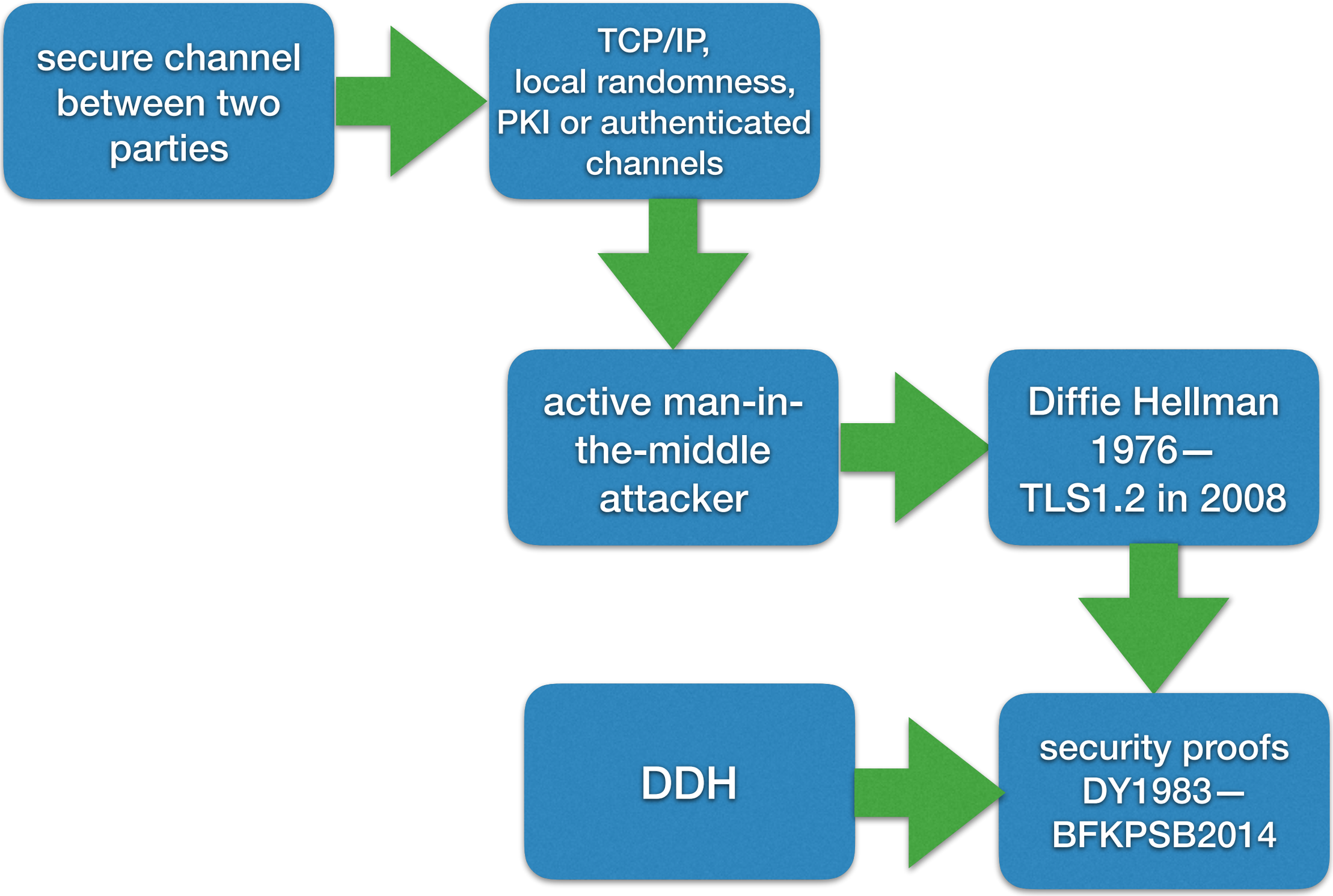
(b) Upon receiving a value $(\text{Corrupt}, sid, P)$ from $\mathcal{S}$, mark $P \in \{S, R\}$ as corrupted and output k to $\mathcal{S}$.

(c) Upon receiving a message of the form $(\text{setkey}, sid, S, R, k')$ from the adversary, if either S or R is corrupt, then set key $k = k'$, else set $k \xleftarrow{\$} \text{D}$. Output a delayed message $(\text{setkey}, sid, S, R, k)$ to S and R and halt.

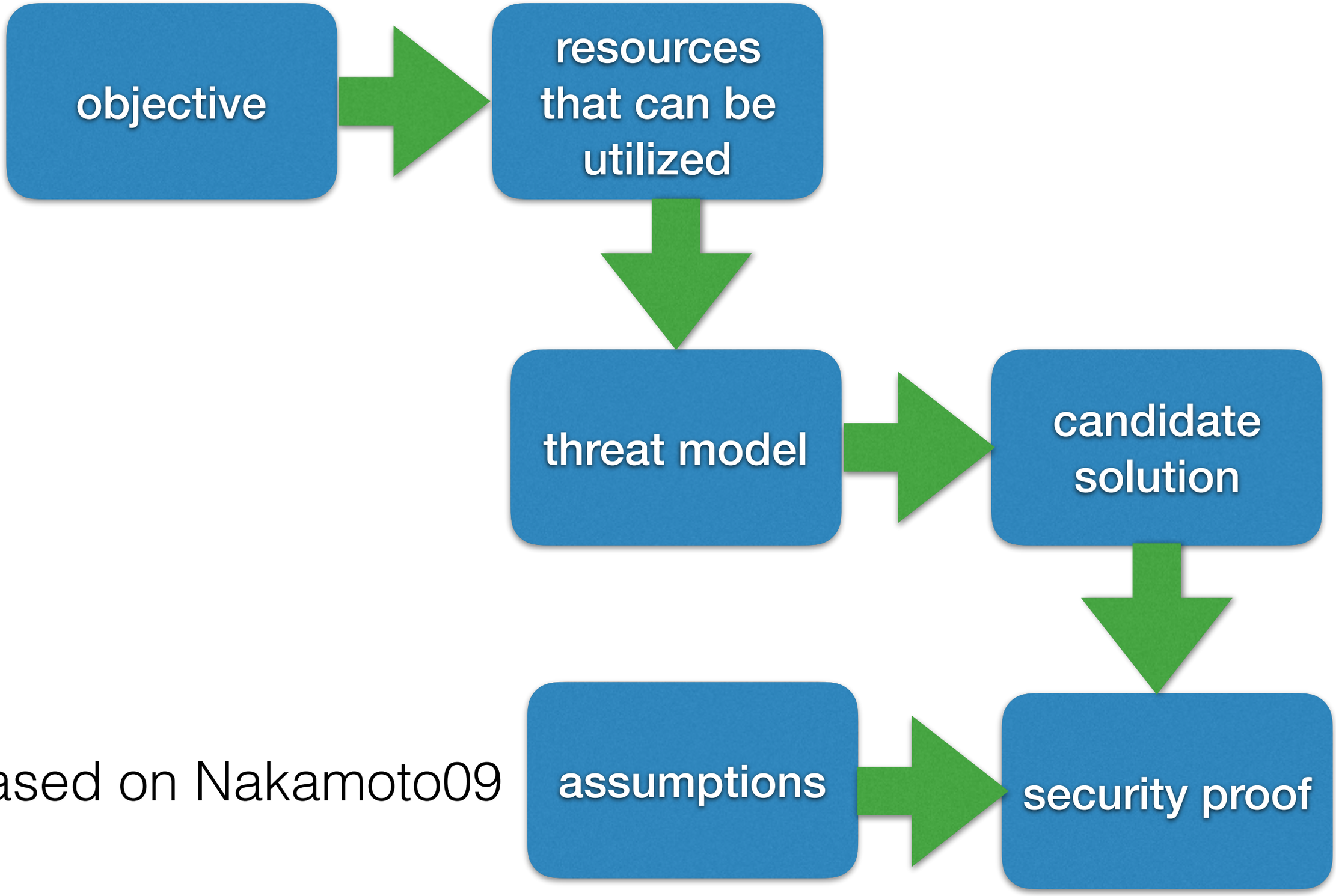
(d) Upon receiving $(\text{External-info}, P, sid, m')$ from the adversary, where $P \in \{S, R\}$, if $k \neq \perp$ and an output was not yet delivered to either party, output $\big(\text{Sign}, (P, (P', sid)), (m', sid, P)\big)$ to $\bar{\mathcal{G}}^{P}_{\text{cert}}$ (where P' is the other party), and forward the response to the adversary.

(e) Upon receiving $(\text{Corrupt-sign}, sid, P, m')$ from the adversary, where $P \in \{S, R\}$, if P is marked as corrupted then output $\big(\text{Sign}, (P, (P', sid)), (m', sid, P)\big)$ to $\bar{\mathcal{G}}^{P}_{\text{cert}}$ and forward the response to the adversary.

Secure Channel as an Objective

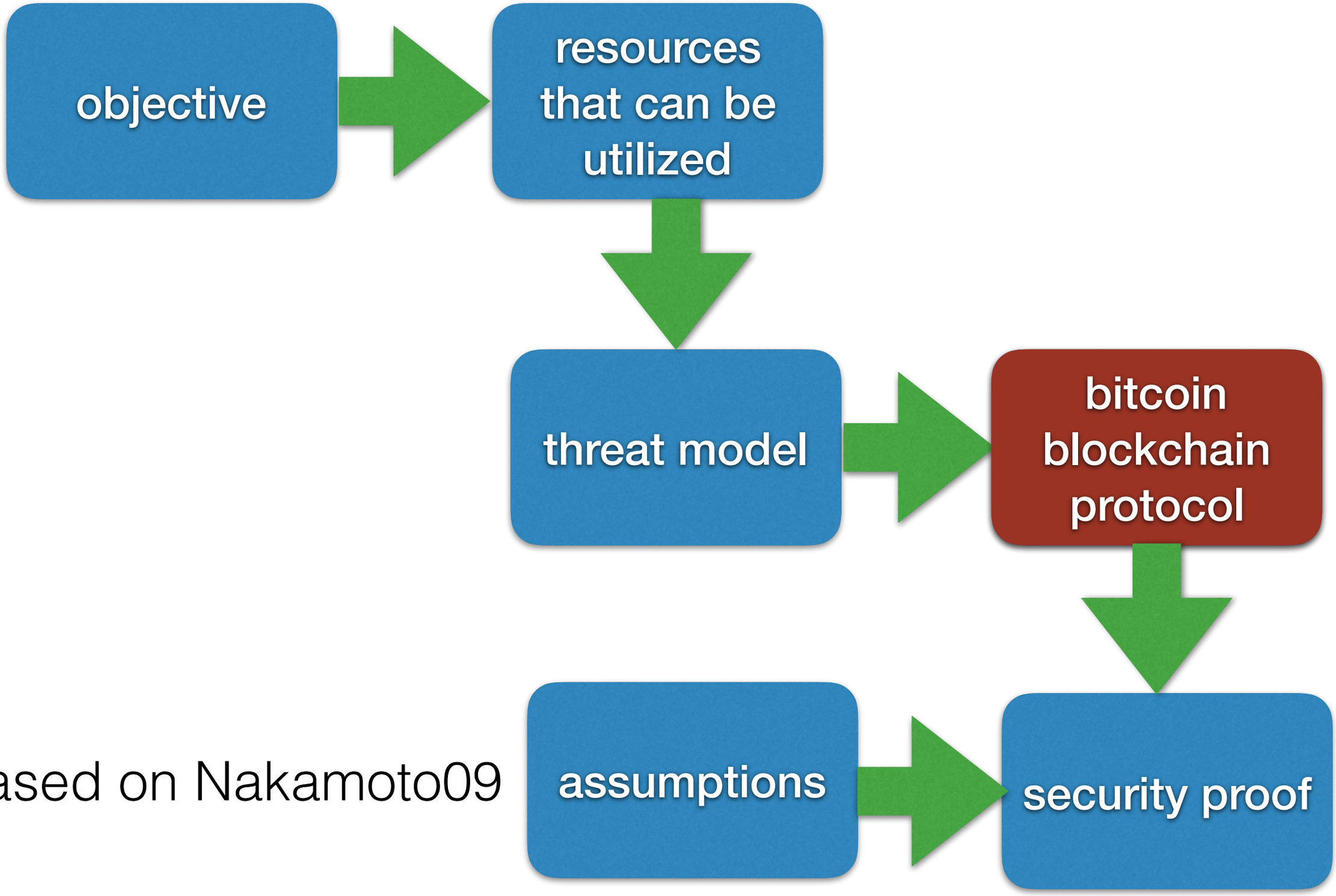


Bitcoin blockchain as an objective?



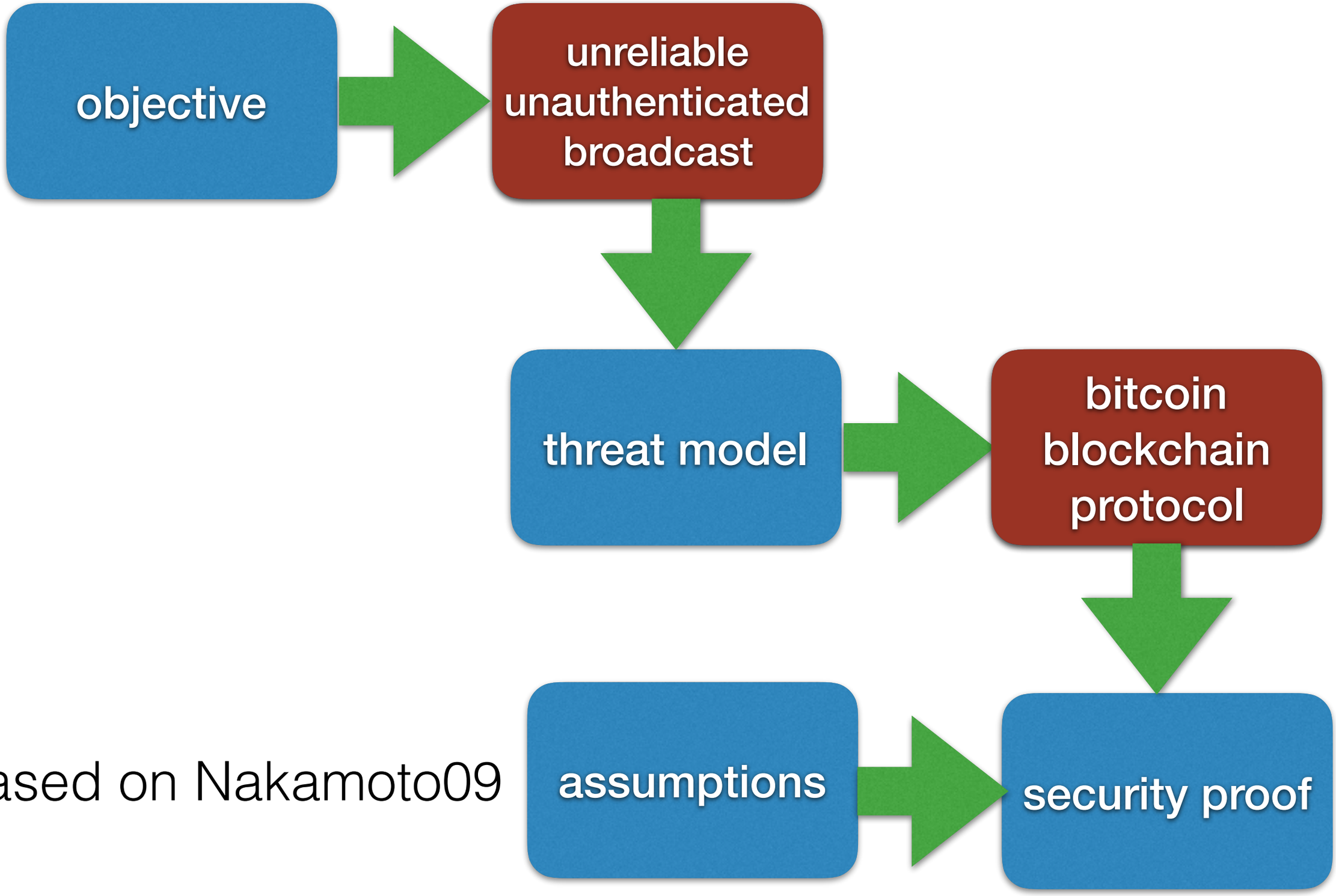
Based on Nakamoto09

Bitcoin blockchain as an objective?



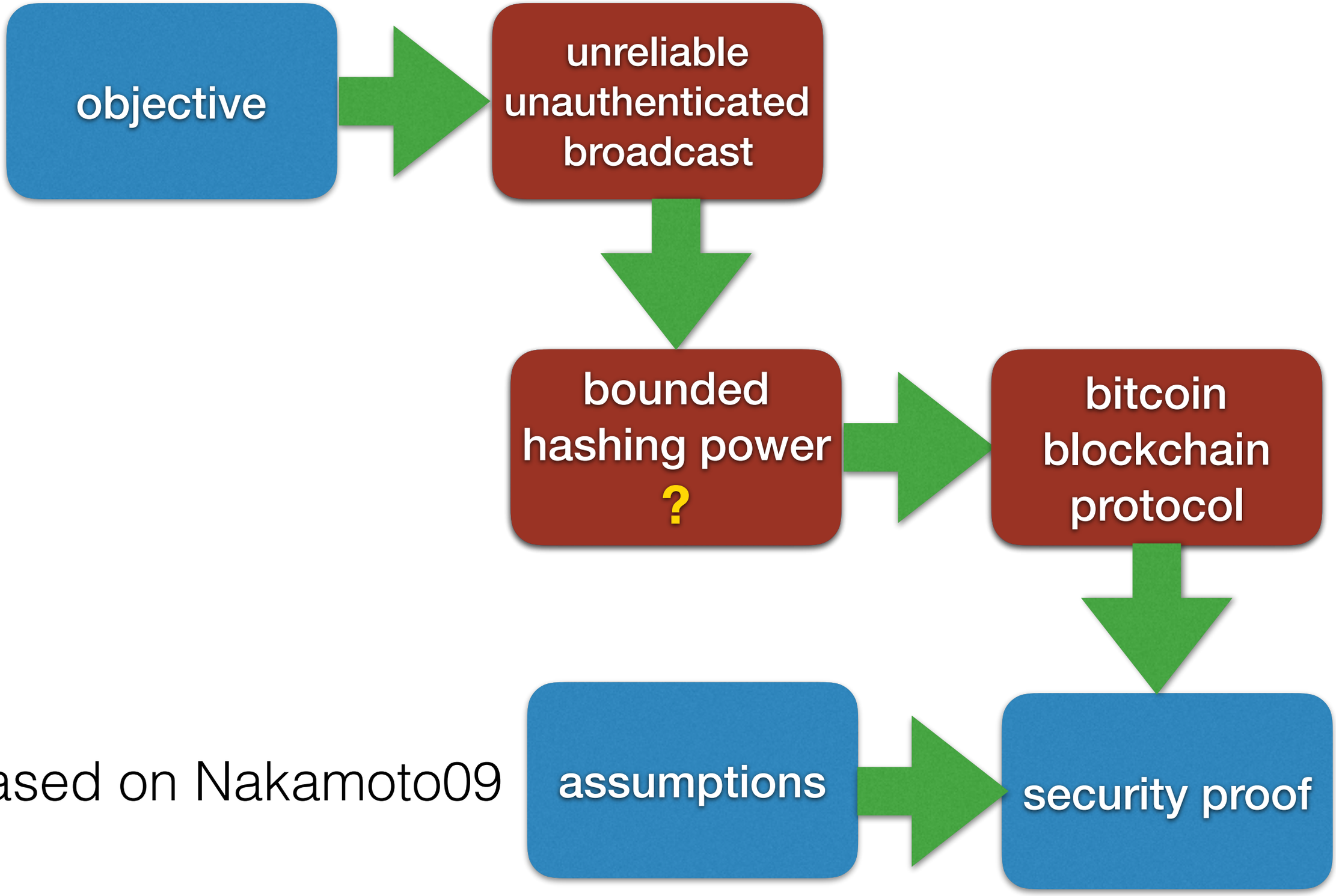
Based on Nakamoto09

Bitcoin blockchain as an objective?



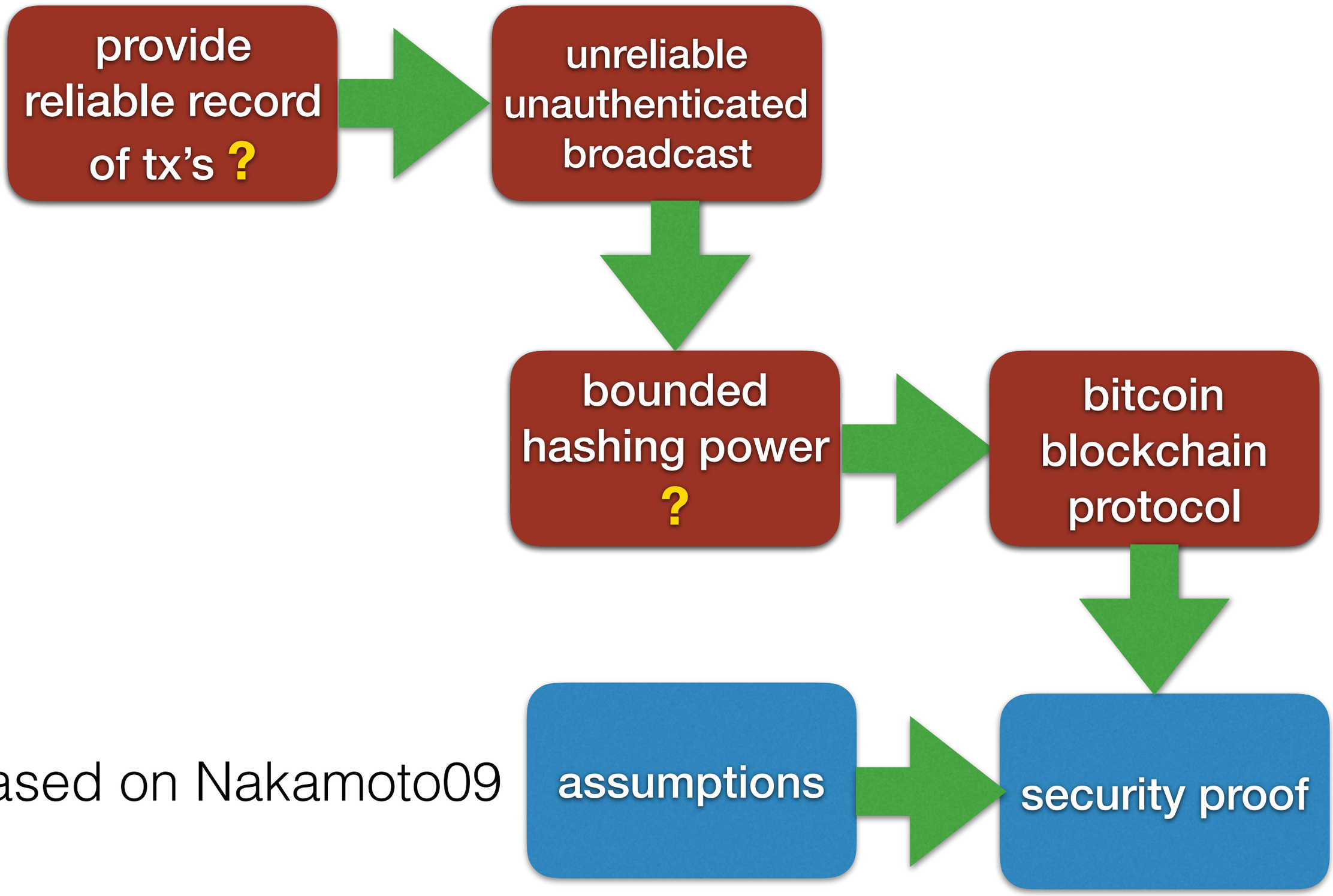
Based on Nakamoto09

Bitcoin blockchain as an objective?



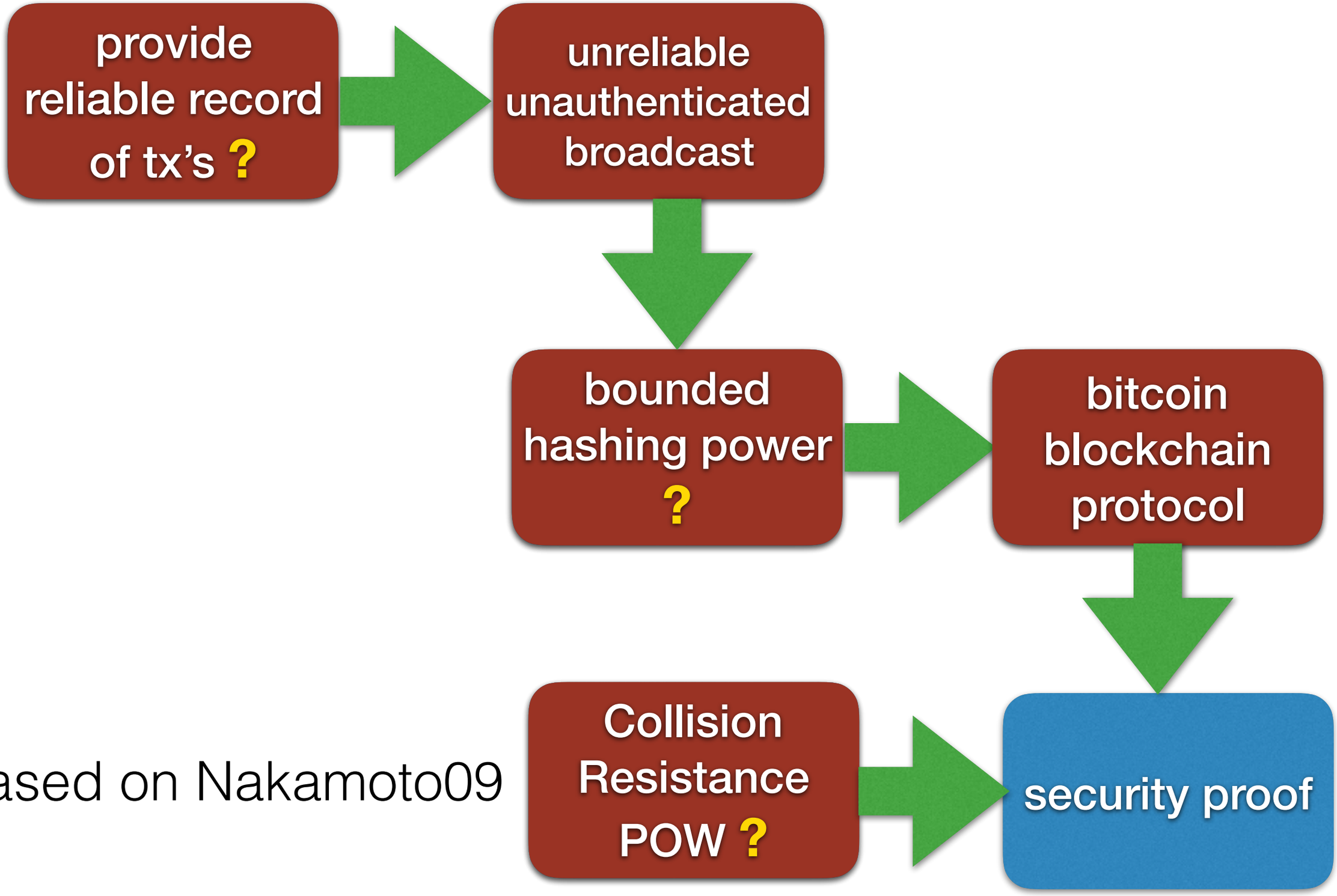
Based on Nakamoto09

Bitcoin blockchain as an objective?



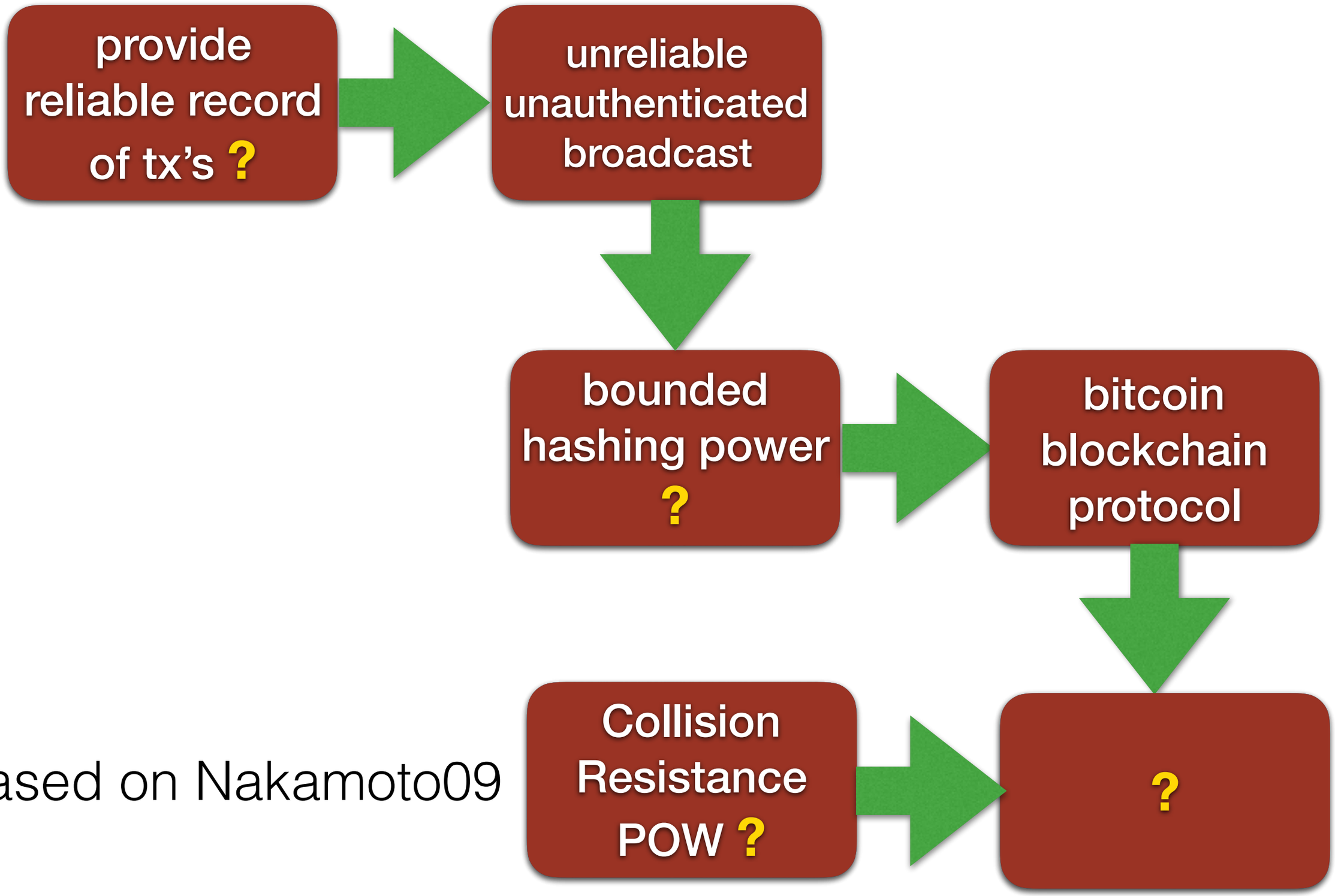
Based on Nakamoto09

Bitcoin blockchain as an objective?



Based on Nakamoto09

Bitcoin blockchain as an objective?



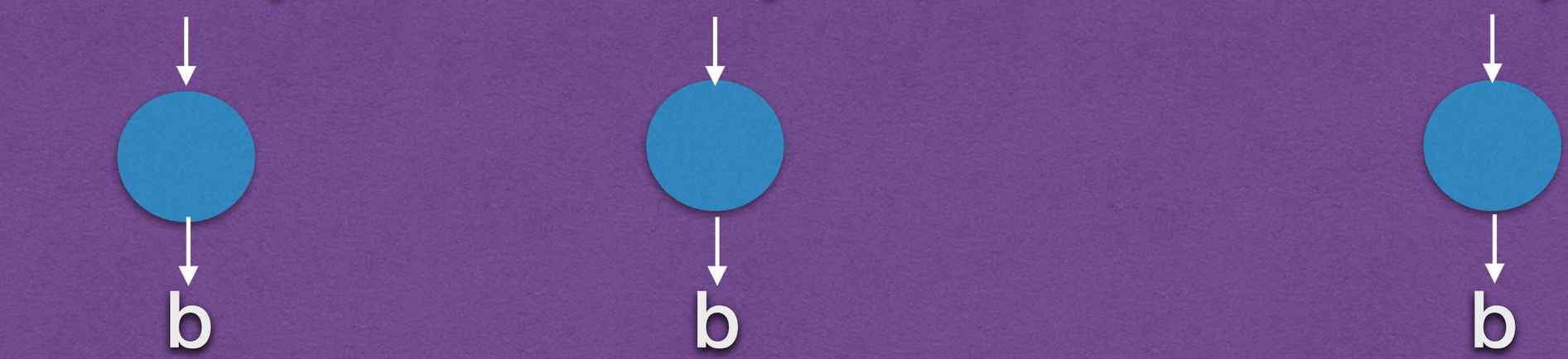
Based on Nakamoto09

What about consensus?

- One of the classical problems in Computer Science, [Lamport,Shostak,Pease 1980].

The Consensus Problem

implementing consensus:

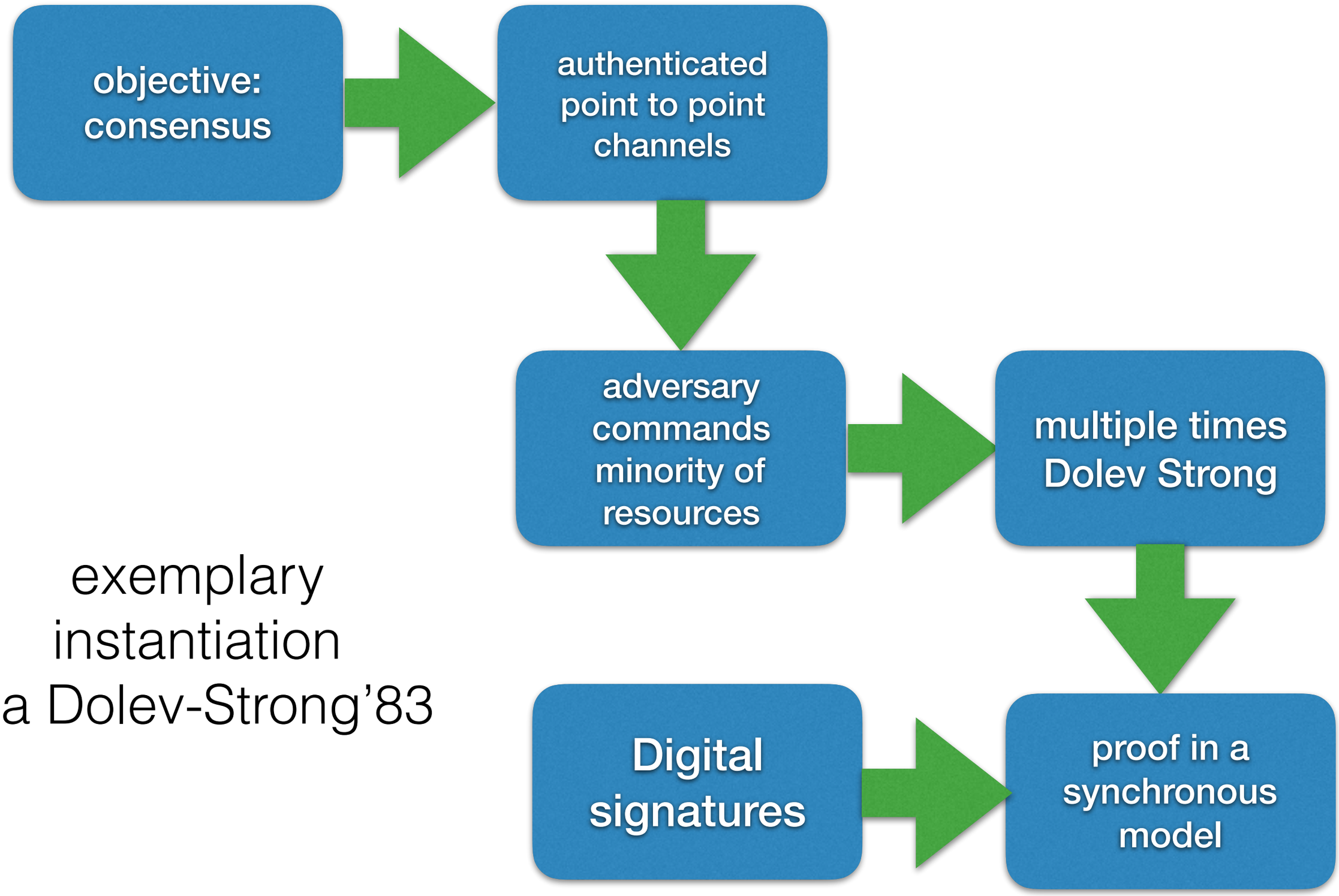


Agreement = all parties output the same value

Validity = if all honest parties have the same insert bit, then this matches the output

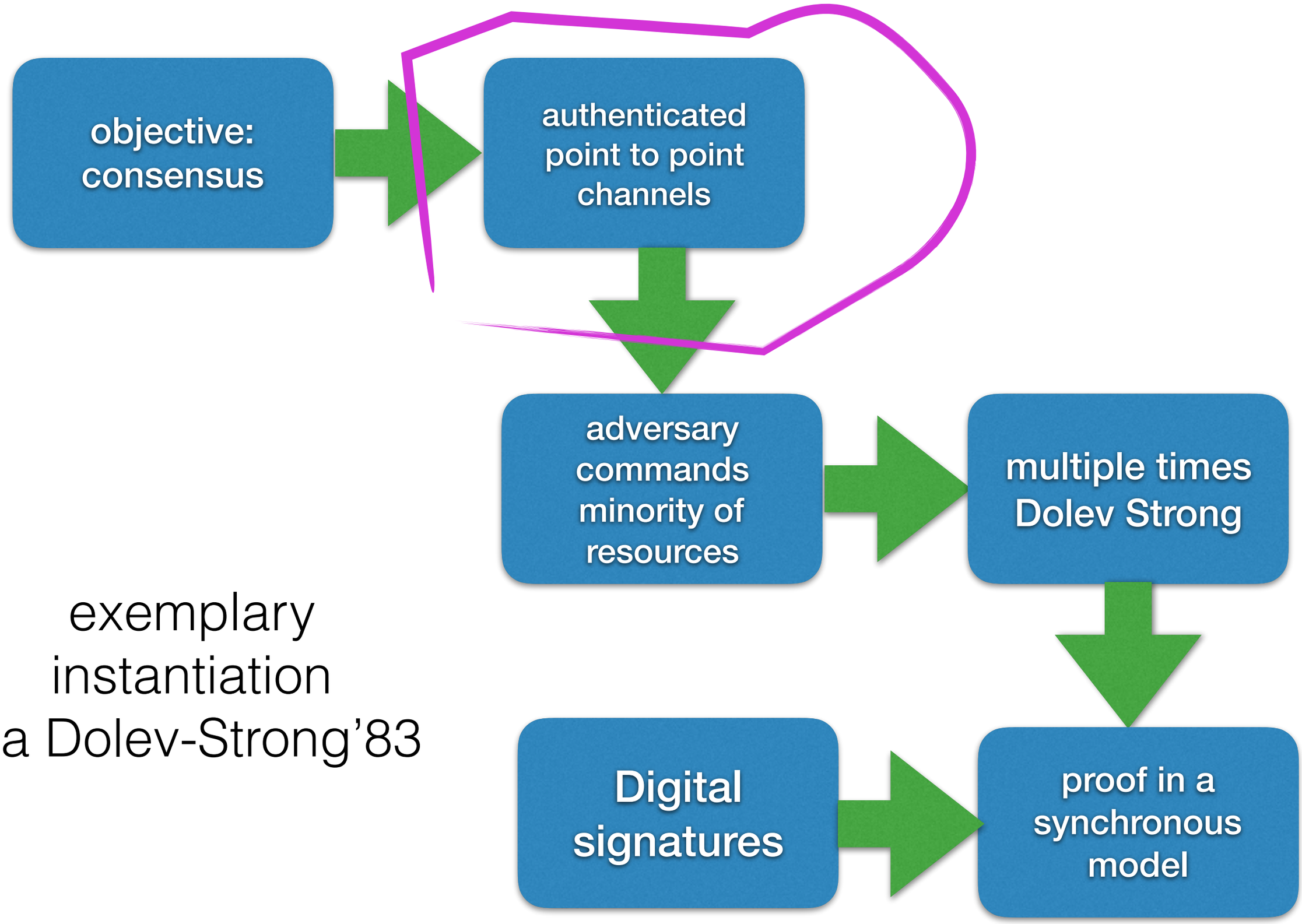
Termination = all honest parties terminate

Consensus as an objective



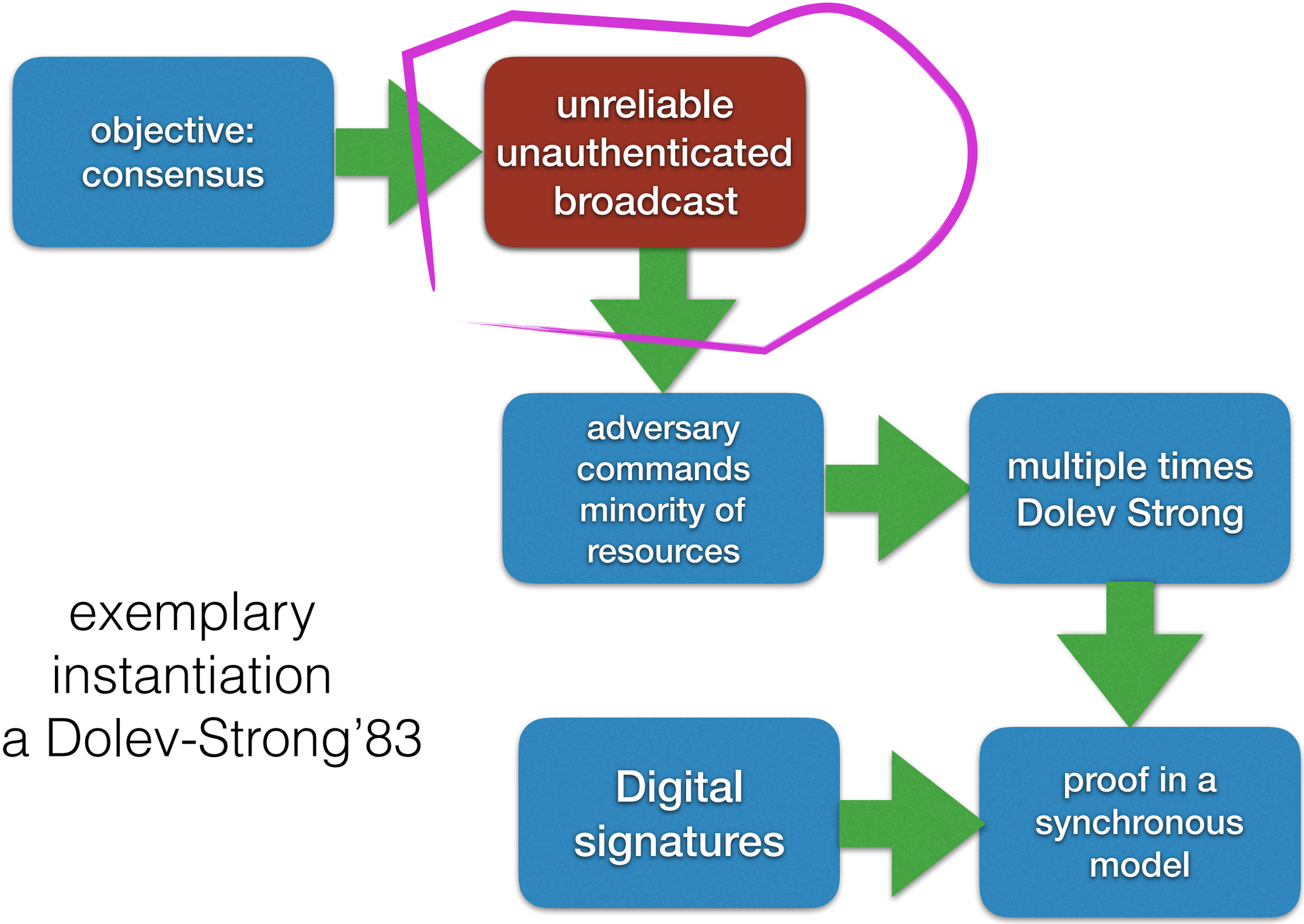
exemplary instantiation via Dolev-Strong'83

Consensus as an objective



exemplary instantiation via Dolev-Strong'83

Consensus as an objective



exemplary instantiation via Dolev-Strong'83

The Ledger Objective

- First formal definition of the objective of a “robust transaction ledger” was formulated by Garay, **K**, Leonardos in [GKL14]

https://eprint.iacr.org/2014/765

The Ledger Objective

- First formal definition of the objective of a “robust transaction ledger” was formulated by Garay, **K**, Leonardos in [GKL14]

https://eprint.iacr.org/2014/765

- In the same work, we proved that a suitable abstraction of the bitcoin protocol (the bitcoin backbone) realizes the ledger objective

- … and also can be used to achieve other primitives such as consensus (with some work..)

Defining the ledger objective

imagine that time is divided in rounds

and protocol organizes transactions in a sequence of blocks

Persistence: parameter *k*. If an honest party reports a transaction tx as "stable" (>*k* blocks deep) then, whenever an honest party reports it as stable, it will be in the same position

Liveness: parameters *u*, *k*. If all honest parties attempt to insert the transaction *tx* in the ledger, then, after *u rounds*, all honest parties will report it as stable (>*k* blocks deep) and will always do so

transaction processing time : ***u* as a function of *k***

Synchronous Model

- Time is divided in rounds.

- In each round each party is allowed *q* queries to a hash function (RO)

- messages are sent through a “diffusion” mechanism

- The adversary is rushing and may :
  1. spoof messages
  2. inject messages
  3. reorder messages

Model Participants

- There are n-t honest parties each one producing q queries to the hash function per round.

- The adversary is able to control t parties acting as a malicious mining pool.

- A “flat” version of the world in terms of hashing power.

- It is worse for honest parties to be separate (they have to pay the price of being decentralized).

Execution & View

3 PPT machines

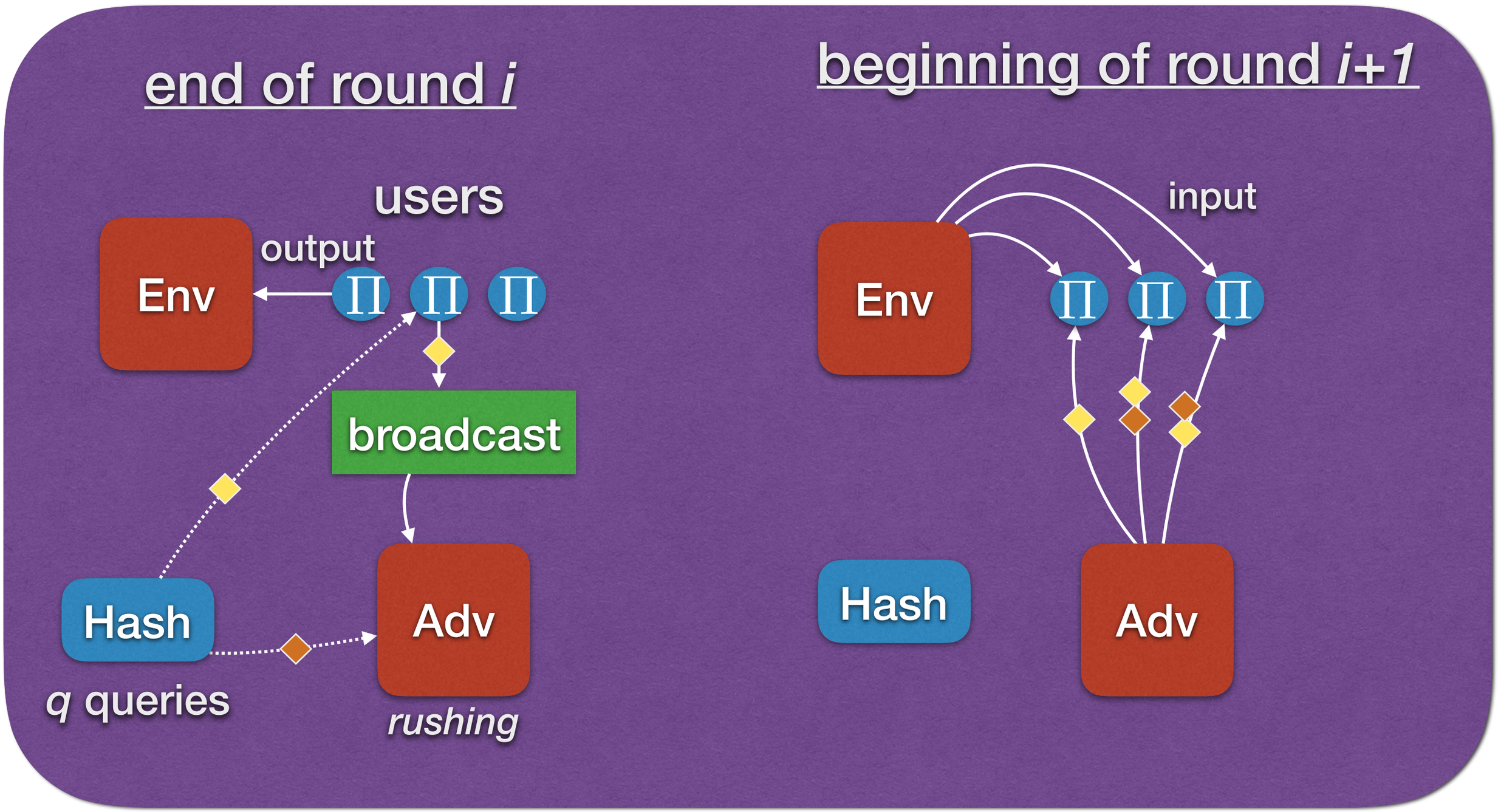
protocol Π
adversary $\mathcal{A}$
environment $\mathcal{Z}$

n parties

$\text{VIEW}^{\Pi}_{\mathcal{A},\mathcal{Z}}(1^{\lambda})$ concatenation of the view of each party at each round

random variable with support :
1. coins of $\mathcal{A}, \mathcal{Z}, n$ copies of Π
2. Random oracle

Round structure



Property of a protocol

fix
a protocol Π
a number of parties *n*, *t* of which
controlled by adversary
a predicate Q

We say that the protocol has property Q
with error ϵ if and only if

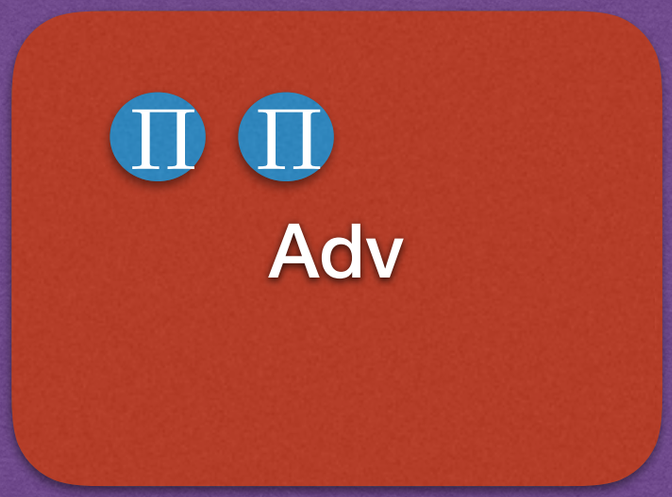
$$\forall \mathcal{A}\ \forall \mathcal{Z}\ \text{Prob}[Q(\text{VIEW}^{\Pi}_{\mathcal{A},\mathcal{Z}}(1^{\lambda})] \geq 1 - \epsilon$$

typically : $\epsilon = \text{negl}(\lambda)$

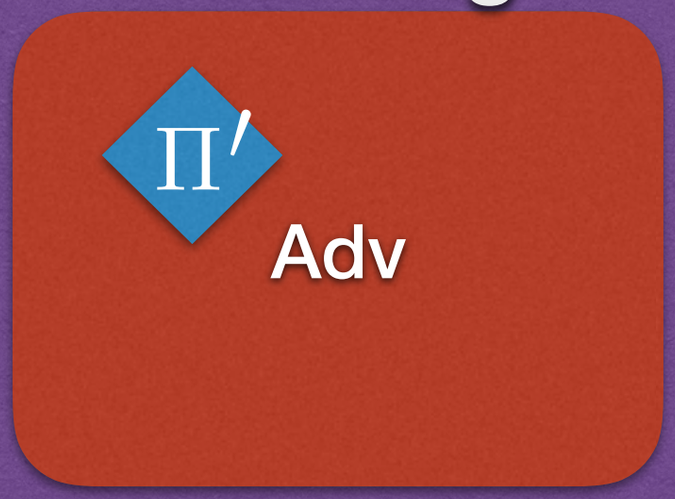
Generality of the model

- We quantify over all possible adversaries; this includes:

some parties receiving only some of the messages



a large mining pool that is performing some type of selfish mining



Or any combination thereof

The Bitcoin Backbone Protocol

[Garay-K-Leonardos2014]

- An abstraction based on the bitcoin implementation.

- **Importantly** : it distinguishes between data structure (blockchain) and application layer (transactions).

Bitcoin Backbone (1)

parameterized by $V(\cdot), I(\cdot), R(\cdot)$
and $G(\cdot), H(\cdot)$ hash functions

- players have a state $\mathcal{C}$ in the form of a "blockchain":

The *contents* of $\mathcal{C}$ satisfy the predicate

$$V(x_1, \ldots, x_i) = \text{true}$$

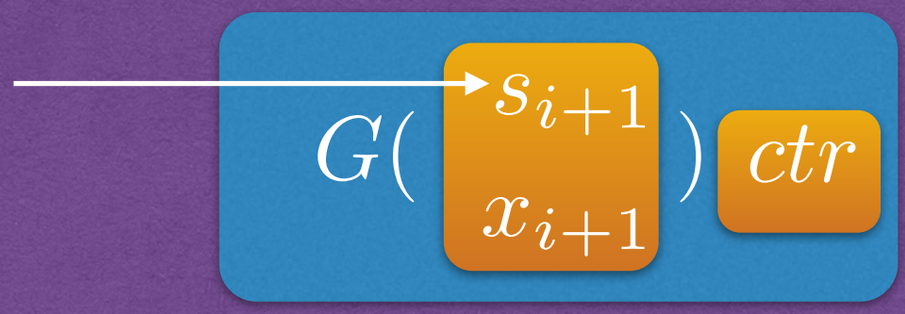
Bitcoin Backbone (2)

parameterized by $V(\cdot), I(\cdot), R(\cdot)$
and $G(\cdot), H(\cdot)$ hash functions

- Within a round, players obtain (INSERT, x) symbols from the environment and network and process them

$$x_{i+1} = I(\ldots \text{all local info} \ldots)$$

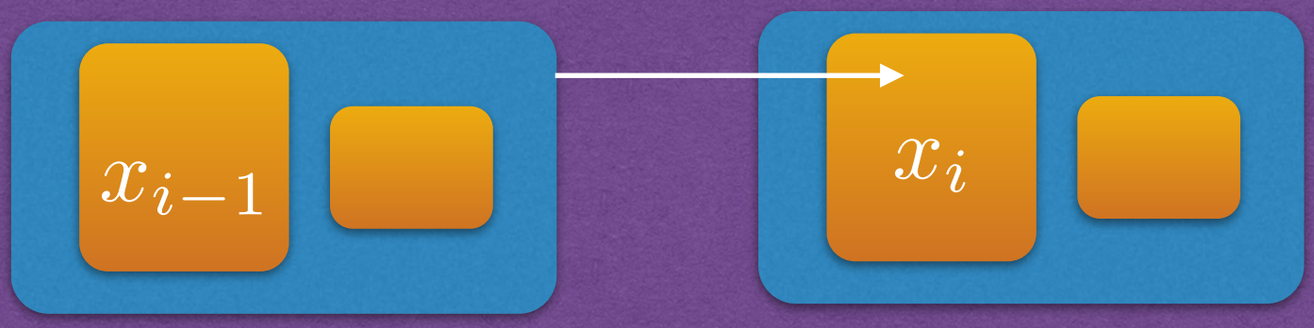
- Then they use their *q* queries to $H(\cdot)$ to obtain a new block by trying $ctr = 0, 1, 2, \ldots$



Bitcoin Backbone (3)

parameterized by $V(\cdot), R(\cdot), I(\cdot)$

- If a player finds a new block it extends $\mathcal{C}$



- The new $\mathcal{C}$ is propagated to all players via the (unreliable/anonymous) broadcast

Bitcoin Backbone (3)

parameterized by $V(\cdot), R(\cdot), I(\cdot)$

- If a player finds a new block it extends $\mathcal{C}$

- The new $\mathcal{C}$ is propagated to all players via the (unreliable/anonymous) broadcast

Bitcoin Backbone (4)

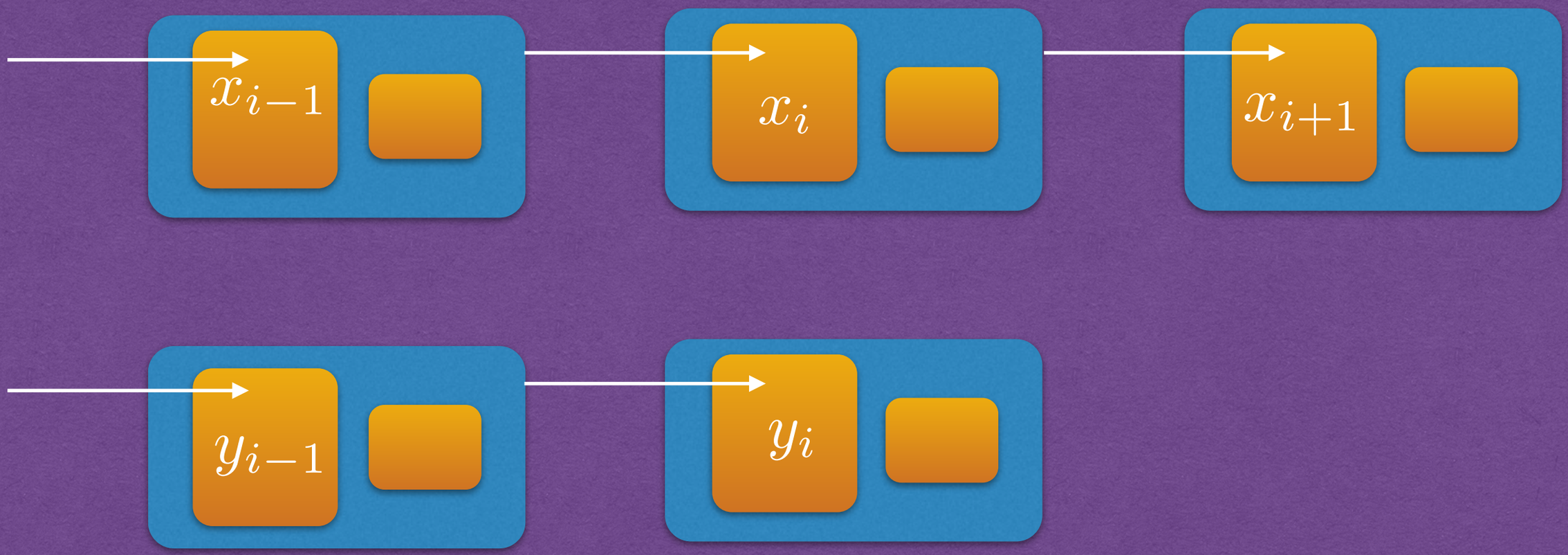
- A player will compare any incoming chains and the local chain w.r.t. their length/difficulty

- Finally a player given a (Read) symbol it will return

$$R(x_1, x_2, \ldots, x_{i+1})$$

Bitcoin Backbone (4)

- A player will compare any incoming chains and the local chain w.r.t. their length/difficulty

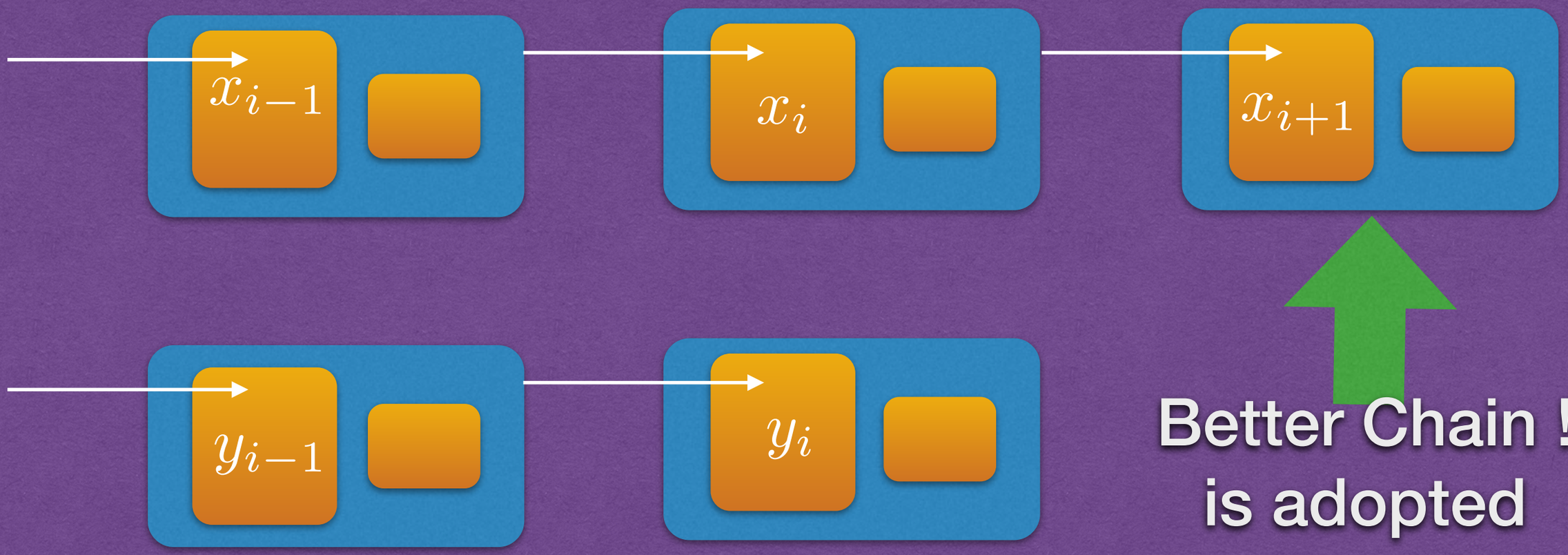


- Finally a player given a (Read) symbol it will return

$$R(x_1, x_2, \ldots, x_{i+1})$$

Bitcoin Backbone (4)

- A player will compare any incoming chains and the local chain w.r.t. their length/difficulty



- Finally a player given a (Read) symbol it will return

$$R(x_1, x_2, \ldots, x_{i+1})$$

Validate

1: **function** validate($\mathcal{C}$)
2: $b \leftarrow V(\mathbf{x}_{\mathcal{C}})$
3: **if** $b \wedge (\mathcal{C} \neq \varepsilon)$ **then** ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
4: $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$
5: $s' \leftarrow H(ctr, G(s, x))$
6: **repeat**
7: $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$
8: **if** $\text{validblock}_q^T(\langle s, x, ctr \rangle) \wedge (H(ctr, G(s, x)) = s')$ **then**
9: $s' \leftarrow s$ ▷ Retain hash value
10: $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$ ▷ Remove the head from $\mathcal{C}$
11: **else**
12: $b \leftarrow \text{False}$
13: **end if**
14: **until** $(\mathcal{C} = \varepsilon) \vee (b = \text{False})$
15: **end if**
16: **return** (b)
17: **end function**

POW

1: **function** pow($x, \mathcal{C}$)
2: **if** $\mathcal{C} = \varepsilon$ **then** ▷ Determine proof of work instance
3: $s \leftarrow 0$
4: **else**
5: $\langle s', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$
6: $s \leftarrow H(ctr', G(s', x'))$
7: **end if**
8: $ctr \leftarrow 1$
9: $B \leftarrow \varepsilon$
10: $h \leftarrow G(s, x)$
11: **while** $(ctr \leq q)$ **do**
12: **if** $(H(ctr, h) < T)$ **then** ▷ This $H(\cdot)$ invocation subject to the q-bound
13: $B \leftarrow \langle s, x, ctr \rangle$
14: **break**
15: **end if**
16: $ctr \leftarrow ctr + 1$
17: **end while**
18: $\mathcal{C} \leftarrow \mathcal{C}B$ ▷ Extend chain
19: **return** $\mathcal{C}$
20: **end function**

Main Loop

1: $\mathcal{C} \leftarrow \varepsilon$
2: $state \leftarrow \varepsilon$
3: $round \leftarrow 0$
4: **while** TRUE **do**
5: $\quad \widetilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, \text{all chains found in } \textsc{Receive}())$
6: $\quad \langle state, x \rangle \leftarrow I(state, \mathcal{C}, round, \textsc{Input}(), \textsc{Receive}())$ ▷ Determine the x-value.
7: $\quad \mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \widetilde{\mathcal{C}})$
8: $\quad$ **if** $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ **then**
9: $\quad\quad \mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
10: $\quad\quad \textsc{Broadcast}(\mathcal{C})$
11: $\quad$ **end if**
12: $\quad round \leftarrow round + 1$
13: $\quad$ **if** $\textsc{Input}()$ contains $\textsc{Read}$ **then**
14: $\quad\quad$ **write** $R(\mathbf{x}_{\mathcal{C}})$ to $\textsc{Output}()$
15: $\quad$ **end if**
16: **end while**

Requirements

- Input Validity. Function I(.) produces inputs acceptable according to V(.)

- Input Entropy. Function I(.) on the same input, will not produce the same output with overwhelming probability.

Input Entropy

$$H(car, G(s, x)) < T$$

- Simplifying assumption: I(.) chooses a random nonce as part of *x*.

- Subsequently, function *G* maps the random nonces to their hashes.

the parties choose the same random nonce twice, has probability <= $\binom{q_{\text{total}}}{2} 2^{-\lambda}$

G(.) maps those values to the same one (collision) <= $\binom{q_{\text{total}}}{2} 2^{-\lambda}$

Backbone Protocol Properties

Common Prefix

(informally)

If two players prune a sufficient number of blocks from their chains they will obtain the same prefix

Chain Quality

(informally)

Any (large enough) chunk of an honest player's chain will contain some blocks from honest players

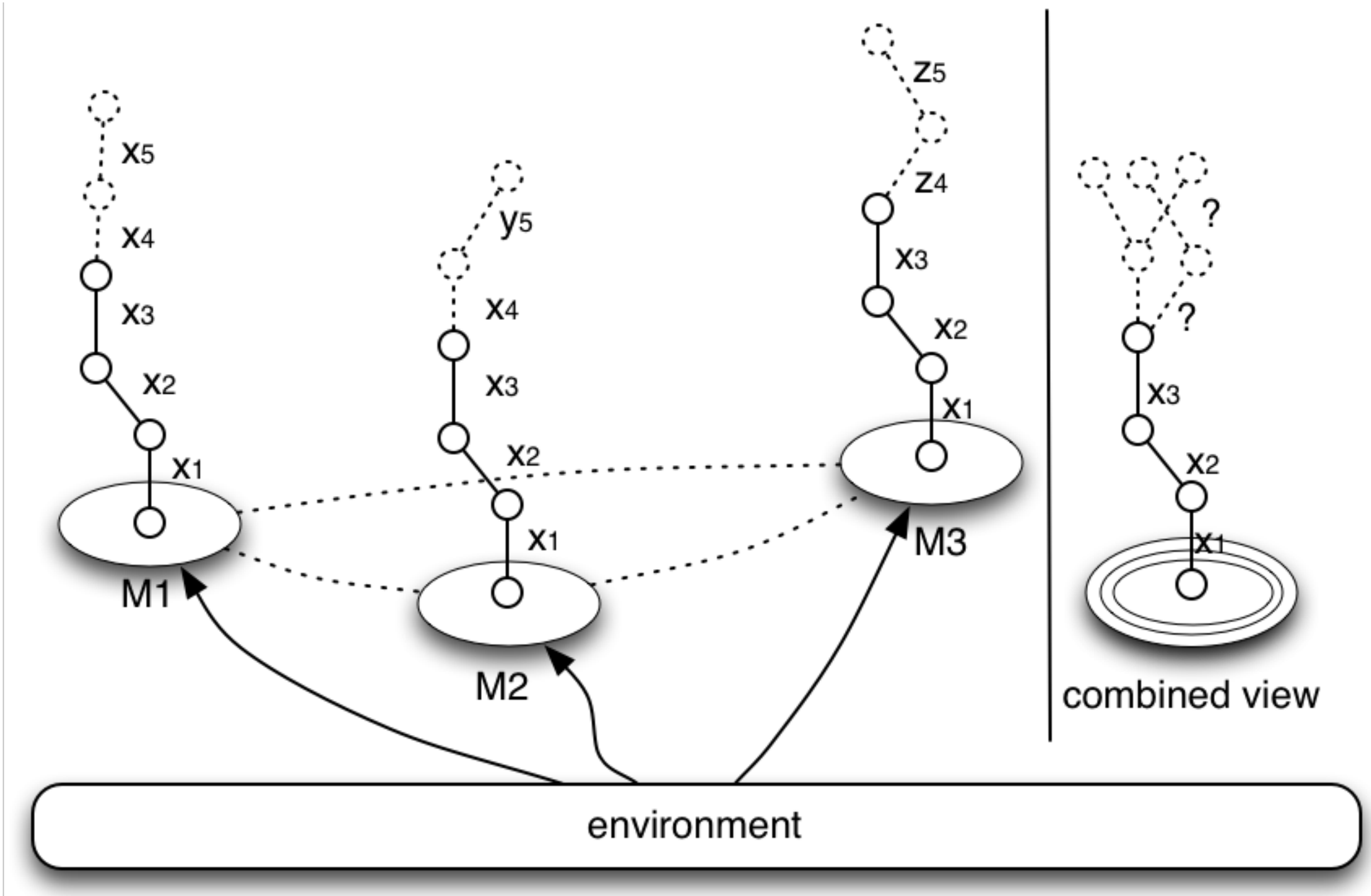
Chain Growth

(informally)

the chain of any honest player grows at least at a steady rate - the chain speed coefficient

Based on work of [GKL14, KP15]

CP: will honest players converge?

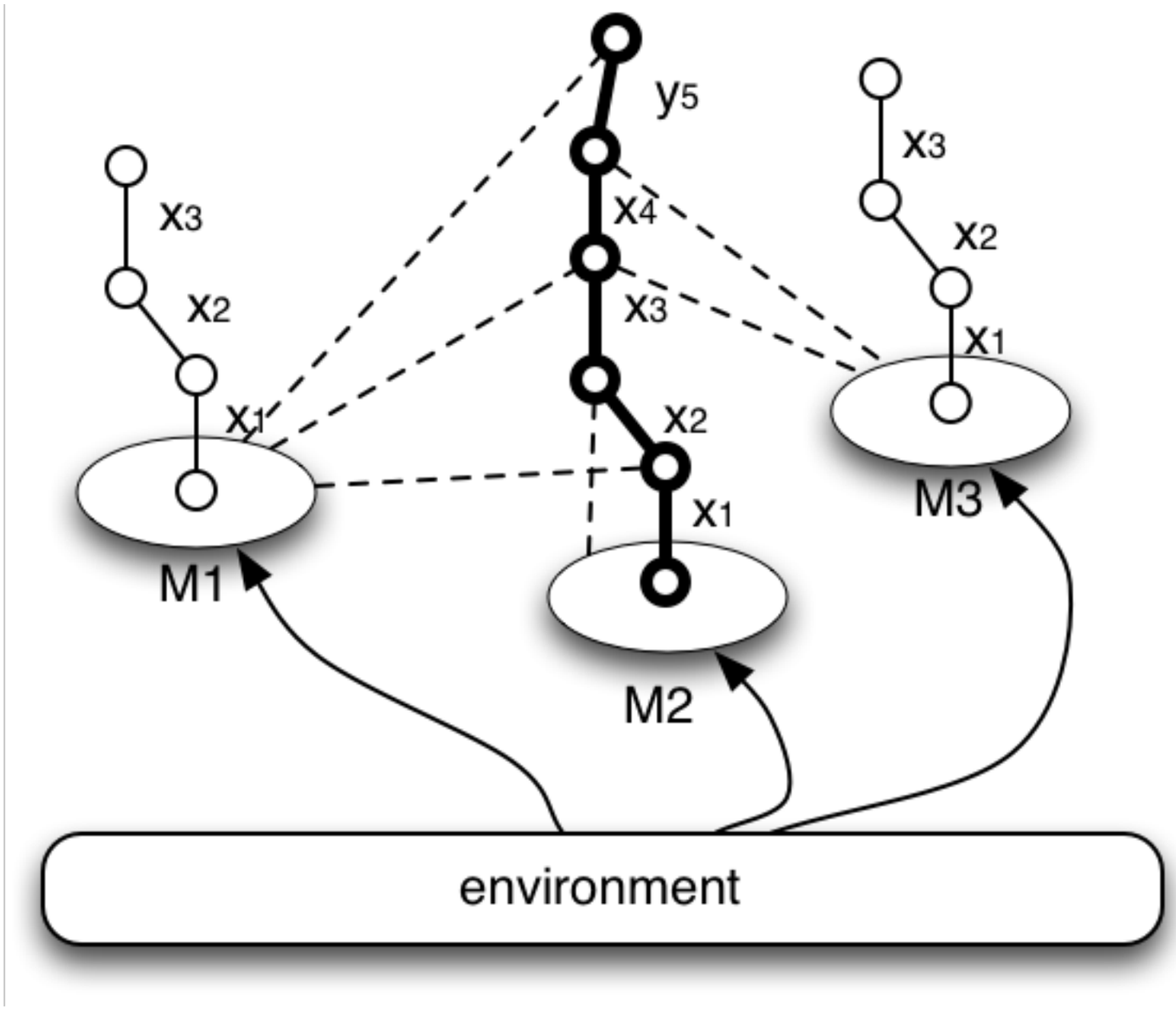


Common Prefix

$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \ \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

History : “strong” version of common prefix [K-Panagiotakos15,16].
Originally [GKL14] considered the case with $r_1 = r_2$
Note that [GKL14] did not black-box reduce persistence to CP.
[PassSeemanShelat16] highlighted this and proposed “consistency”
to provide a black-box reduction. For the same goal,
[K-P15,16] strengthened common prefix, as shown above.

CQ: are honest blocks going to be adopted by the parties?



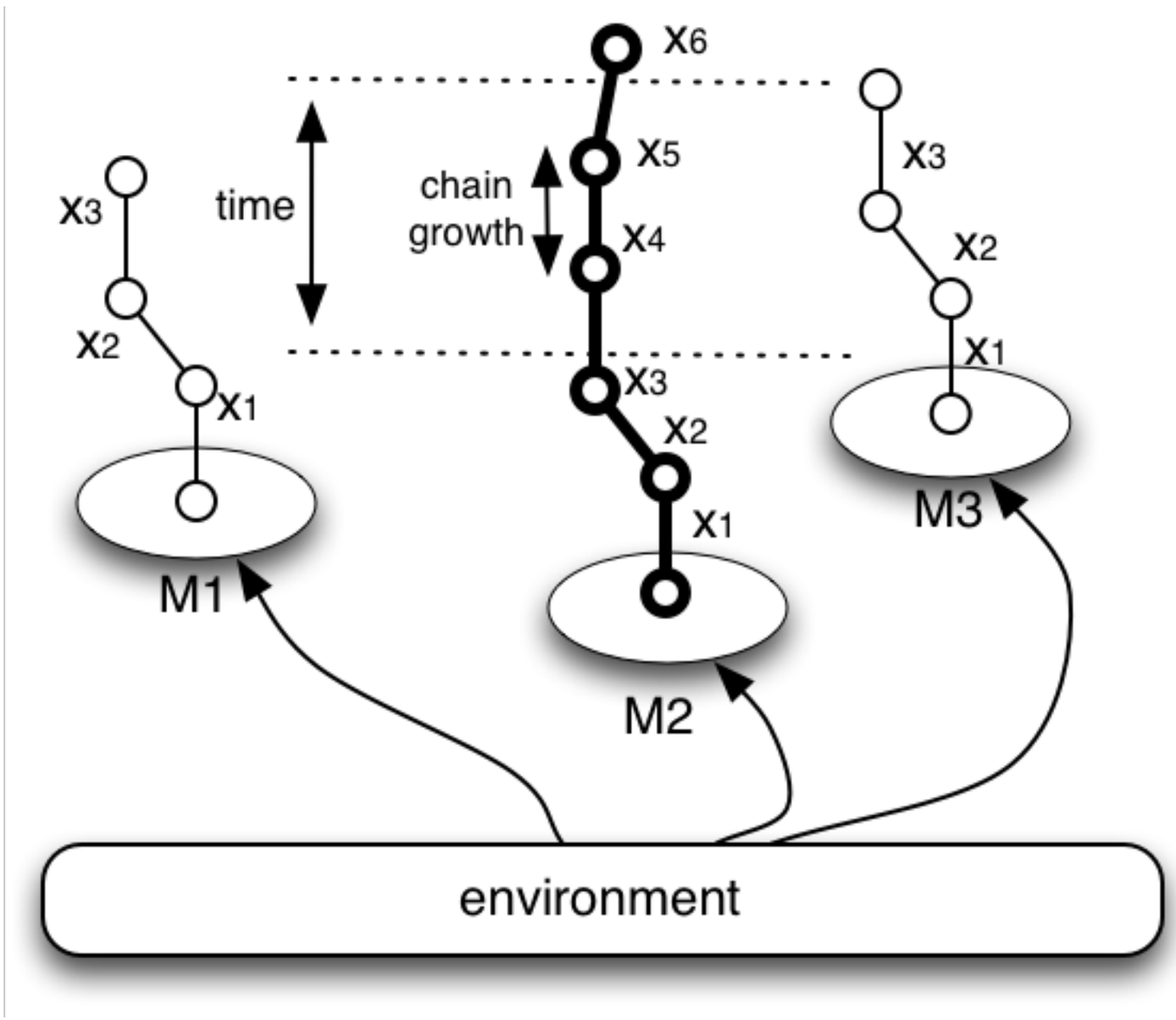
Chain Quality

Parameters $\mu \in (0,1), k \in \mathbb{N}$

The proportion of blocks in any k-long subsequence produced by the adversary is less than μk

History : Property introduced in [GKL14]

Chain Growth: does the chain grow?



Chain Growth

Parameters $\tau \in (0,1), s \in \mathbb{N}$

$\forall r_1, r_2$ honest player P with chains $\mathcal{C}_1, \mathcal{C}_2$

$$r_2 - r_1 \geq s \implies |\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau s$$

History : Property introduced in [KP15], while it was implicit in [GKL14] (proven a related lemma but never given name).

Proof strategy

1. Define the notion of *typical execution.*

2. Argue that typical executions have with overwhelming probability.

3. Prove CG, CP, CQ

4. Derive persistence and liveness.

Notations, (1)

- Let *S* a set of consecutive rounds.
- *X*(*S*) = number of *successful rounds.*
- *Y*(*S*) = number of *uniquely successful rounds*.
- *Z*(*S*) = total number of PoWs computed during *S*.

Notations, (2)

probability
at least one honest
party finds a POW
in a round

$$f = 1 - (1 - \frac{T}{2^\kappa})^{q(n-t)}$$

$$p = q/2^\kappa$$

Observe

$$f = pT(n-t) - (\frac{T}{2^\kappa})^2(\ldots) \approx pT(n-t)$$

---

probability
exactly one honest
party finds a PoW

$$\geq pT(n-t)(1 - \frac{T}{2^\kappa})^{q(n-t)-1} > (1-f)pT(n-t)$$

Expectations

Easy from linearity :

$$E[X(S)] \approx pT(n-t)|S|$$

$$E[Y(S)] > (1-f)E[X(S)]$$

$$E[Z(S)] = pTt|S|$$

Suppose now that $\frac{n-t}{t} > 1+\delta$

It follows

$$E[X(S)] > (1+\delta)E[Z(S)]$$

$$E[Y(S)] > (1+\delta)(1-f)E[Z(S)]$$

Typical Executions, (1)

- Let κ be the security parameter.
- A polynomial in κ execution is typical with parameter ε if for any set of rounds $S, |S| = \Omega(\kappa)$

$$X(S) > (1 - \epsilon)E[X(S)]$$
$$Y(S) > (1 - \epsilon)E[Y(S)]$$
$$Z(S) < (1 + \epsilon)E[Z(S)]$$

- No collisions, or predictions take place against *H*(.)

Typical Executions, (2)

Theorem. Typical executions happen almost always

Proof

Case 1. Suppose that

$$\exists S : X(S) \leq (1-\epsilon)E[X(S)]$$
$$\vee Y(S) \leq (1-\epsilon)E[Y(S)]$$
$$\vee Z(S) \geq (1+\epsilon)E[Z(S)]$$

X,Y,Z the binomial distribution so we can show with overwhelming probability in κ via a Chernoff bound.

Case 2. There is a collision or prediction for the hash function.

Follows from RO assumption + input entropy assumption.

Typical Executions, (2)

- Chernoff bounds

X is a binomial distribution $\quad \mu = E[X]$

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2 \mu/2}$$

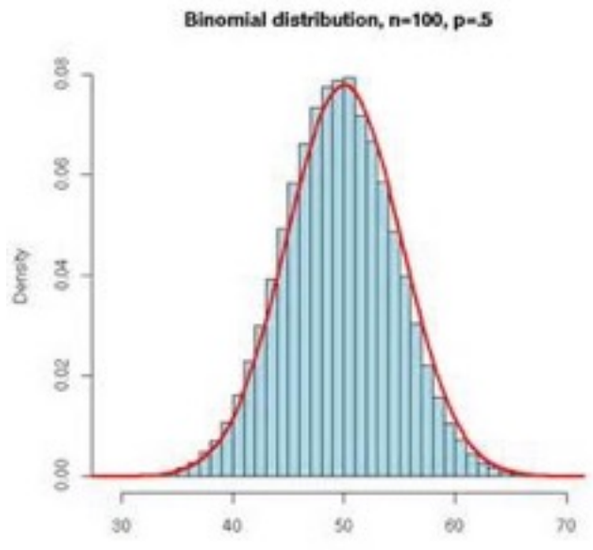
$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2 \mu/3}$$

E.g., sequence of n independent Bernoulli Trials

$$X = \sum_{i=1}^{n} X_i$$

$$X_i \in \{0,1\}, \Pr[X_i = 1] = p$$

$$\mu = np$$

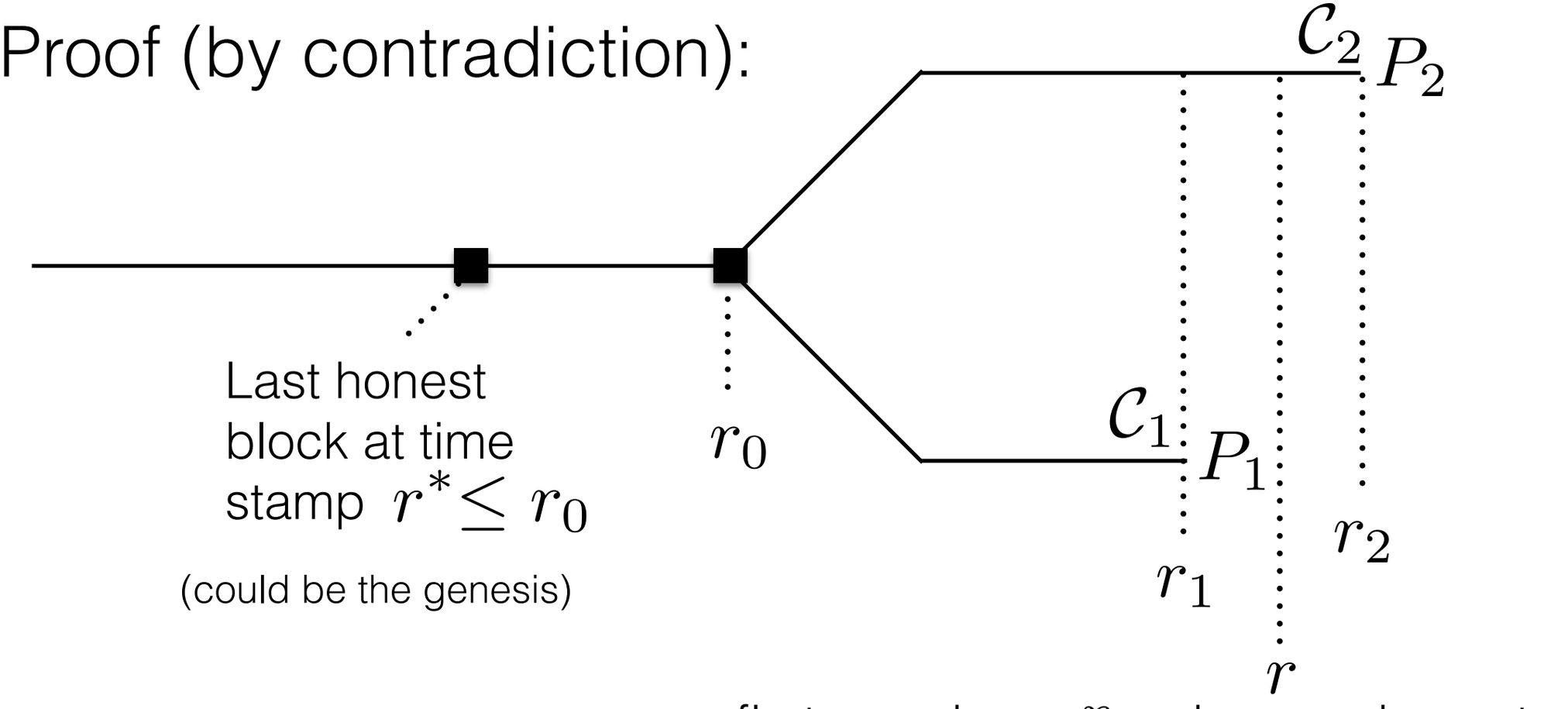


Common Prefix, (1)

Recall:

$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

Proof (by contradiction):



$k = \Omega(\kappa)$

first round >= r_1 where an honest party has a chain $\mathcal{C}_2'$ with $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2'$

Common Prefix, (2)

At round $r-1$

All honest parties have a chain $\mathcal{C}_i^{r-1}$ with $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_i^{r-1}$

At the end of round $r-1$ chain $\mathcal{C}_2'$ is transmitted for which we know that

$$\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2'$$

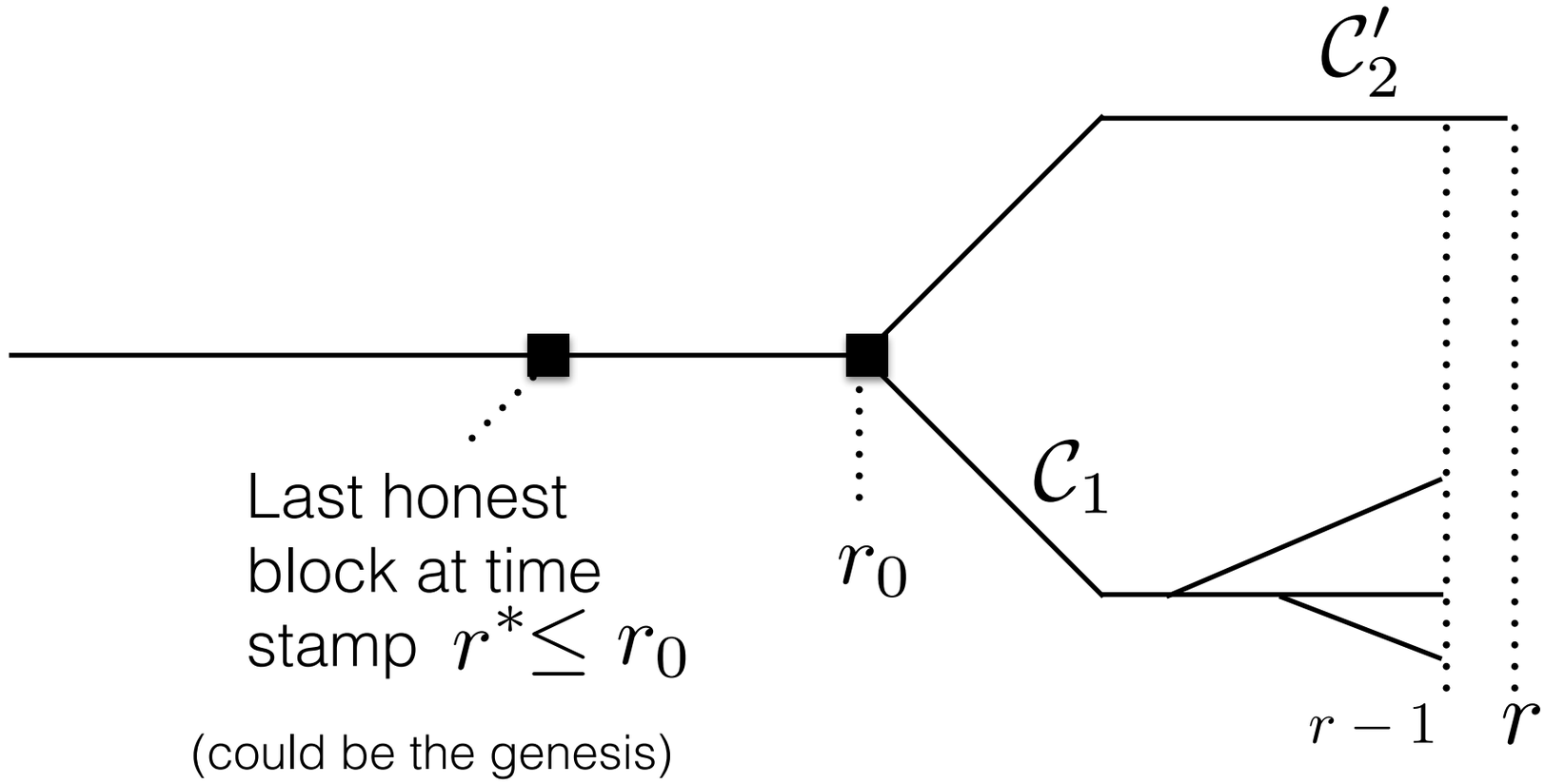
[by assumption]

$$|\mathcal{C}_2'| \geq |\mathcal{C}_1|$$

[by the fact that $\mathcal{C}_2'$ will be accepted at round r by an honest party while at least one honest party at round $r_1 \leq r$ possessed chain $\mathcal{C}_1$]

Common Prefix, (3)

Consider the set of rounds $S = \{r^* + 1, \ldots, r - 1\}$



Common Prefix, (4)

- Consider a uniquely successful round in $S = \{r^* + 1, \ldots, r - 1\}$
  - **lemma #1.** If a block created in a uniquely successful round at position *m* in a blockchain, then no other honest player will ever mine at position *m in any blockchain.*

- Therefore each uniquely successful round in *S* creates a block that must be matched by another block of the adversary

  - **lemma #2.** Such adversarial block should also be created within *S* (by the choice of *r** and typicality)

Common Prefix, (5)

- It follows that $Z(S) \geq Y(S)$ and $|S| = \Omega(\kappa)$

By typicality: $Y(S) > (1-\epsilon)E[Y(S)]$

$Z(S) < (1+\epsilon)E[Z(S)]$

recall:

$E[X(S)] \approx pT(n-t)|S|$

$E[Y(S)] > (1-f)E[X(S)]$

$E[Z(S)] = pTt|S|$

$$\frac{n-t}{t} > 1+\delta$$

$$(1+\epsilon)pTt|S| > (1-\epsilon)(1-f)pT(n-t)|S|$$

$$\iff \frac{n-t}{t} < \frac{1+\epsilon}{(1-\epsilon)(1-f)}$$

from which we obtain a contradiction as long as

$$1+\delta > \frac{1+\epsilon}{(1-\epsilon)(1-f)}$$

which is implied by: $\delta > 2\epsilon + f$

Common Prefix, (5)

- It follows that $Z(S) \geq Y(S)$ and $|S| = \Omega(\kappa)$

By typicality: $Y(S) > (1-\epsilon)E[Y(S)]$

$Z(S) < (1+\epsilon)E[Z(S)]$

recall:

$E[X(S)] \approx pT(n-t)|S|$

$E[Y(S)] > (1-f)E[X(S)]$

$E[Z(S)] = pTt|S|$

$$\frac{n-t}{t} > 1+\delta$$

$$(1+\epsilon)pTt|S| > (1-\epsilon)(1-f)pT(n-t)|S|$$

$$\iff \frac{n-t}{t} < \frac{1+\epsilon}{(1-\epsilon)(1-f)}$$

from which we obtain a contradiction as long as

$$1+\delta > \frac{1+\epsilon}{(1-\epsilon)(1-f)}$$

which is implied by: $\delta > 2\epsilon + f$

QED

Chain Quality, (1)

- Consider a chain $\mathcal{C}$ of an honest party and ℓ consecutive blocks from that chain.
- The chain quality coefficient is $\mu = \frac{1}{\lambda}$ where $\frac{n-t}{t} > \lambda(1+\delta)$

Proof (by contradiction)

Consider a sequence of blocks $B_u \ldots B_v$
in the chain of an honest party with $\ell = v - u + 1$

Chain Quality, (2)

- Define an expanded sequence of blocks

$$B_{u'} \ldots B_{v'} \qquad L = v' - u' + 1 \geq \ell$$

So that (or is genesis)

(1) $B_{u'}$ was produced by an honest party at round r_1

(2) $B_{v'}$ was accepted by an honest party at round r_2

(such extension is well defined) $S = \{r_1, \ldots, r_2\}$

$x =$ number of blocks produced by honest parties

For the sake of contradiction: $x < (1 - \mu)\ell$

Chain Quality, (3)

- **Lemma #1**. Because of typicality all the *L* blocks are computed within $S = \{r_1, \ldots, r_2\}$

- **Lemma #2**. Because of the choice of *S*, we have that $L \geq X(S)$ (otherwise no honest party would accept $B_{v'}$)

- Using the above and $x < (1-\mu)\ell$ we have :

$$Z(S) \geq L - x \geq \mu L \geq \mu X(S)$$

Chain Quality, (4)

- It follows that $Z(S) \geq \mu X(S)$ and $|S| = \Omega(\kappa)$

By typicality: $X(S) > (1-\epsilon)E[X(S)]$

$Z(S) < (1+\epsilon)E[Z(S)]$

recall:

$E[X(S)] \approx pT(n-t)|S|$

$E[Y(S)] > (1-f)E[X(S)]$

$E[Z(S)] = pTt|S|$

$\frac{n-t}{t} > \lambda(1+\delta)$

$\lambda(1+\epsilon)pTt|S| > (1-\epsilon)pT(n-t)|S|$

$\iff \frac{n-t}{t} < \lambda\frac{1+\epsilon}{1-\epsilon}$

from which we obtain a contradiction as long as

$1+\delta > \frac{1+\epsilon}{1-\epsilon}$

which is implied by:

$\delta > 2\epsilon$

Chain Quality, (4)

- It follows that $Z(S) \geq \mu X(S)$ and $|S| = \Omega(\kappa)$

By typicality: $X(S) > (1-\epsilon)E[X(S)]$

$Z(S) < (1+\epsilon)E[Z(S)]$

recall:

$E[X(S)] \approx pT(n-t)|S|$

$E[Y(S)] > (1-f)E[X(S)]$

$E[Z(S)] = pTt|S|$

$\frac{n-t}{t} > \lambda(1+\delta)$

$\lambda(1+\epsilon)pTt|S| > (1-\epsilon)pT(n-t)|S|$

$\iff \frac{n-t}{t} < \lambda\frac{1+\epsilon}{1-\epsilon}$

from which we obtain a contradiction as long as

$1+\delta > \frac{1+\epsilon}{1-\epsilon}$

which is implied by:

$\delta > 2\epsilon$

QED

Chain Growth, (1)

- Consider a chain $\mathcal{C}$ of an honest party
- The chain growth coefficient is $\tau = (1 - \epsilon)f$
  $s = \Omega(\kappa)$

  Proof (direct)

  Observe that with any successful round the chain of the honest parties grows by a block (independently of the adversarial strategy)

Chain Growth, (2)

- In $s = |S|$ rounds, we have an expectation of

$$E[X(S)] \approx pT(n-t)|S| \quad \text{blocks}$$

- Due to typicality: $X(S) > (1-\epsilon)E[X(S)]$

Thus, we will obtain $(1-\epsilon)pT(n-t)s$ blocks

$$= (1-\epsilon)fs$$

Chain Growth, (2)

- In $s = |S|$ rounds, we have an expectation of

$$E[X(S)] \approx pT(n-t)|S| \quad \text{blocks}$$

- Due to typicality: $X(S) > (1-\epsilon)E[X(S)]$

Thus, we will obtain $(1-\epsilon)pT(n-t)s$ blocks

$$= (1-\epsilon)fs$$

QED

Proving our objective

"The bitcoin backbone implements a robust transaction ledger."

Assumptions : (1) typical executions with error ϵ $\quad \delta > 2\epsilon + f$

(2) $\dfrac{n-t}{t} > \lambda(1+\delta)$ $\quad f \approx pT(n-t)$

Part 1 : ***Persistence***

Assume persistence **fails**, i.e.,
there is a transaction reported as stable s_1
by an honest player P_1 at round r_1
but at round $r_2 \geq r_1$ P_2
is reported as stable by honest player
in a different position

Proving our objective, (2)

Given the condition

it should hold that the chains $\mathcal{C}_1, \mathcal{C}_2$ of P_1, P_2 satisfy

$\mathcal{C}_1^{\lceil k}$ contains *tx* at round r_1

$\mathcal{C}_2^{\lceil k}$ contains *tx* at round $r_2 \geq r_1$

but in a different position, thus

$\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2$ *which violates CP*

Proving our objective, (3)

"The bitcoin backbone implements
a robust transaction ledger."

Part 2 : ***Liveness*** with parameter $u = \frac{1}{(1-\epsilon)f(1-\frac{1}{\lambda})}$

Consider a transaction transmitted for u rounds, we examine what happens at the onset of the next round.

Proving our objective, (4)

Given the chain growth
there will be τu blocks in each honest party chain
of those, $(1 - \frac{1}{\lambda})\tau u$ will originate to an honest

party. Given the choice of u
we have that this is at least one, and hence
it will include the transaction tx.

Proving our objective, (4)

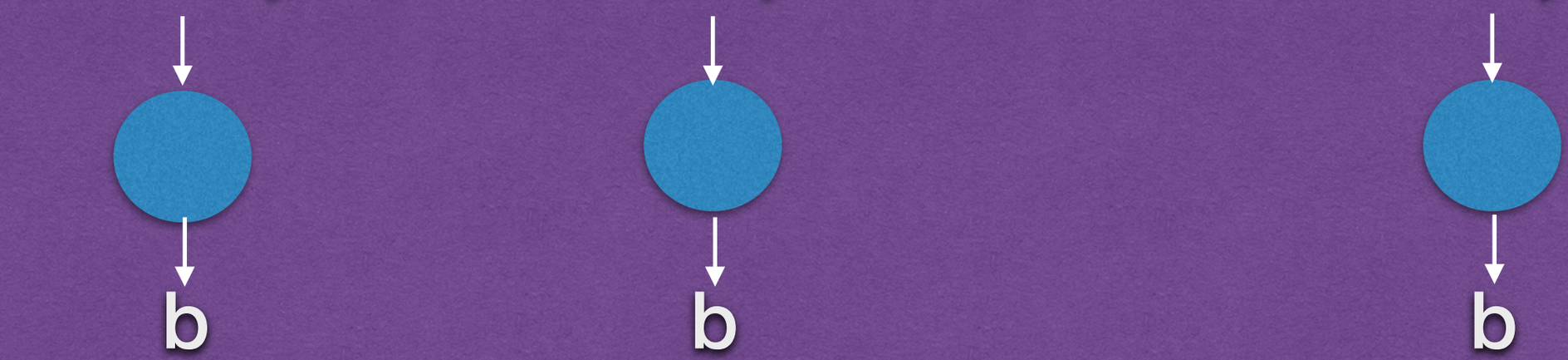
Given the chain growth
there will be τu blocks in each honest party chain
of those, $(1 - \frac{1}{\lambda})\tau u$ will originate to an honest

party. Given the choice of u
we have that this is at least one, and hence
it will include the transaction tx.

QED

Recall : Consensus

implementing consensus:



Agreement = all parties output the same value

Validity = if all honest parties have the same insert bit, then this matches the output

Termination = all honest parties terminate

Applying the backbone protocol

- It is all about defining V, I, R :
  - V = validity predicate.
  - I = input function
  - R = read function

The Nakamoto “consensus protocol”

Re: Bitcoin P2P e-cash paper

Satoshi Nakamoto | Thu, 13 Nov 2008 19:34:25 -0800

```
James A. Donald wrote:
> It is not sufficient that everyone knows X. We also
> need everyone to know that everyone knows X, and that
> everyone knows that everyone knows that everyone knows X
> - which, as in the Byzantine Generals problem, is the
> classic hard problem of distributed data processing.

The proof-of-work chain is a solution to the Byzantine Generals' Problem.  I'll
try to rephrase it in that context.

A number of Byzantine Generals each have a computer and want to attack the
King's wi-fi by brute forcing the password, which they've learned is a certain
number of characters in length.  Once they stimulate the network to generate a
packet, they must crack the password within a limited time to break in and
erase the logs, otherwise they will be discovered and get in trouble.  They
only have enough CPU power to crack it fast enough if a majority of them attack
at the same time.

They don't particularly care when the attack will be, just that they all agree.
 It has been decided that anyone who feels like it will announce a time, and
whatever time is heard first will be the official attack time.  The problem is
```

https://www.mail-archive.com/cryptography@metzdowd.com/msg09997.html

Applying the backbone for consensus, (1)

Nakamoto “consensus protocol”

Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_n \rangle)$ is true if and only if it holds that $v_1 = \ldots = v_n \in \{0,1\}, \rho_1, \ldots, \rho_n \in \{0,1\}^\kappa$ where $x_i = \langle v_i, \rho_i \rangle$.
Chain reading function $R(\cdot)$ (parameterized by k)	If $V(x_C) = \text{True}$ and $\text{len}(\mathcal{C}) \geq k$, the value of $R(\mathcal{C})$ is the (unique) value v that is present in each block of $\mathcal{C}$, while it is undefined if $V(x_C) = \text{False}$ or $\text{len}(\mathcal{C}) < k$.
Input contribution function $I(\cdot)$	If $\mathcal{C} = \emptyset$ and (INSERT, v) is in the input tape then $I(st, \mathcal{C}, round, \text{INPUT}())$ is equal to $\langle v, \rho \rangle$ where $\rho \in \{0,1\}^\kappa$ is a random value; otherwise (i.e., the case $\mathcal{C} \neq \emptyset$), it is equal to $\langle v, \rho \rangle$ where v is the unique $v \in \{0,1\}$ value that is present in $\mathcal{C}$ and $\rho \in \{0,1\}^\kappa$ is a random value. The state st always remains ϵ.

It works .. but only with constant probability of success (not overwhelming)

Applying the backbone for consensus, (2)

A (1/3) “consensus protocol” (from GKL14)

Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_n \rangle)$ is true if and only if $v_1, \ldots, v_n \in \{0,1\}, \rho_1, \ldots, \rho_n \in \{0,1\}^\kappa$ where v_i, ρ_i are the values from the pair $x_i = \langle v_i, \rho_i \rangle$.
Chain reading function $R(\cdot)$ (parameterized by k)	If $V(\langle x_1, \ldots, x_n \rangle) = \text{True}$ and $n \geq 2k$, the value $R(\mathcal{C})$ is the majority bit of $v_1, \ldots, v_k$ where $x_i = \langle v_i, \rho_i \rangle$; otherwise (i.e., the case $V(\langle x_1, \ldots, x_n \rangle) = \text{False}$ or $n < 2k$) the output value is undefined.
Input contribution function $I(\cdot)$	$I(st, \mathcal{C}, round, \text{INPUT}())$ is equal to $\langle v, \rho \rangle$ if the input tape contains (INSERT, v); ρ is a random κ-bit string. The state st remains always ϵ.

It works .. but only up to 1/3 adversarial power.

Applying the backbone for transaction ledger

(from GKL14)

Content validation predicate $V(\cdot)$	$V(\langle x_1, \ldots, x_m \rangle)$ is true if and only if the vector $\langle x_1, \ldots, x_m \rangle$ is a valid ledger, i.e., $\langle x_1, \ldots, x_m \rangle \in \mathcal{L}$.
Chain reading function $R(\cdot)$	If $V(\langle x_1, \ldots, x_m \rangle) = \text{True}$, the value $R(\mathcal{C})$ is equal to $\langle x_1, \ldots, x_m \rangle$; undefined otherwise.
Input contribution function $I(\cdot)$	$I(st, \mathcal{C}, round, \text{INPUT}())$ operates as follows: if the input tape contains (INSERT, v), it parses v as a sequence of transactions and retains the largest subsequence $x' \preceq v$ that is valid with respect to $\mathbf{x}_\mathcal{C}$ (and whose transactions are not already included in $\mathbf{x}_\mathcal{C}$). Finally, $x = \text{tx}_0 x'$ where tx_0 is a neutral random nonce transaction.

it satisfies persistence and liveness with overwhelming probability as long as also digital signature security holds.

Applying the backbone for consensus, (3)

- Main obstacle (intuitively)
  the blockchain protocol does not provide sufficiently high chain quality.

- … we cannot guarantee that we have enough blocks originating from honest parties.

- How to fix this?

Applying the protocol for consensus, (4)

- The *n* parties build a ledger but now generate transactions based on POW that contain their inputs.

- Once the blockchain is long enough the parties' prune the last *k* blocks and output the majority of the values drawn from the set of transactions in the ledger.

Beware! given that POW's are used for two different tasks how do we prevent the attacker from shifting its hashing power from the one to the other?

2-for-1 POWs

parallel composition of POW protocols

$h \leftarrow G(s, x)$
if $H(h, ctr) < T \ldots$

$h' \leftarrow G(s', x')$
if $H(h', ctr') < T' \ldots$

given $((s, x), ctr)$
verify:
$H(G(s, x), ctr) < T$
given $((s', x'), ctr')$
verify:
$H(G(s', x'), ctr') < T'$

2-for-1 POWs

parallel composition of POW protocols

$h \leftarrow G(s, x)$
if $H(h, ctr) < T \ldots$

$h' \leftarrow G(s', x')$
if $H(h', ctr') < T' \ldots$

given $((s, x), ctr)$
verify:
$H(G(s, x), ctr) < T$
given $((s', x'), ctr')$
verify:
$H(G(s', x'), ctr') < T'$

Not Secure

2-for-1 POWs

parallel composition of POW protocols

$h \leftarrow G(s, x)$
if $H(h, ctr) < T \ldots$

$h' \leftarrow G(s', x')$
if $H(h', ctr') < T' \ldots$

$h \leftarrow G(s, x)$
$h' \leftarrow G(s', x')$
$w \leftarrow H(h, h', ctr)$
if $w < T \ldots$
if $[w]^{\mathsf{R}} < T' \ldots$

Not Secure

given $((s, x), ctr)$
verify:
$H(G(s, x), ctr) < T$
given $((s', x'), ctr')$
verify:
$H(G(s', x'), ctr') < T'$

given
$((*, *), (s', x'), ctr')$
verify:
$[H(G(*, *), G(s', x'), ctr')]^{\mathsf{R}} < T'$

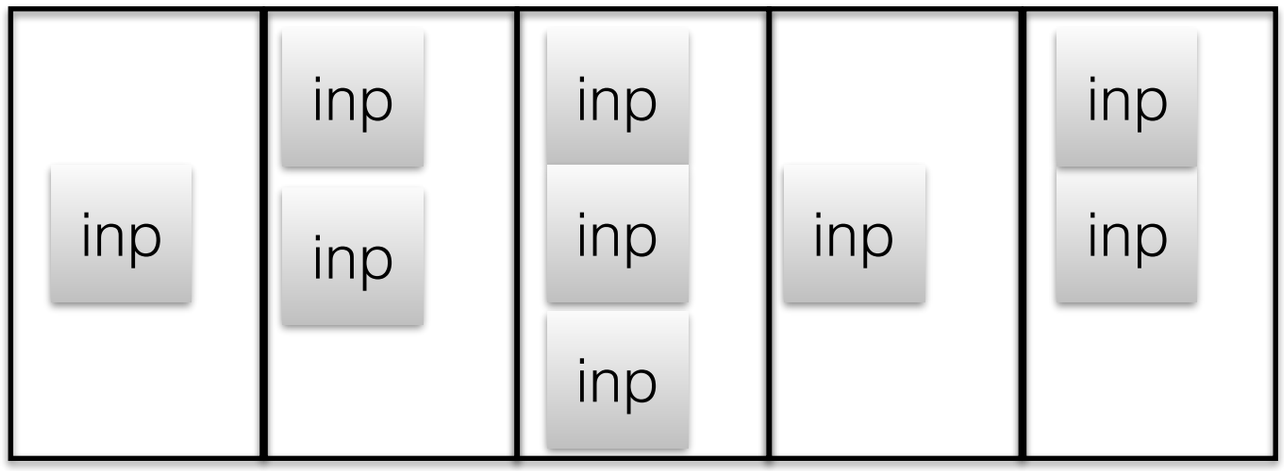
Key Lemma

- **Lemma.** Finding POW solution for either “side” of the POW protocol is an independent event.

- [note that it works only for a suitable choice of T and T’]

[GKL14] consensus protocol, (1)

parties **mine** POWs for each block (as in bitcoin backbone)



parties **mine** POWs for each input in {0,1} (input+nonce)

they keep **transmitting** POW-inputs, until they are accepted.

> Finally, after the blockchain **grows sufficiently**, they **chop** the last *k* blocks and return the **majority among unique inputs** in the (common) prefix.

[GKL14] consensus protocol, (2)

Theorem. The [GKL14] protocol using 2-for-1 POW solves consensus for honest majority.

[GKL14] consensus protocol, (2)

Theorem. The [GKL14] protocol using 2-for-1 POW solves consensus for honest majority.

Key proof idea

[GKL14] consensus protocol, (2)

Theorem. The [GKL14] protocol using 2-for-1 POW solves consensus for honest majority.

Key proof idea

recall : the output of the protocol is the **majority** bit from the inputs in the **common prefix**

[GKL14] consensus protocol, (2)

Theorem. The [GKL14] protocol using 2-for-1 POW solves consensus for honest majority.

Key proof idea

recall : the output of the protocol is the **majority** bit from the inputs in the **common prefix**

The **(minuscule) chain quality** of the protocol, given parties transmit inputs until they are accepted, it guaranteeing that they **will be included** eventually.

[GKL14] consensus protocol, (2)

Theorem. The [GKL14] protocol using 2-for-1 POW solves consensus for honest majority.

Key proof idea

recall : the output of the protocol is the **majority** bit from the inputs in the **common prefix**

The **(minuscule) chain quality** of the protocol, given parties transmit inputs until they are accepted, it guaranteeing that they **will be included** eventually.

Moreover, because each input, has a POW, the **majority** of unique POW-inputs will be **originating from honest parties** in any **sufficiently long** part of the chain (such as the common prefix)

Implications to Fairness

- In the [GKL14] consensus protocol, each set of parties' inputs is **fairly represented** (in terms of proportionality) in the blockchain. This is **not the case** in Bitcoin! (due to block withholding / selfish mining, [Eyal-Sirer14])

- Using this approach [Pass-Shi16] (and allowing the protocol to run continuously) argued a fair blockchain "fruitchain" (where rewards are allocated fairly).

[GKL16] The Dynamic setting : New Objective

- Analyze the bitcoin “backbone” [GKL14] in the *Dynamic* - Byzantine setting.
- prove it satisfies the properties of
  - common prefix
  - chain quality
  - chain growth
- Then shot it implements a **robust transaction ledger**

Dynamic Execution

- Environment creates/disables parties.
- Count the number of ‘Ready’ parties (those that are mining) in each round *r* and adversarial parties

$$n_r \text{ versus } t_r$$

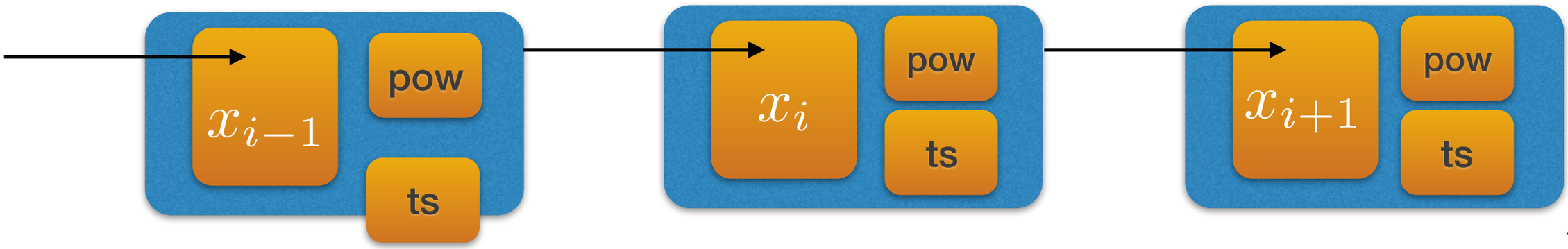
- The environment may increase or decrease the number of honest parties, but will be subject to a constraint

$$\forall S, |S| \leq s \quad \max_{r \in S} n_r \leq \gamma \min_{r \in S} n_r$$

(γ, s)-respecting environment

Protocol Intuition

Parties include inputs x_i in a block structure and find POWs [a hash value less than a target *T*]



Parties adopt chain with highest "difficulty" $\sum_i \frac{1}{T_i}$

Probability
at least one of *n* parties
finds a POW with target *T*
within one round

$$f(T,n) = 1 - \left(1 - \frac{T}{2^\kappa}\right)^{qn}$$

$$\approx pTn$$

$pT \approx$ expected # of blocks per round, per party

$$p = q \cdot 2^{-\kappa}$$

From [GKL14] : *f* should be not too small or too big.

importance of *f*

- If *f* becomes too small, parties do not do progress; chain growth goes too slow. [liveness is hurt]
- if *f* becomes too large, parties "collide" all the time; an adversary, exploiting network scheduling, can exploit that and lead them to a forked state. [agreement is hurt]

To resolve this in a dynamic environment, we may **recalculate the target** *T* to keep *f* constant $f(T, n) \approx f(T_0, n_0) = f_0$

Target Recalculation

$n_0 =$ estimation of the number of ready parties at the onset

$T_0 =$ initial target

$m =$ epoch length in blocks

$\tau =$ recalculation threshold parameter

$T =$ target in effect

$$\text{next target} = \begin{cases} \frac{1}{\tau} \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 < \frac{1}{\tau} \cdot T; \\ \tau \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 > \tau \cdot T; \\ \frac{n_0}{n} \cdot T_0 & \text{otherwise} \end{cases}$$

$\Delta =$ last epoch duration
based on block timestamps

$n = \frac{m}{pT\Delta}$ the "effective" number of parties of the epoch

… with recalculation, more attacks may be possible!

- Bahack [B13] attack:
  - Mine a chain in private with timestamps in rapid succession.
  - This will induce an artificially high target.
- Which increases variance! Concentration bounds do not hold anymore!

Proof Ideas

introduce a **measure of "goodness"** regarding the approximation that is performed on *f*

Proof Ideas

introduce a **measure of "goodness"** regarding the approximation that is performed on *f*

introduce a **notion of typicality** for executions that enables concentration arguments

Proof Ideas

introduce a **measure of "goodness"** regarding the approximation that is performed on *f*

introduce a **notion of typicality** for executions that enables concentration arguments

show (given a bound on fluctuation) that a good initial approximation on *f*, provides good Targets **per round** for a number of rounds.

Proof Ideas

introduce a **measure of "goodness"** regarding the approximation that is performed on *f*

introduce a **notion of typicality** for executions that enables concentration arguments

show (given a bound on fluctuation) that a good initial approximation on *f*, provides good Targets **per round** for a number of rounds.

show that "per round goodness" enforces **sufficiently correct timestamps**.

Proof Ideas

introduce a **measure of "goodness"** regarding the approximation that is performed on *f*

introduce a **notion of typicality** for executions that enables concentration arguments

show (given a bound on fluctuation) that a good initial approximation on *f*, provides good Targets **per round** for a number of rounds.

show that "per round goodness" enforces **sufficiently correct timestamps**.

show that sufficiently correct timestamps result in sufficiently good **next approximation for *f***

(η,θ)-Goodness

- (η,θ)-good chain: all recalculation points satisfy:

$$\eta f_0 \leq f(T, n_r) \leq \theta f_0$$

- (η,θ)-good round in execution E

$$\eta f \leq f(T_r^{\min}(E), n_r) \qquad f(T_r^{\max}(E), n_r) \leq \theta f$$

- (η,θ)-good execution E : when all rounds are (η,θ)-good in E.

(η,θ)-Goodness (2)

[recall a block of target T has difficulty 1/T]

$Q_r =$ "unique" *difficulty* calculated in round *r* using "normal" (not too hard) targets

- **Theorem.** If *r* is an (η,θ)-good round in execution *E*,

$$E[Q_r(E_{r-1})] \geq (1 - \theta f)pn_r$$

$Q_r(E_{r-1}) =$ Unique difficulty conditioned on the history of the execution so far.

Concentration

- “Per round” arguments regarding relevant random variables are not sufficient.

- We need executions with “good behavior” over a sequence of rounds, i.e., variables should be concentrated around their means.

- This is not easy anymore to get: the probabilities of the experiments performed per round depend on the history! (due to target recalculation).

Typical Execution

- If a given sequence of rounds S, is such that honest parties would have collected sufficiently many blocks (~ m) of “reasonably” big target then
  - The average unique difficulty is lower bounded.

$$\frac{1}{|S|}\Big(\sum_{r \in S} E[Q_r(E_{r-1})] - \epsilon(1-\theta f)p \sum_{r \in S} n_r \Big)$$

  - The average max difficulty is upper bounded

$$\frac{1}{|S|}\,(1+\epsilon)p \sum_{r \in S} n_r$$

Typical Execution, (2)

- The adversary has
  - (1) acquired $\ll m$ blocks of very small target
  - (2) the average sum of difficulty of higher targets is upper bounded by $\frac{1}{|S|}(1+\epsilon)p\sum_{r\in S} t_r$

Typical Execution, (3)

- and as before:
  - No hash function collisions.
  - No hash value predictions.

Typical Execution, (4)

- **Theorem.** Most poly-bounded executions are typical.

  Proof Idea. Focus on a single sequence of rounds (union bound will imply the statement).

  (case 1) Consider unique difficulty and define the martingale

$$X_0 = 0; X_r = \sum_{i \in [r]} Q_i - \sum E[Q_i \mid \mathcal{E}_{i-1}]$$

By bounding the variance of $X_r - X_{r-1}$

we can obtain a tail bound

(other cases similar)

Theorem

- There are $\gamma > 1$, $s \sim$ epoch time m/f, small δ, so that for any (γ, s) - respecting environment, common prefix and chain quality will fail with probability negligible in m, κ (for $t_r = (1-\delta)n_r$)

- Hence we obtain a robust transaction ledger.

- preconditions consistent with bitcoin parameterization, (however our bounds not tight enough for security to follows)

So… is it done?

So.. is it done?

No!

So.. is it done?

No!

- Investigate further rationality / incentive-compatibility e.g... [**K**-Koutsoupias-Kyropoulou-Tselekounis16,PS16]…

So.. is it done?

No!

- Investigate further rationality / incentive-compatibility e.g... [**K**-Koutsoupias-Kyropoulou-Tselekounis16,PS16]…

- Investigate further semi-synchronous / asynchronous behavior … [SompolinskyZohar15,PSS16]…

So.. is it done?

No!

- Investigate further rationality / incentive-compatibility e.g... [**K**-Koutsoupias-Kyropoulou-Tselekounis16,PS16]…
- Investigate further semi-synchronous / asynchronous behavior … [SompolinskyZohar15,PSS16]…
- Investigate alternative protocols, POW-based blockchain protocols, … [Sompolinsky-Zohar14,EyalGencerSirerRenesse15,**K-**Panagiotakos16,Pass-Shi16]…

So.. is it done?

No!

- Investigate further rationality / incentive-compatibility e.g... [**K**-Koutsoupias-Kyropoulou-Tselekounis16,PS16]…

- Investigate further semi-synchronous / asynchronous behavior … [SompolinskyZohar15,PSS16]…

- Investigate alternative protocols, POW-based blockchain protocols, … [Sompolinsky-Zohar14,EyalGencerSirerRenesse15,**K-**Panagiotakos16,Pass-Shi16]…

- Alternatives to proof-of-work, proof-of-stake, proof-of-space e.g.,… [**K**-Russell-David-Oliynykov16, ParkPietrzakKwonAlwenFuchsbauerGazi15]…

So.. is it done?

No!

- Investigate further rationality / incentive-compatibility e.g... [**K**-Koutsoupias-Kyropoulou-Tselekounis16,PS16]…

- Investigate further semi-synchronous / asynchronous behavior … [SompolinskyZohar15,PSS16]…

- Investigate alternative protocols, POW-based blockchain protocols, … [Sompolinsky-Zohar14,EyalGencerSirerRenesse15,**K-**Panagiotakos16,Pass-Shi16]…

- Alternatives to proof-of-work, proof-of-stake, proof-of-space e.g.,… [**K**-Russell-David-Oliynykov16, ParkPietrzakKwonAlwenFuchsbauerGazi15]…

- Understand/design multi-party protocols using blockchain, e.g.,.. [KatzMillerShi14, AndrychowiczDziembowski14,**K**ZikasZhou15]…

Proving the security of blockchain protocols

Aggelos Kiayias
aggelos.kiayias@ed.ac.uk

Based on joint work with Juan Garay, Nikos Leonardos

Gratefully acknowledging research and curriculum development support

project CODAMODA

project PANORAMIX

for supporting lecture material development