CRYPTOEXPERTS

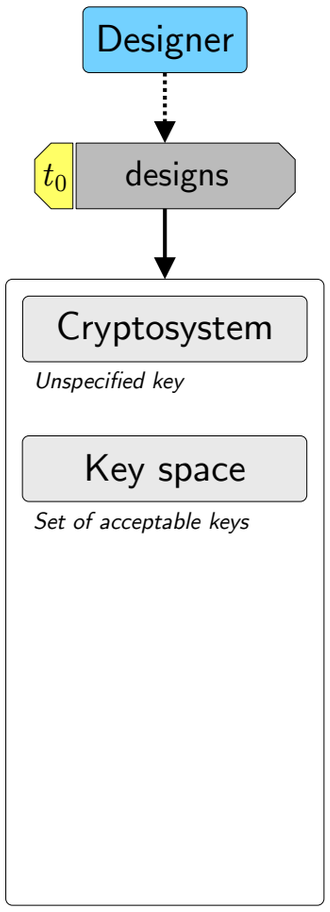
WE INNOVATE TO SECURE YOUR BUSINESS

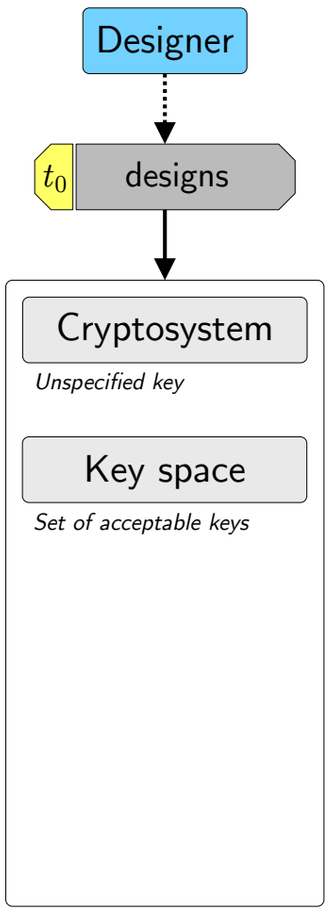
The importance of rigidity in cryptographic standards

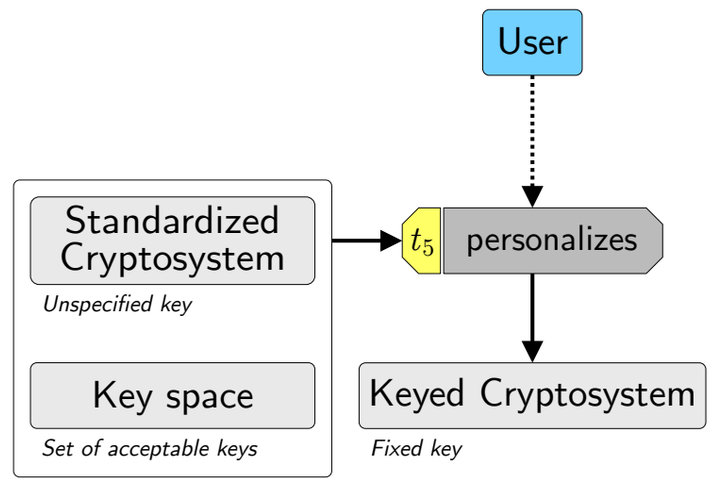
AWACS 2016

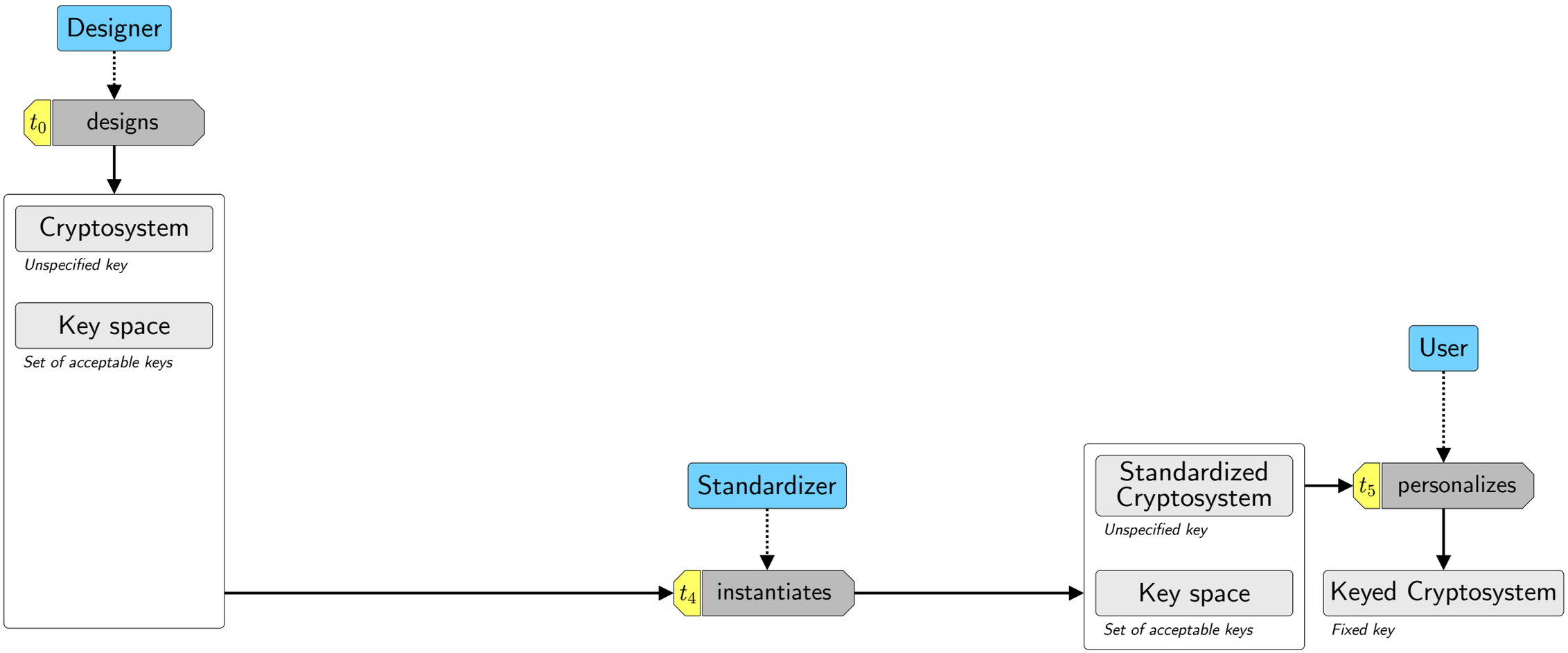
Thomas Baignères

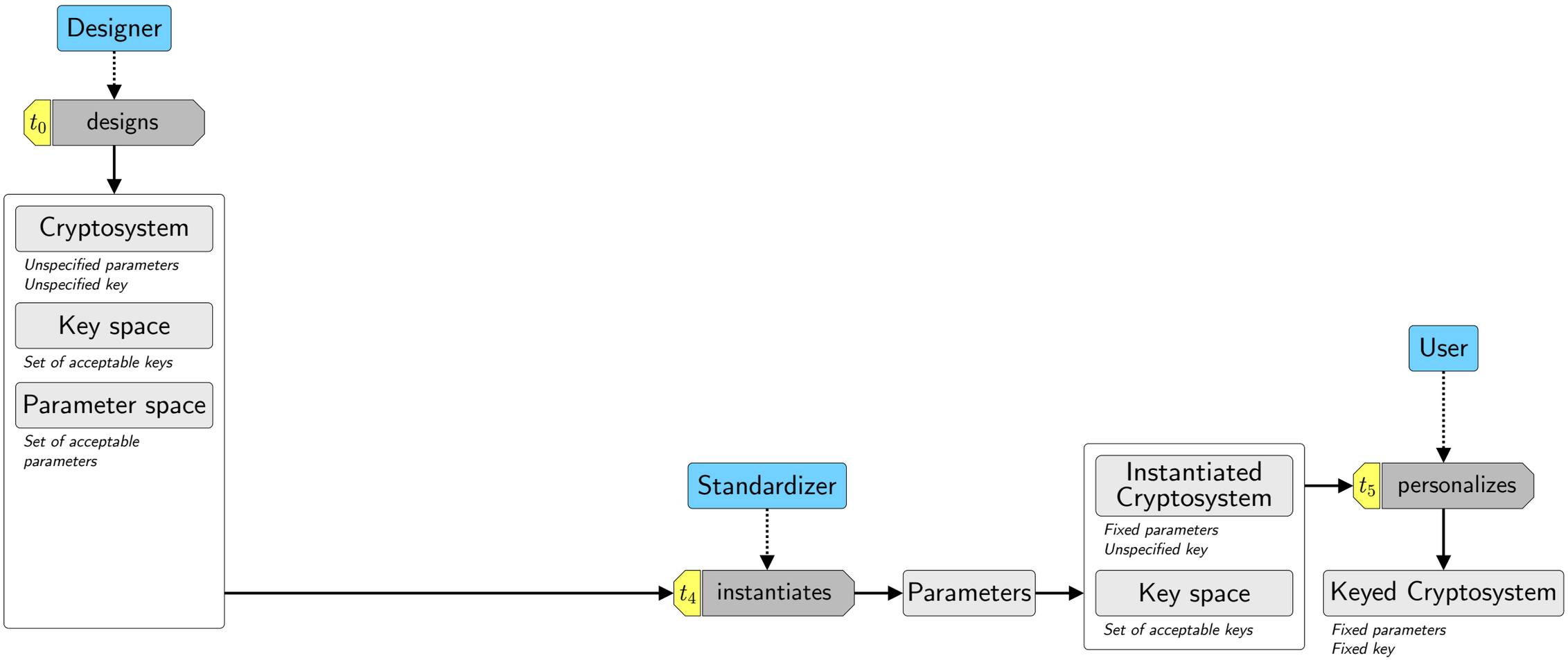
May 8, 2016

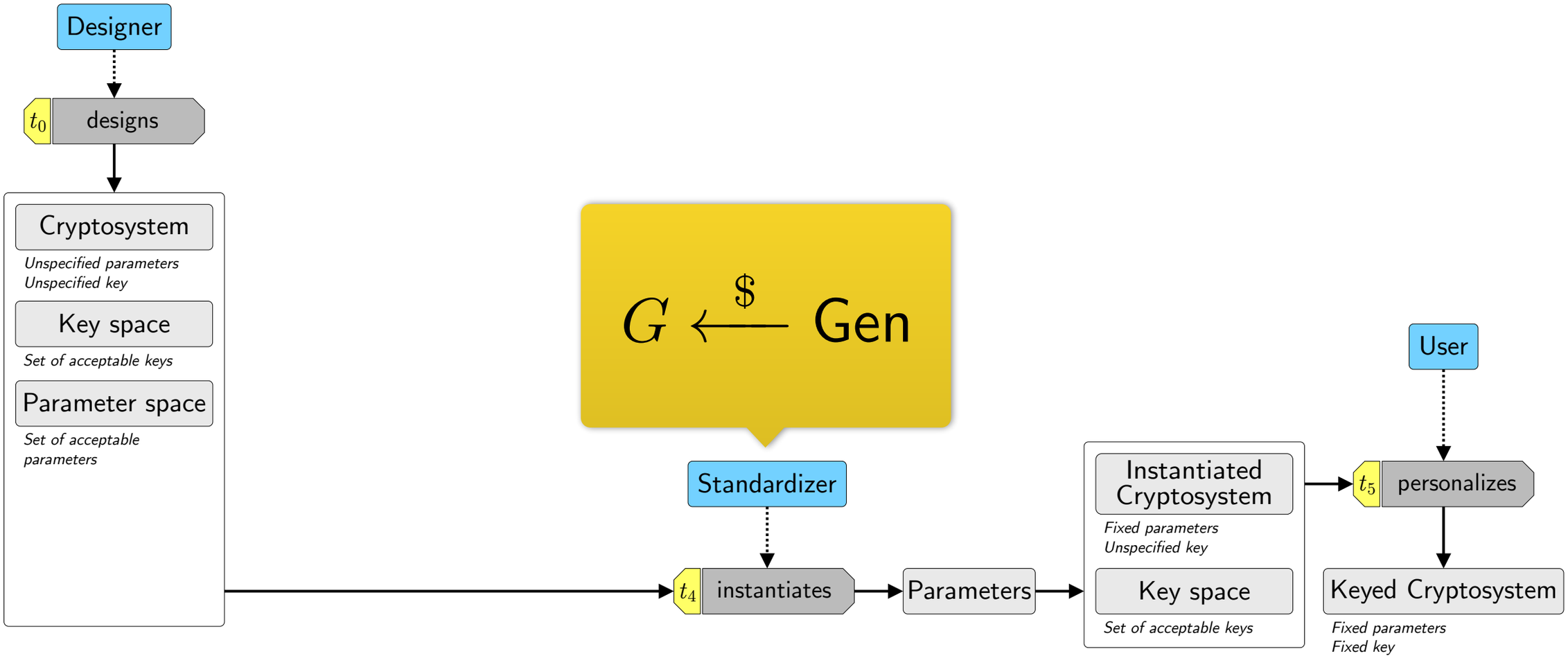

Designer
t_0
designs
Cryptosystem
Unspecified key
Key space
Set of acceptable keys


Designer
t_0
designs
Cryptosystem
Unspecified key
Key space
Set of acceptable keys

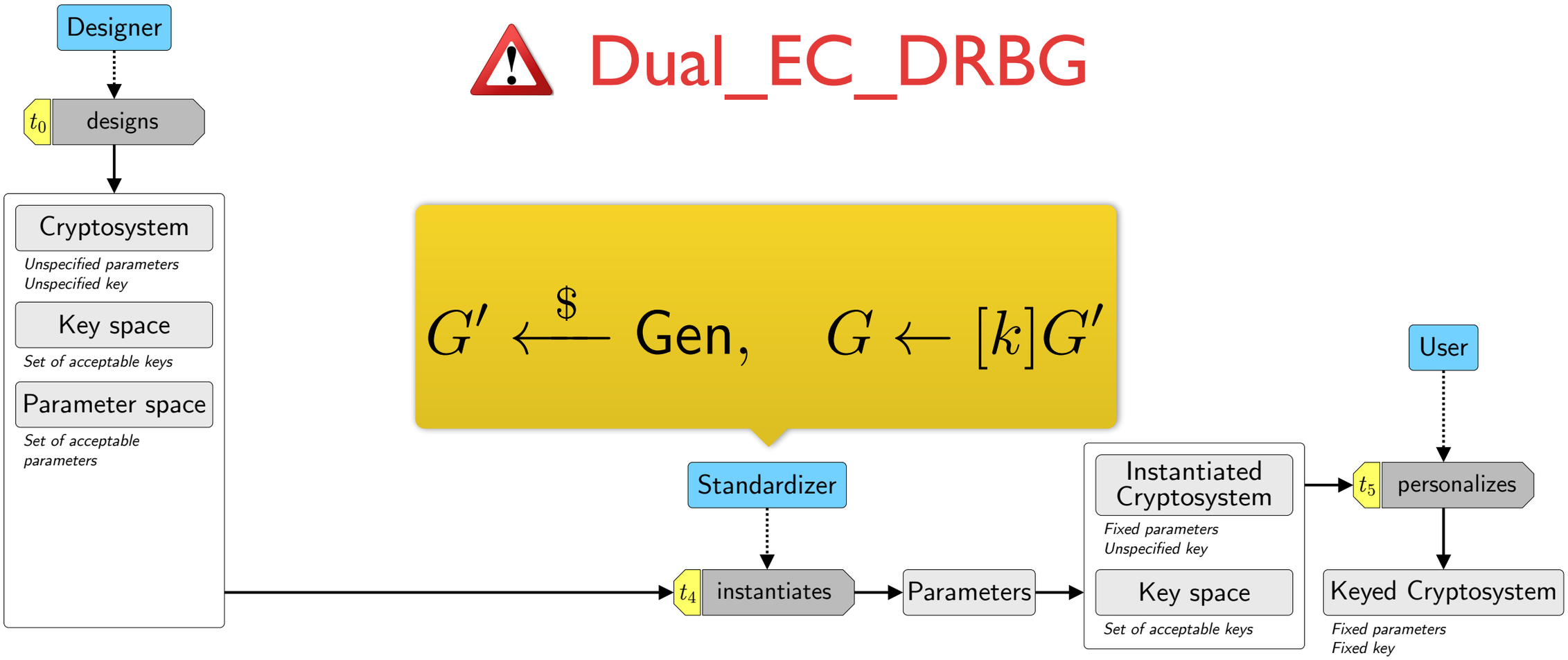

User
Standardized Cryptosystem
Unspecified key
Key space
Set of acceptable keys
t_5
personalizes
Keyed Cryptosystem
Fixed key


Designer
t_0
designs
Cryptosystem
Unspecified key
Key space
Set of acceptable keys
Standardizer
t_4
instantiates
Standardized Cryptosystem
Unspecified key
Key space
Set of acceptable keys
User
t_5
personalizes
Keyed Cryptosystem
Fixed key


Designer
t_0
designs
Cryptosystem
Unspecified parameters
Unspecified key
Key space
Set of acceptable keys
Parameter space
Set of acceptable parameters
Standardizer
t_4
instantiates
Parameters
Instantiated Cryptosystem
Fixed parameters
Unspecified key
Key space
Set of acceptable keys
User
t_5
personalizes
Keyed Cryptosystem
Fixed parameters
Fixed key


Designer
t_0
designs
Cryptosystem
Unspecified parameters
Unspecified key
Key space
Set of acceptable keys
Parameter space
Set of acceptable parameters
$G \xleftarrow{\$}$ Gen
Standardizer
t_4
instantiates
Parameters
Instantiated Cryptosystem
Fixed parameters
Unspecified key
Key space
Set of acceptable keys
User
t_5
personalizes
Keyed Cryptosystem
Fixed parameters
Fixed key

⚠ Dual_EC_DRBG


Designer
t_0
designs
Cryptosystem
Unspecified parameters
Unspecified key
Key space
Set of acceptable keys
Parameter space
Set of acceptable parameters
$G' \xleftarrow{\$} \mathsf{Gen}, \quad G \leftarrow [k]G'$
Standardizer
t_4
instantiates
Parameters
Instantiated Cryptosystem
Fixed parameters
Unspecified key
Key space
Set of acceptable keys
User
t_5
personalizes
Keyed Cryptosystem
Fixed parameters
Fixed key

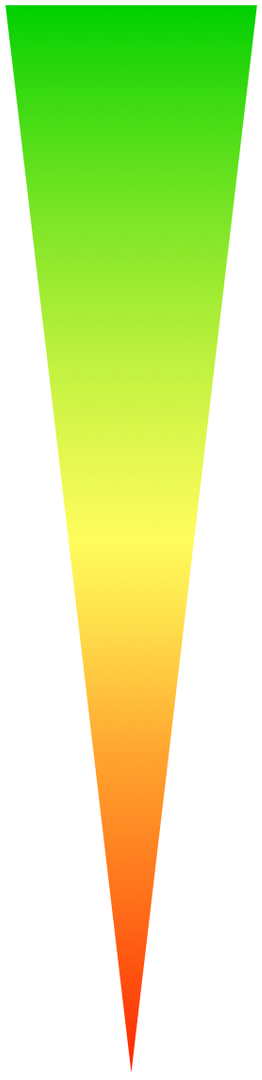
Rigidity of cryptographic parameters

Rigidity is a **subjective** caracterization of the parameter-generation process of a (well designed) cryptosystem, representing the **level of trust** one has in the security of the instantiated cryptosystem.

- From Bernstein & Lange's safecurves.cr.yp.to, a generation can be…
  - Fully rigid (good)
  - Somewhat rigid
  - Manipulatable
  - Trivially manipulatable (not good)

Rigidity of a few existing standards

Fully rigid

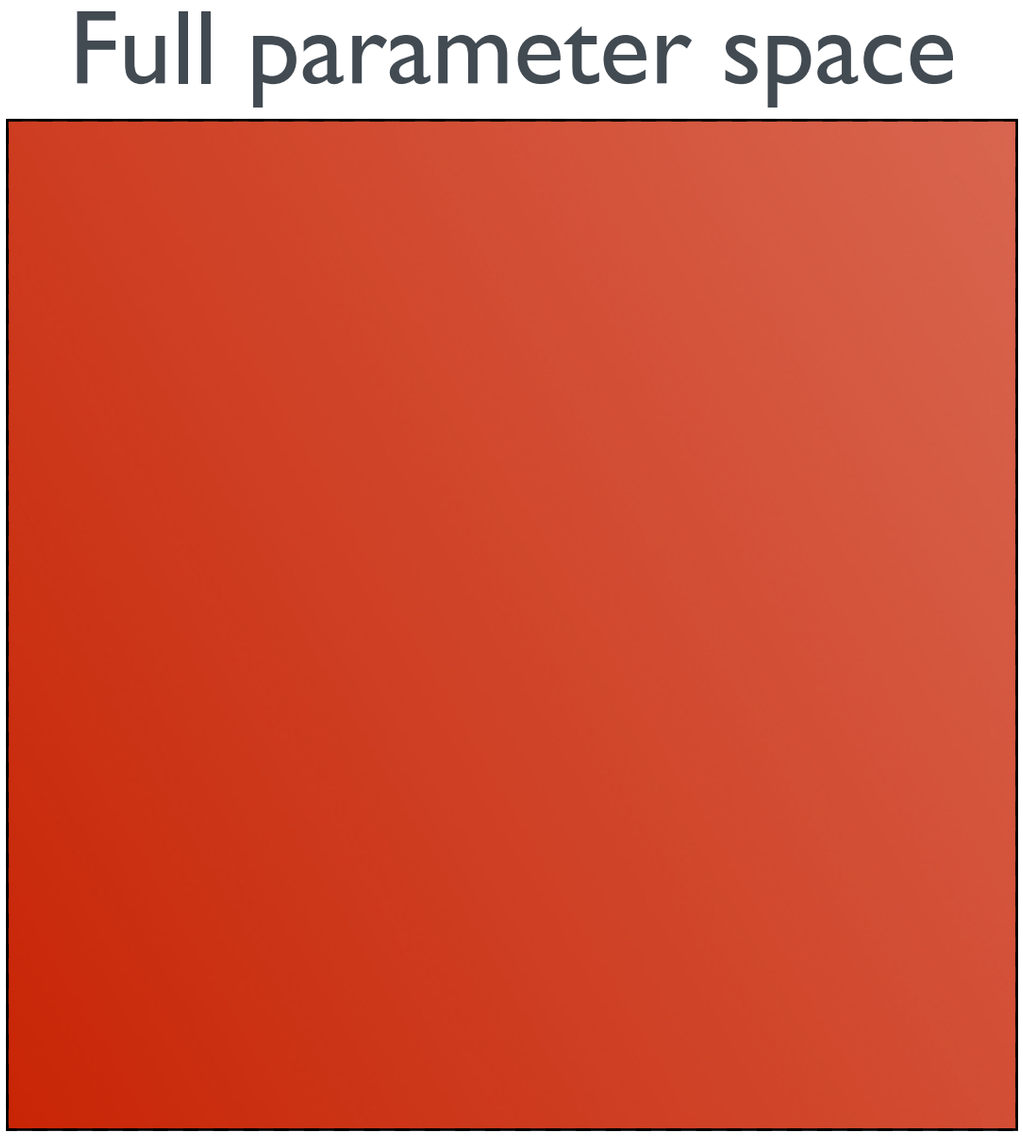


- Any standard without any parameter choice (RSA-OAEP in PKCS#1 v2)
- Curve25519 (parameters chosen to achieve optimal performances)
- AES
  - S-box choice (field representation, affine transformation)
  - Very unlikely to allow introducing a trap
  - ⋮
- Dual_EC_DRBG (withdrawn from NIST SP 800-90A)

Trivially manipulatable

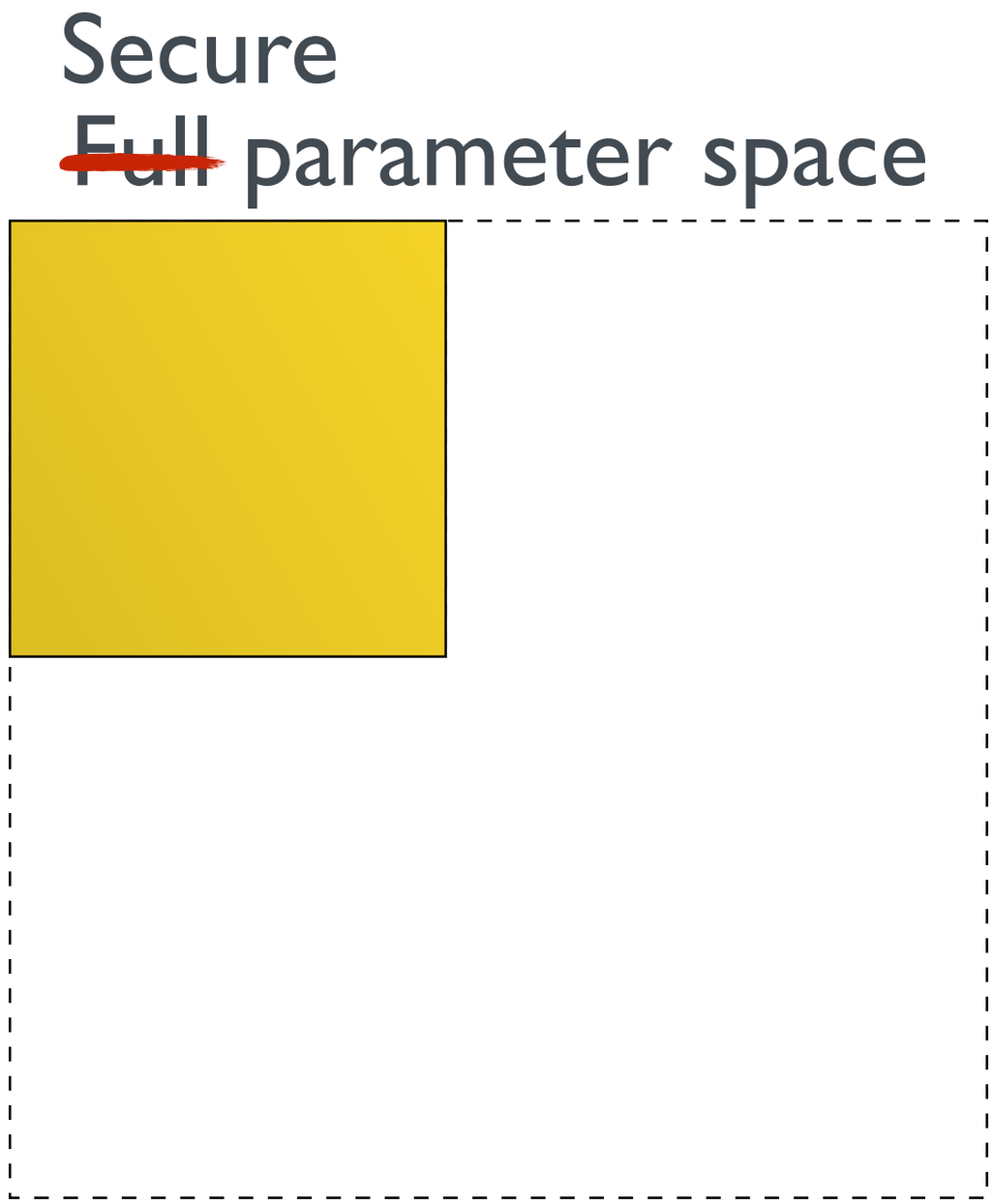
When is it hard to be rigid?

- The Designer starts with the full parameter space



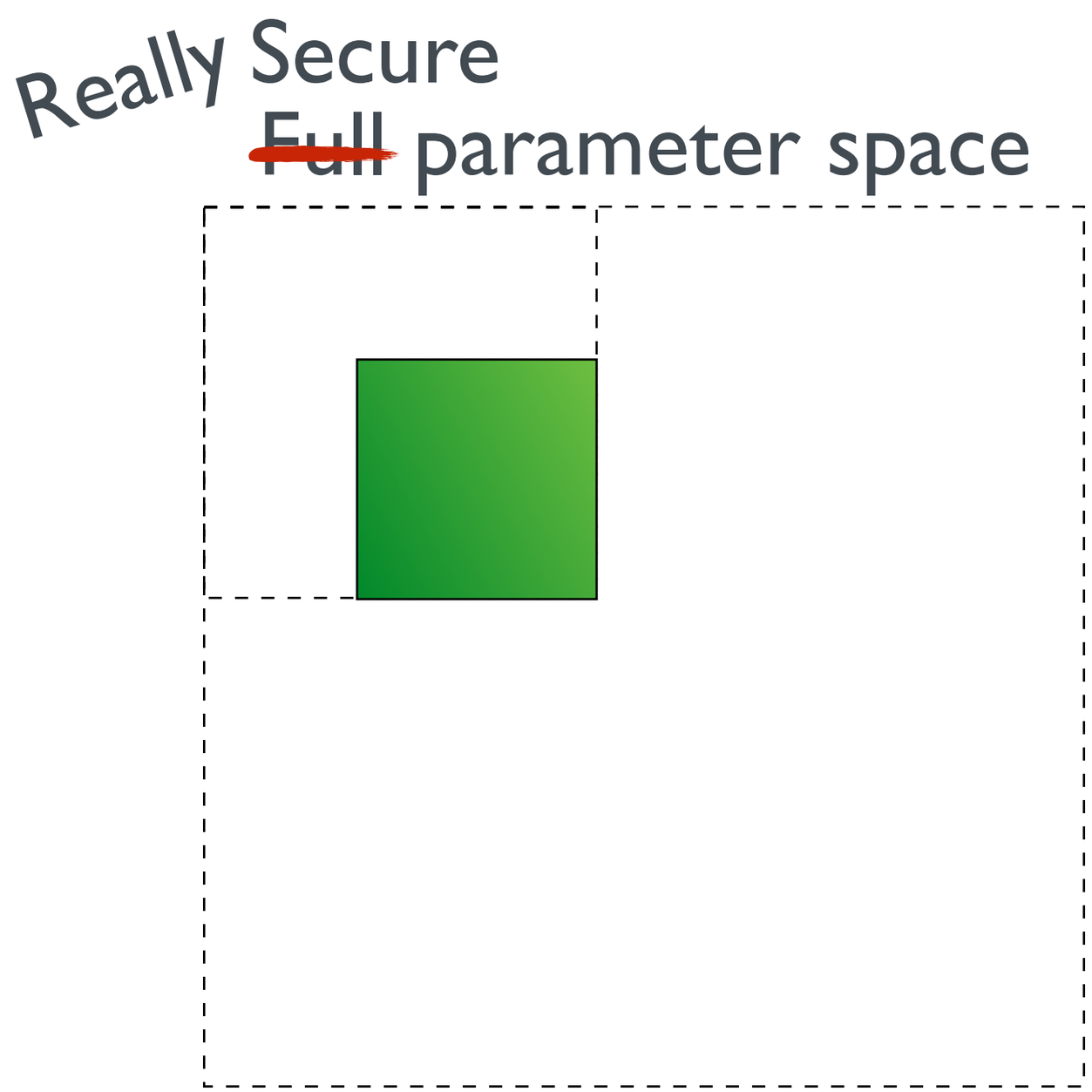
When is it hard to be rigid?

- The Designer starts with the full parameter space
- List all attacks and define security criterias



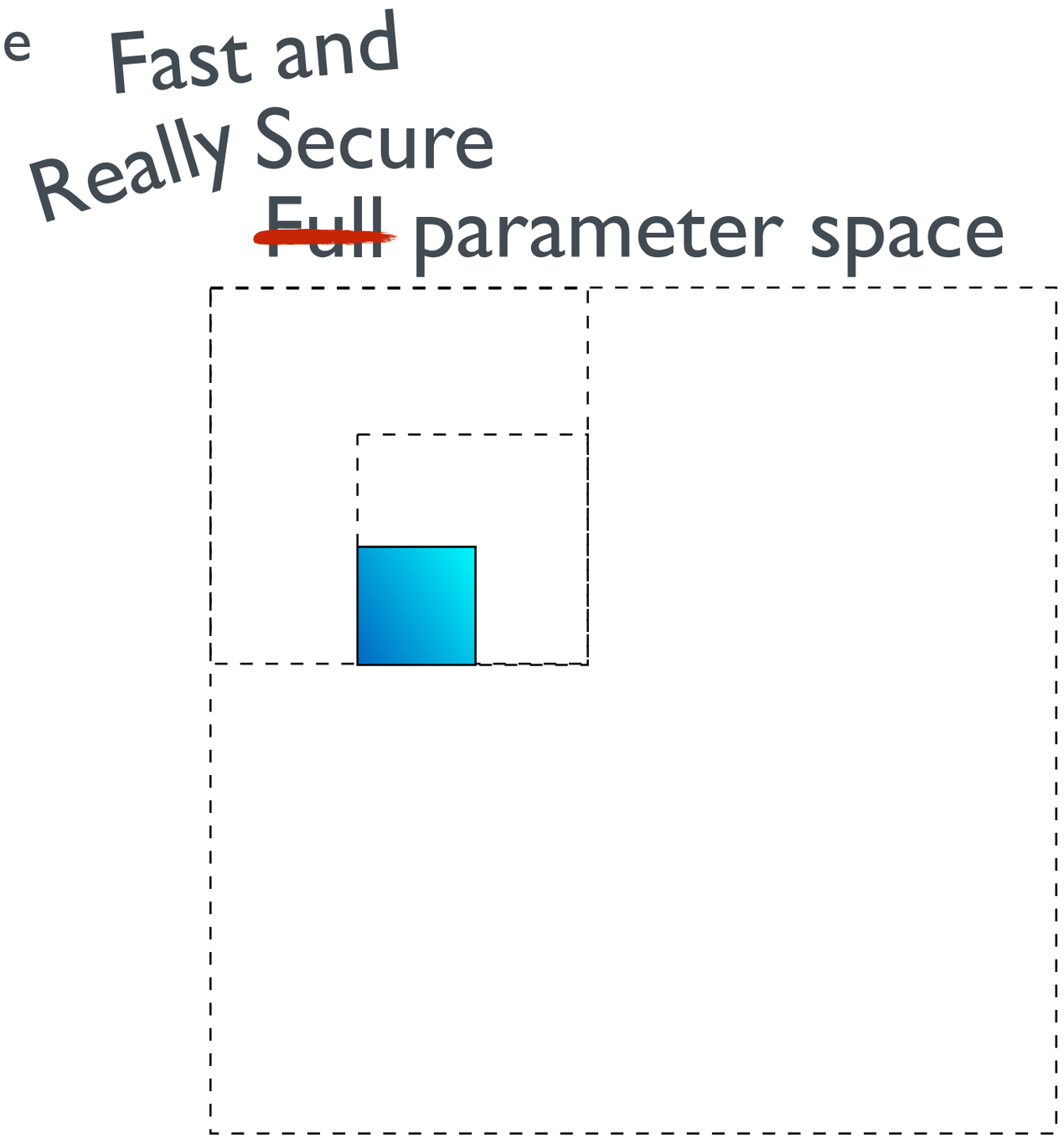
When is it hard to be rigid?

- The Designer starts with the full parameter space
- List all attacks and define security criterias
- Survive cryptanalysis and update security criterias



When is it hard to be rigid?

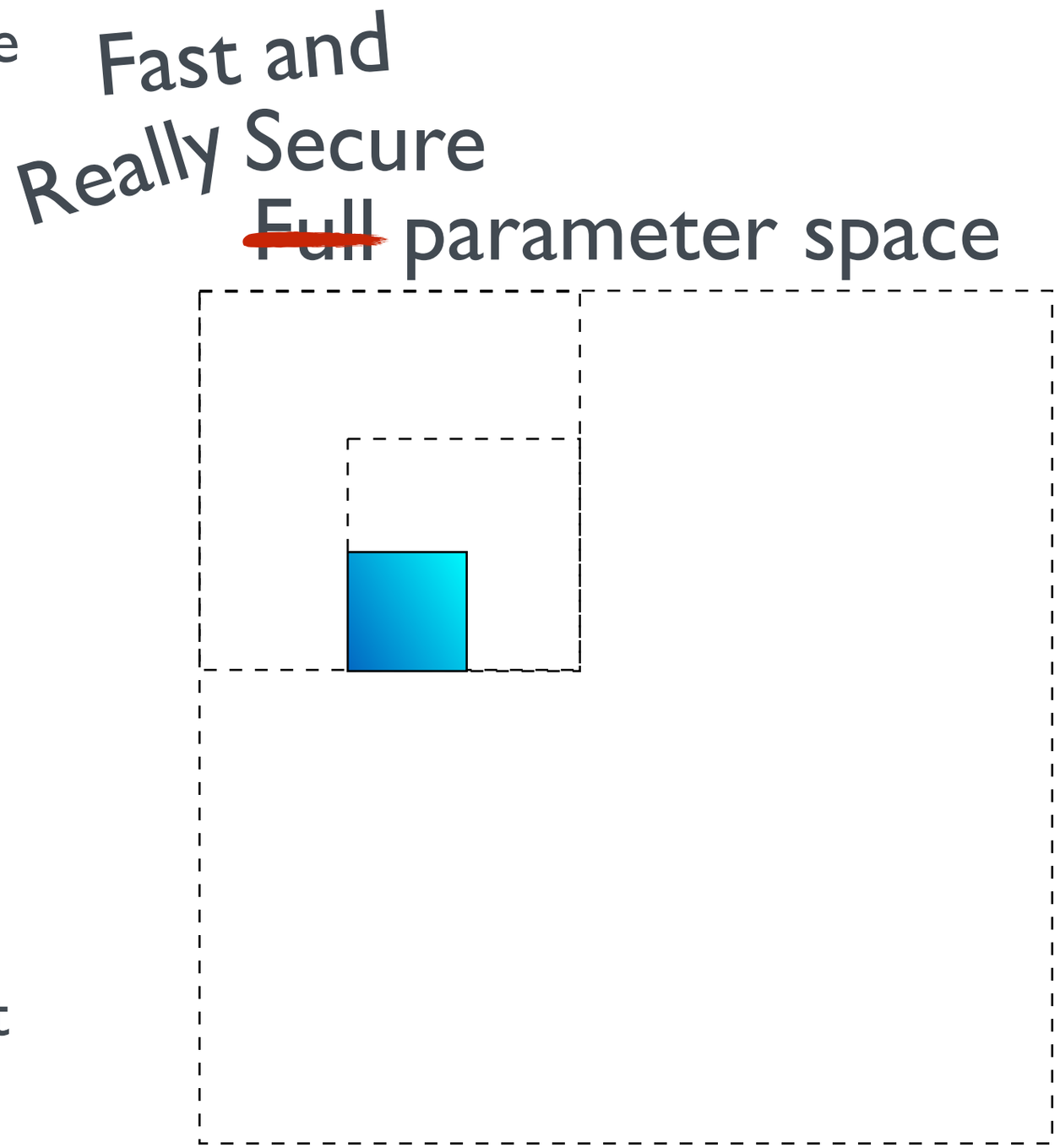
- The Designer starts with the full parameter space
- List all attacks and define security criterias
- Survive cryptanalysis and update security criterias
- Add implementation/ efficiency constraints



When is it hard to be rigid?

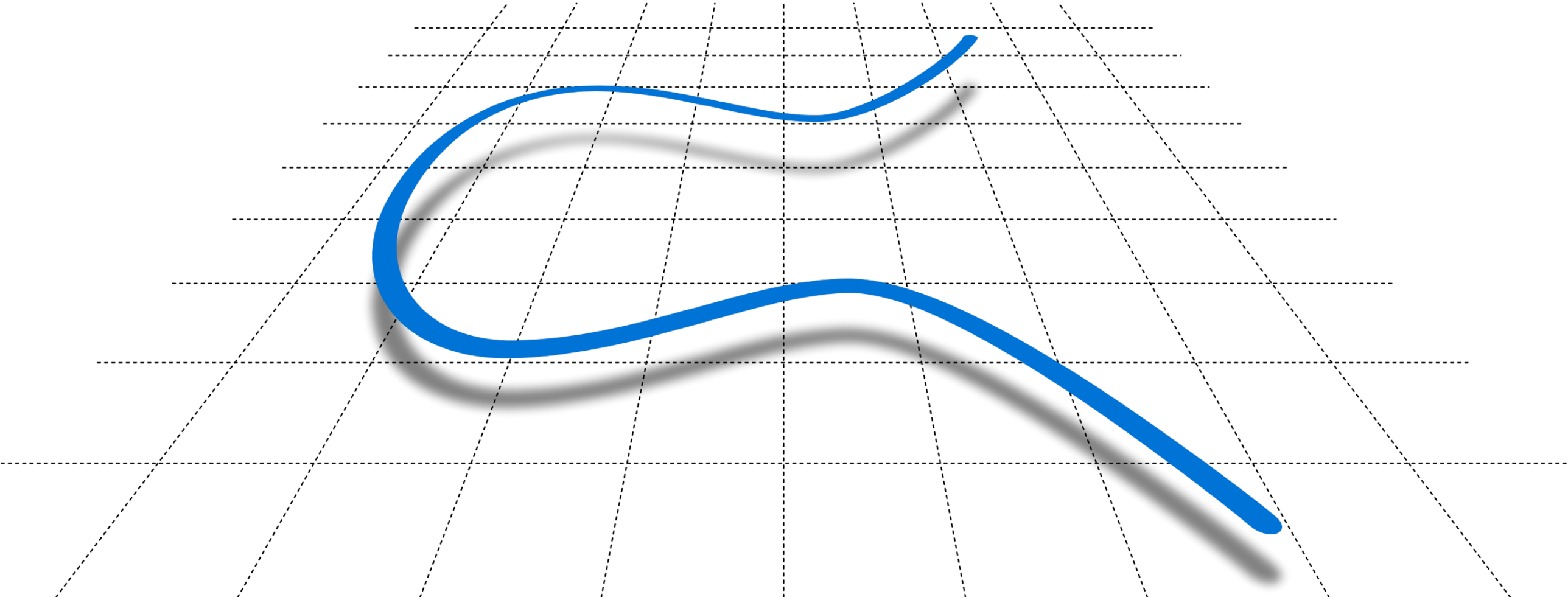
- The Designer starts with the full parameter space
- List all attacks and define security criterias
- Survive cryptanalysis and update security criterias

- Add implementation/ efficiency constraints
- Two cases:
  - A single parameter set left
  - Many choices left

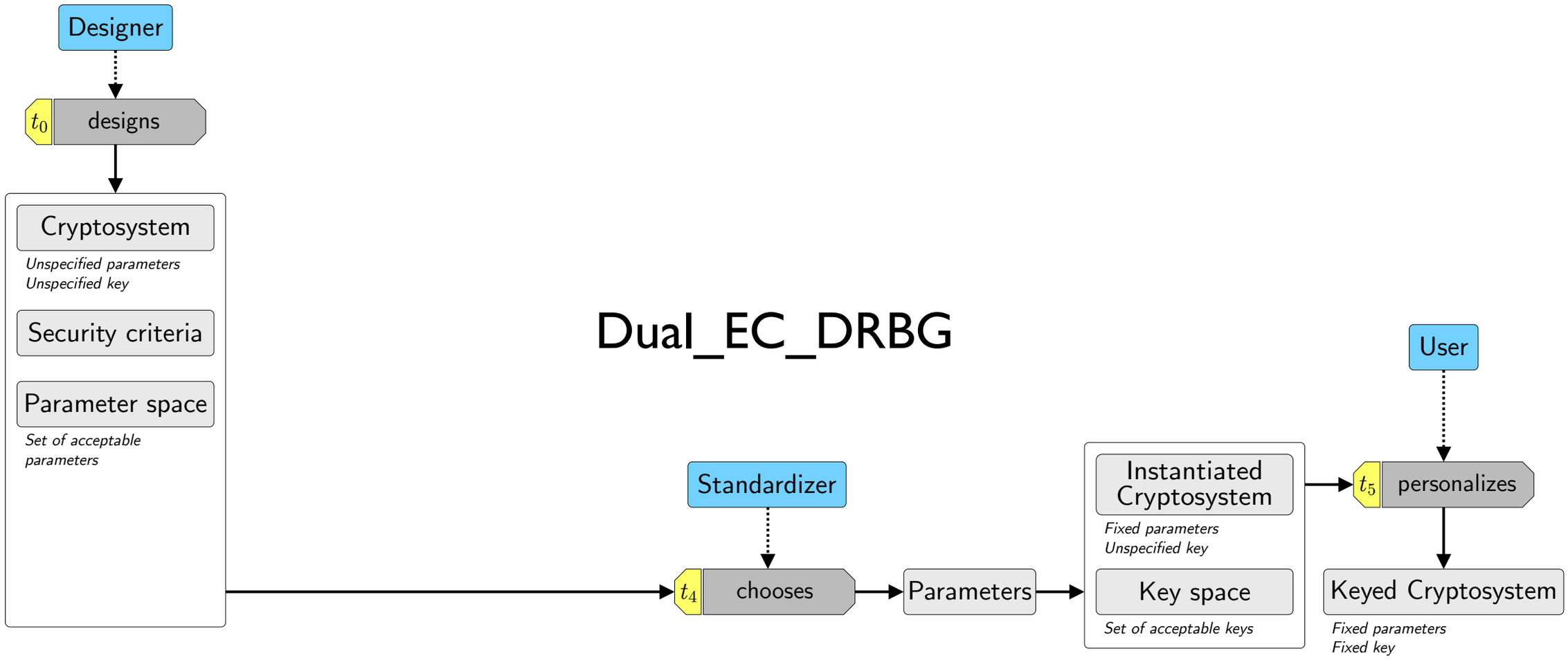


Focus of the rest of this talk

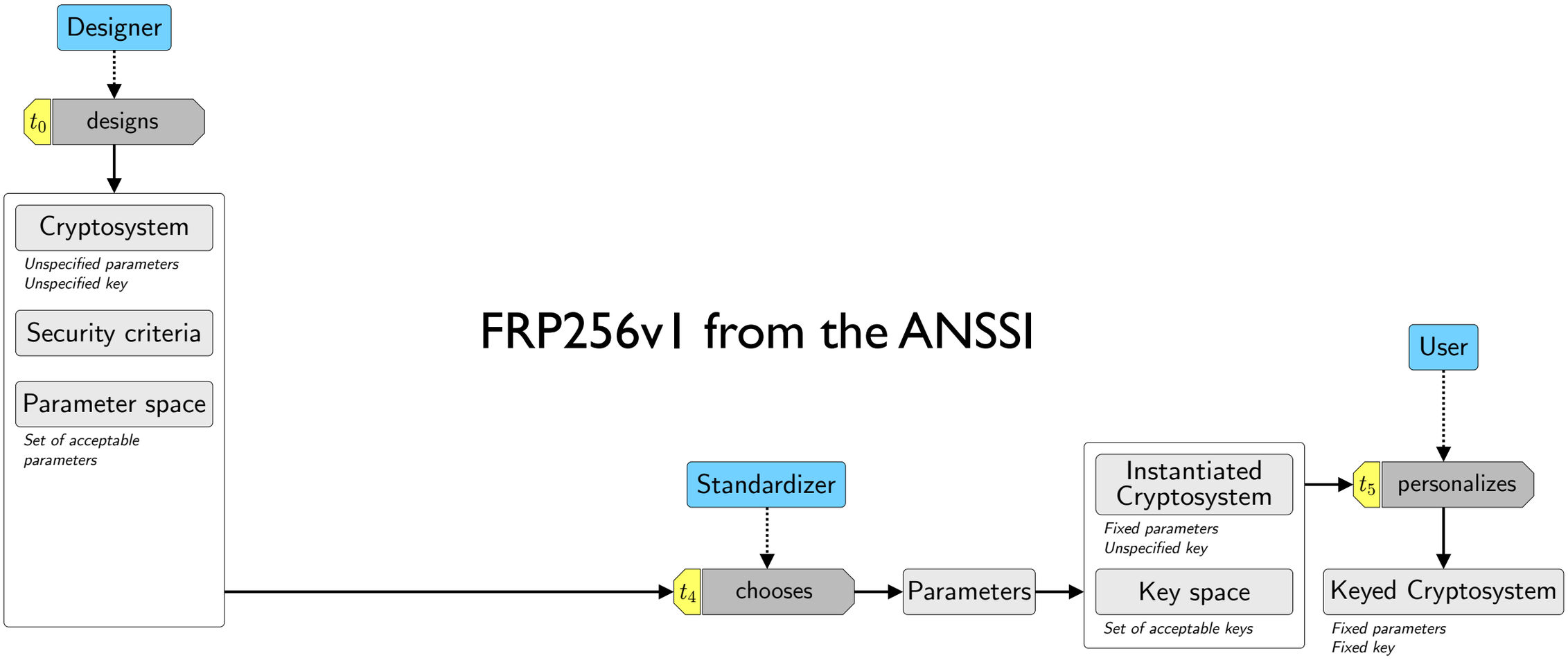
- Explore how a honnest Standardizer can instantiate **fully rigid** cryptosystem parameters when there are *many choices left*
- Running example: elliptic curve parameters selection



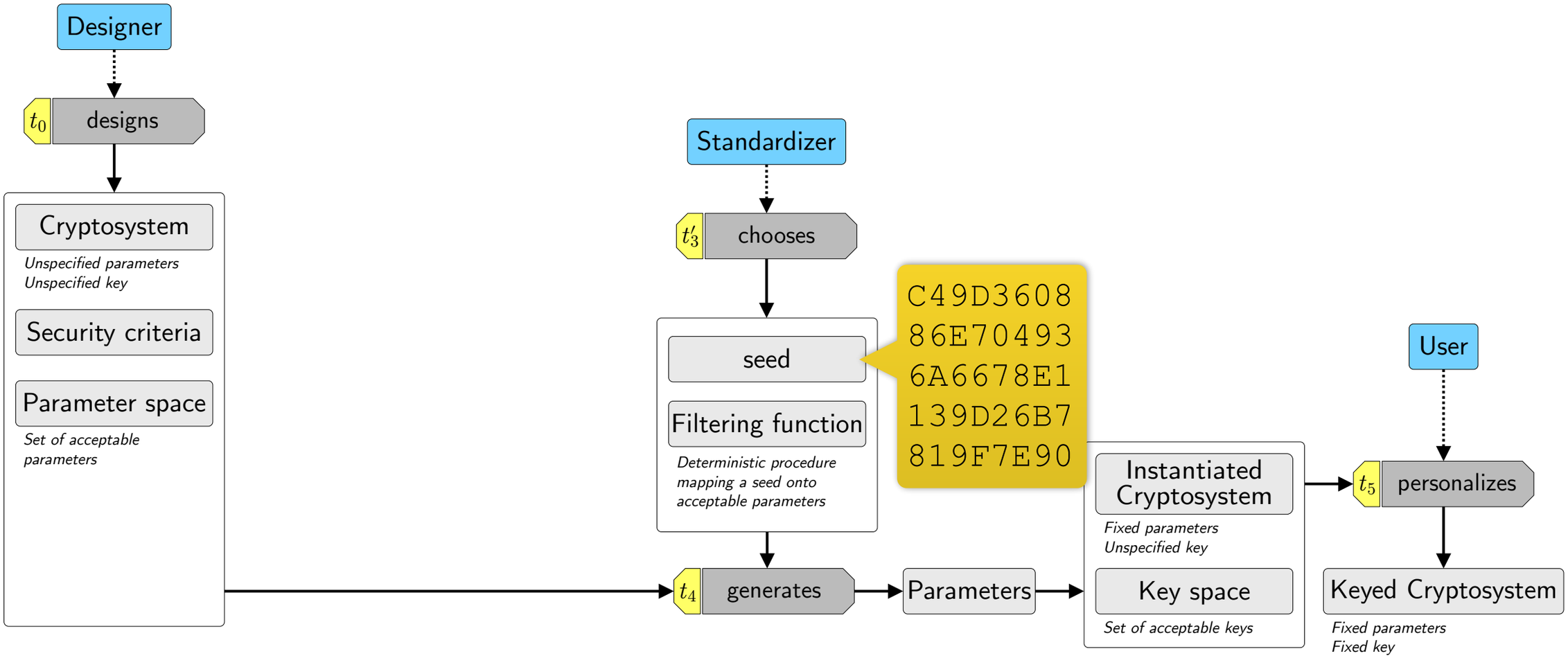
Dual_EC_DRBG


Designer
t_0
designs
Cryptosystem
Unspecified parameters
Unspecified key
Security criteria
Parameter space
Set of acceptable parameters
Standardizer
t_4
chooses
Parameters
Instantiated Cryptosystem
Fixed parameters
Unspecified key
Key space
Set of acceptable keys
User
t_5
personalizes
Keyed Cryptosystem
Fixed parameters
Fixed key

FRP256v1 from the ANSSI

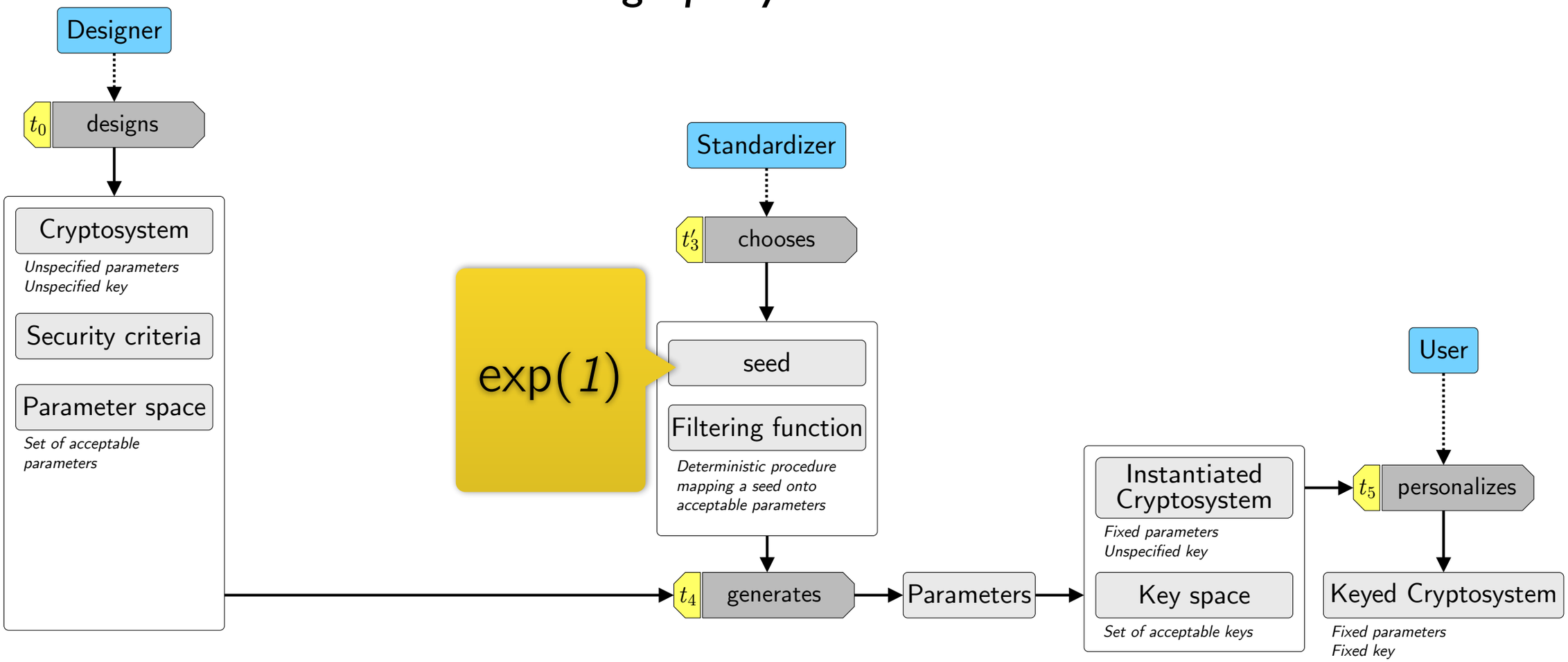


P256 (NIST)

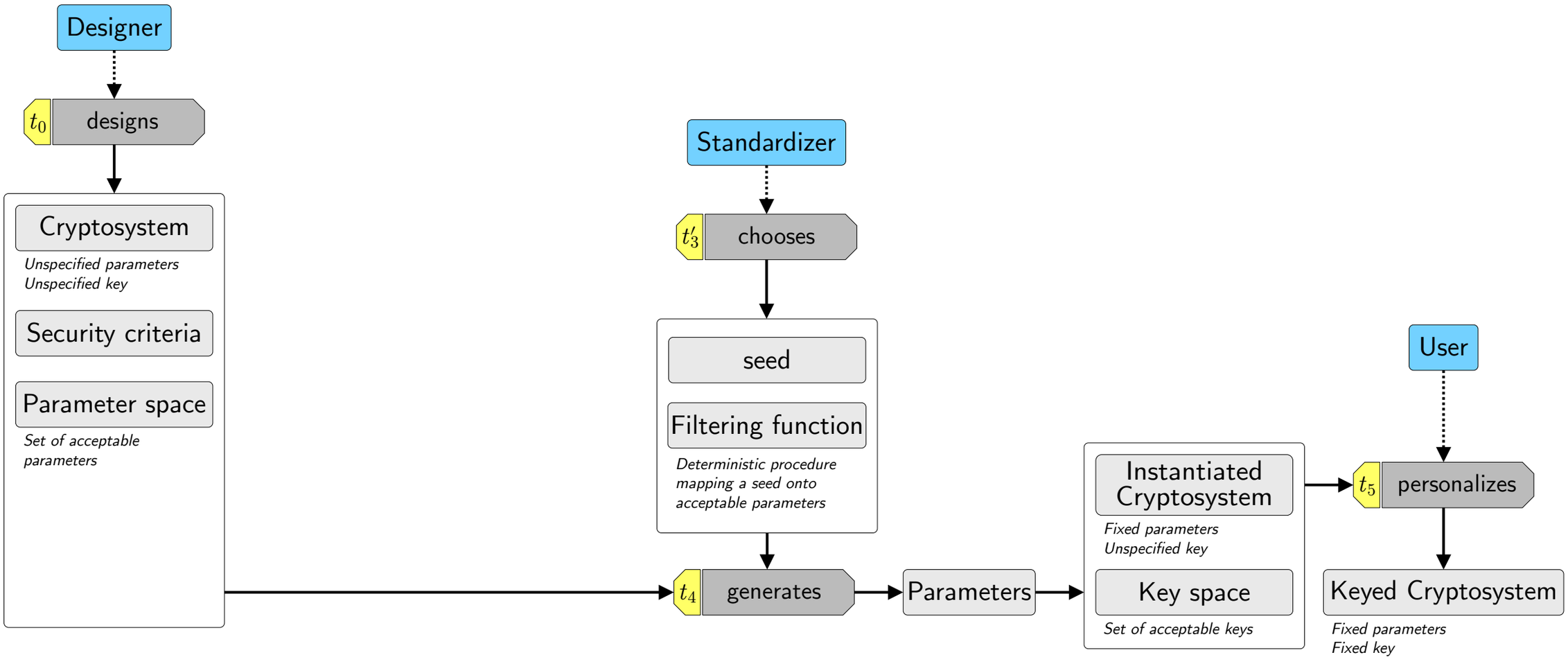

Designer
t_0
designs
Cryptosystem
Unspecified parameters
Unspecified key
Security criteria
Parameter space
Set of acceptable parameters
Standardizer
t'_3
chooses
seed
Filtering function
Deterministic procedure mapping a seed onto acceptable parameters
C49D3608
86E70493
6A6678E1
139D26B7
819F7E90
t_4
generates
Parameters
Instantiated Cryptosystem
Fixed parameters
Unspecified key
Key space
Set of acceptable keys
User
t_5
personalizes
Keyed Cryptosystem
Fixed parameters
Fixed key

Brainpool

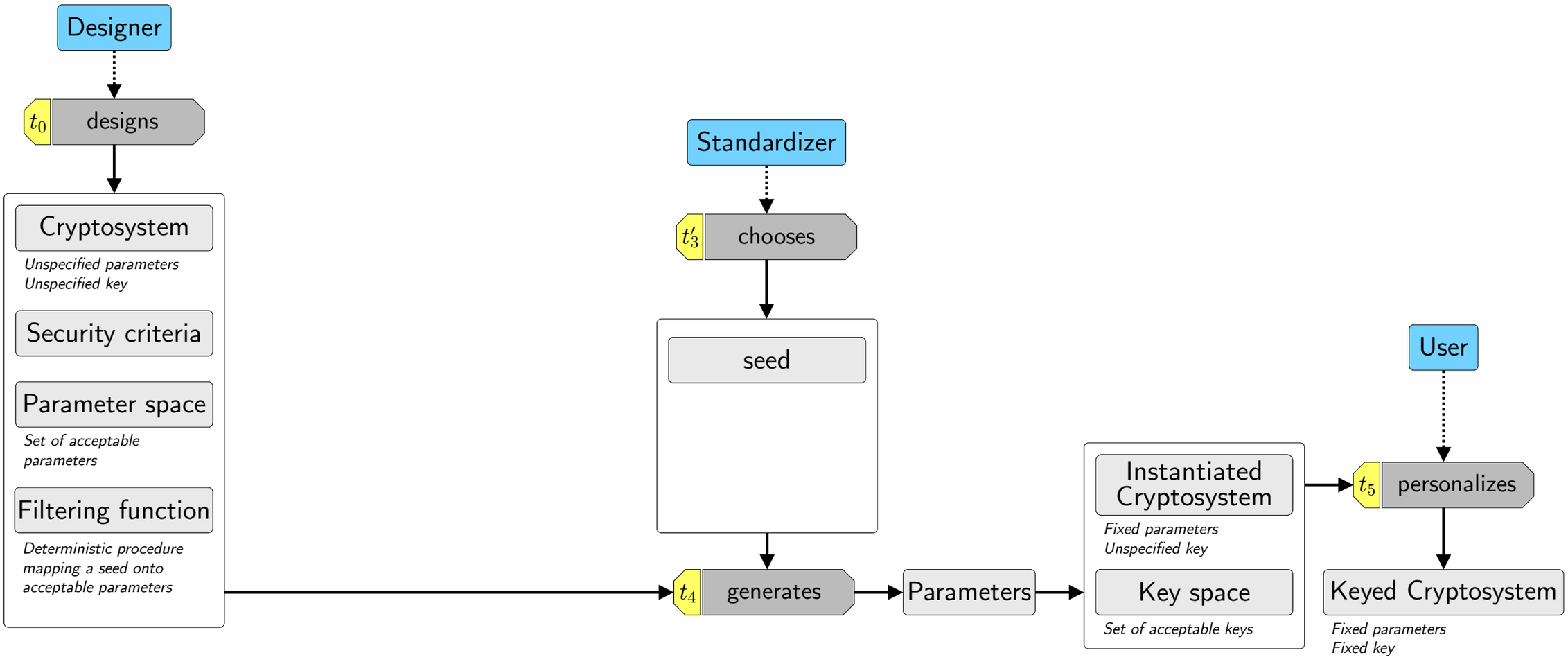
"Nothing up my sleeve number"



We can do better than this!



We can do better than this!

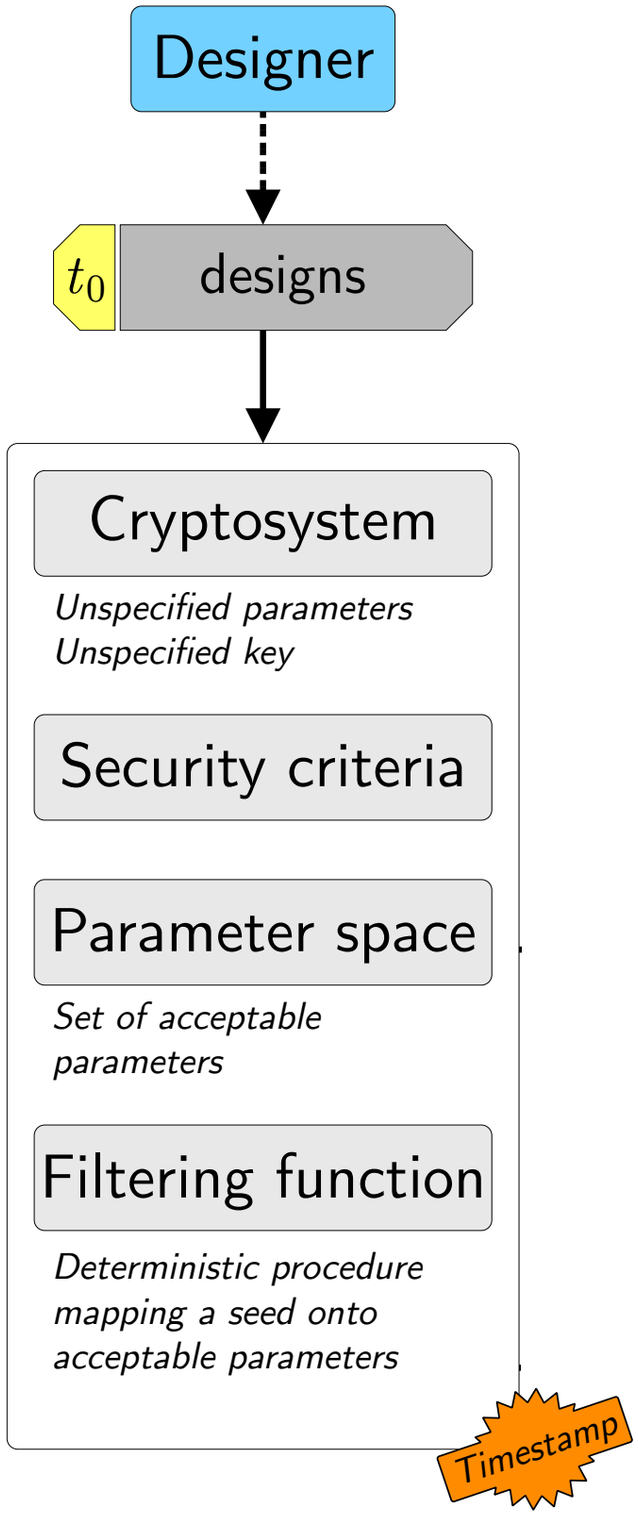


Publicly Verifiable Randomness

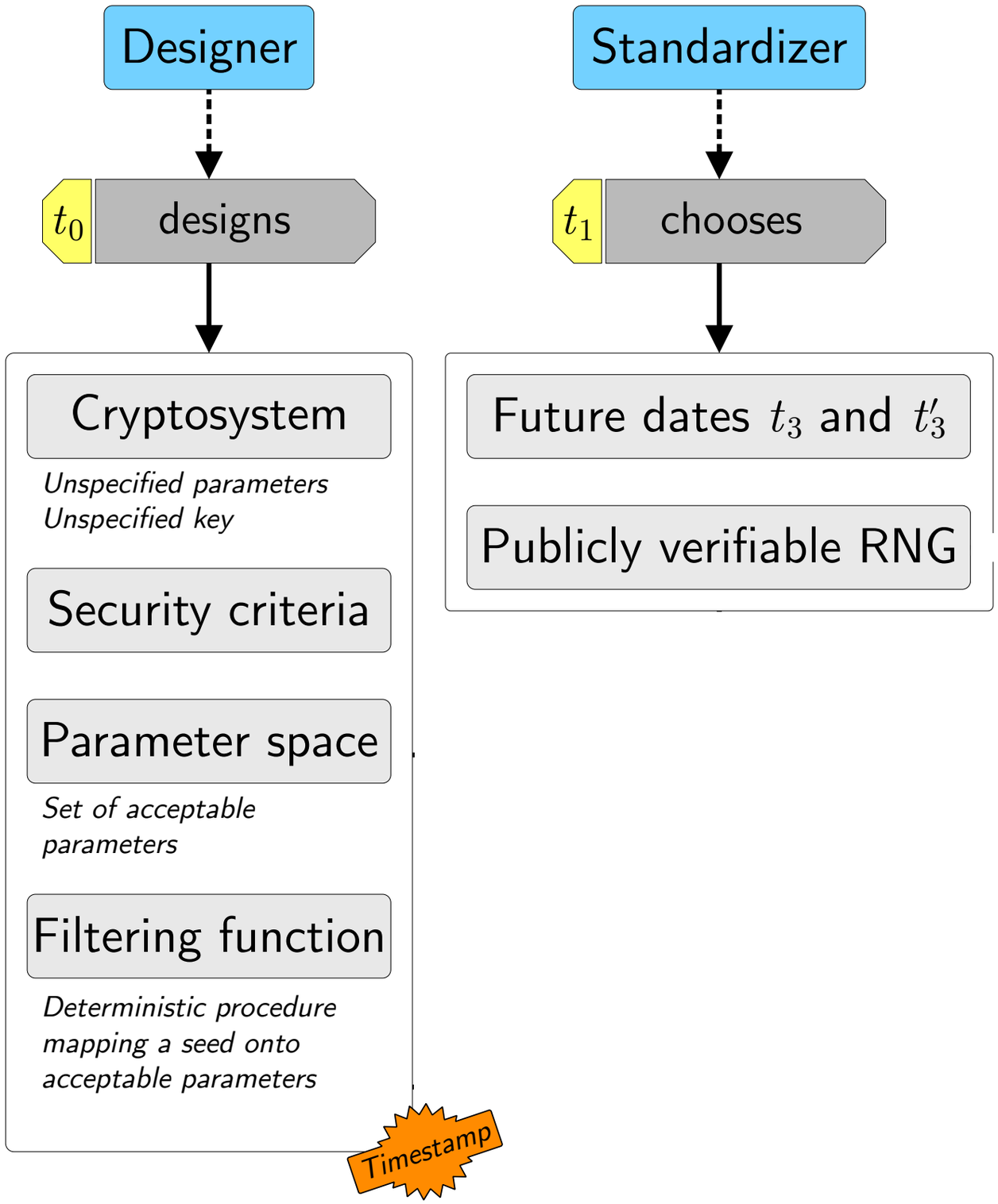
- The key to generating a trustworthy seed is to make it **publicly verifiable**, i.e., generated by…

- A **Publicly verifiable RNG** (PVR)
  - very hard to manipulate/predict
  - publicly observable, i.e., such that *all* outcomes are public
  - archived, so that anyone can pick a past date and get the corresponding outcome.

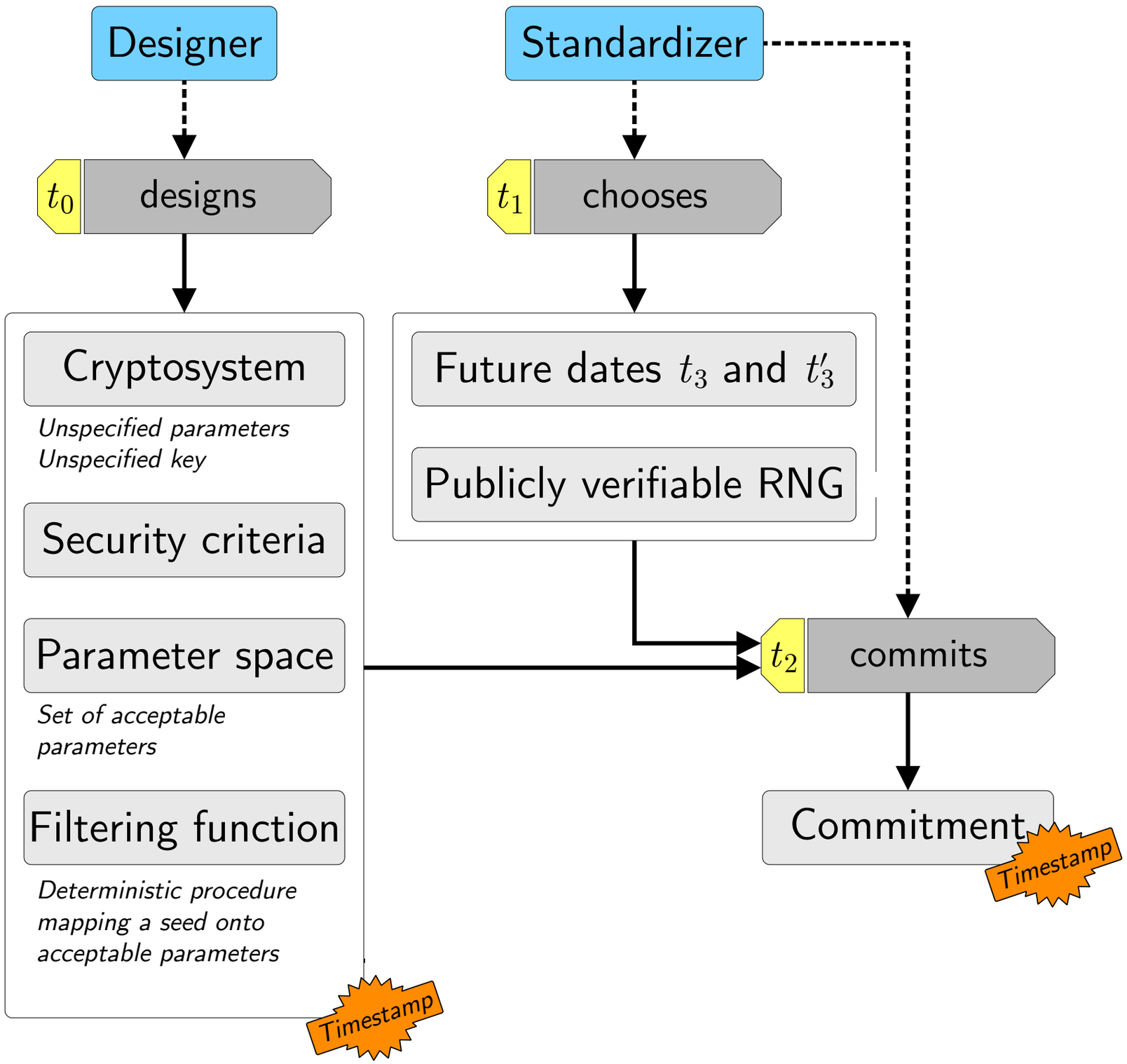
- Let's see how to use a PVR in our generation process…



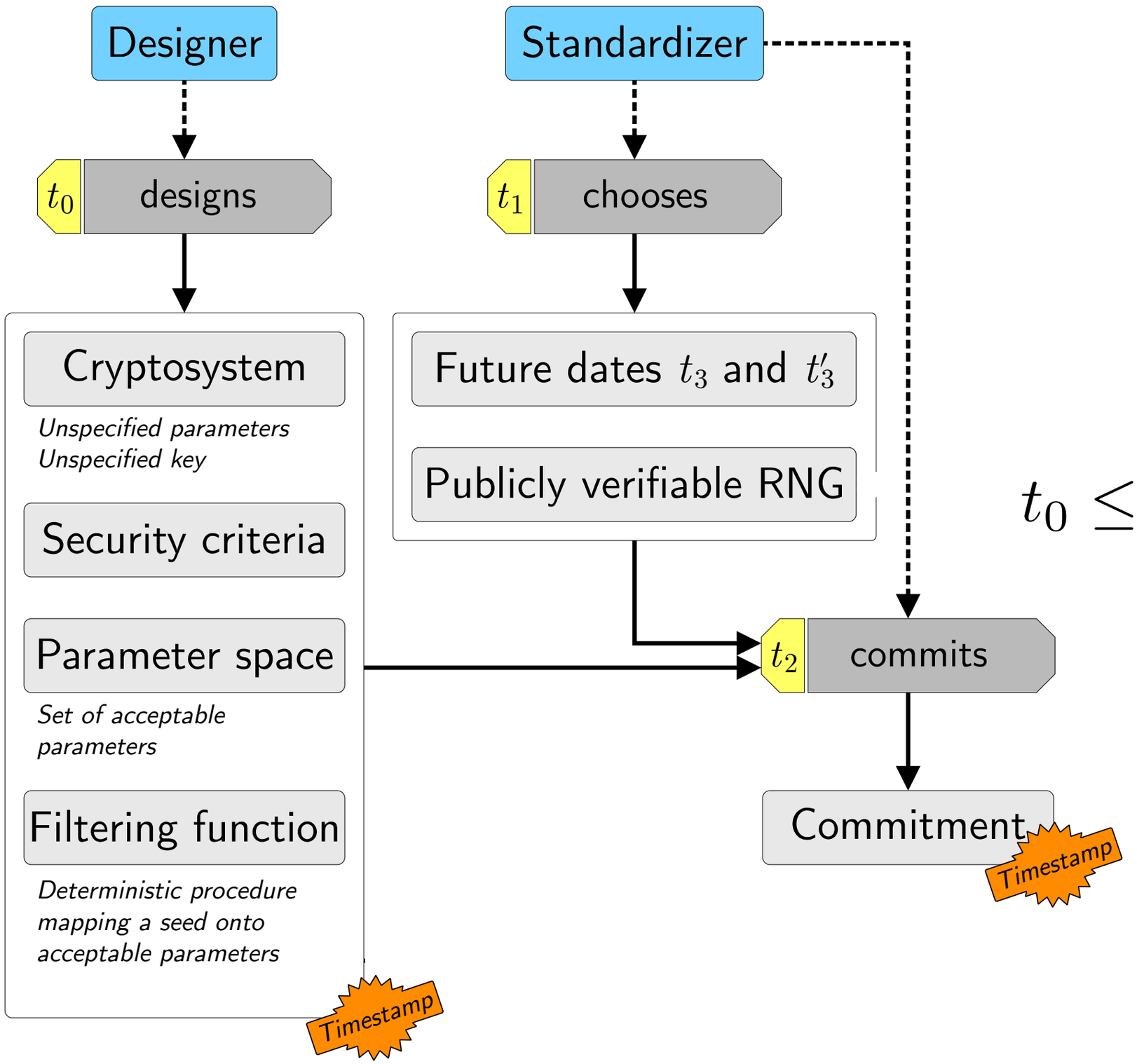
At any time $t \geq t_0$, anyone can check that the Design has not changed since t_0.



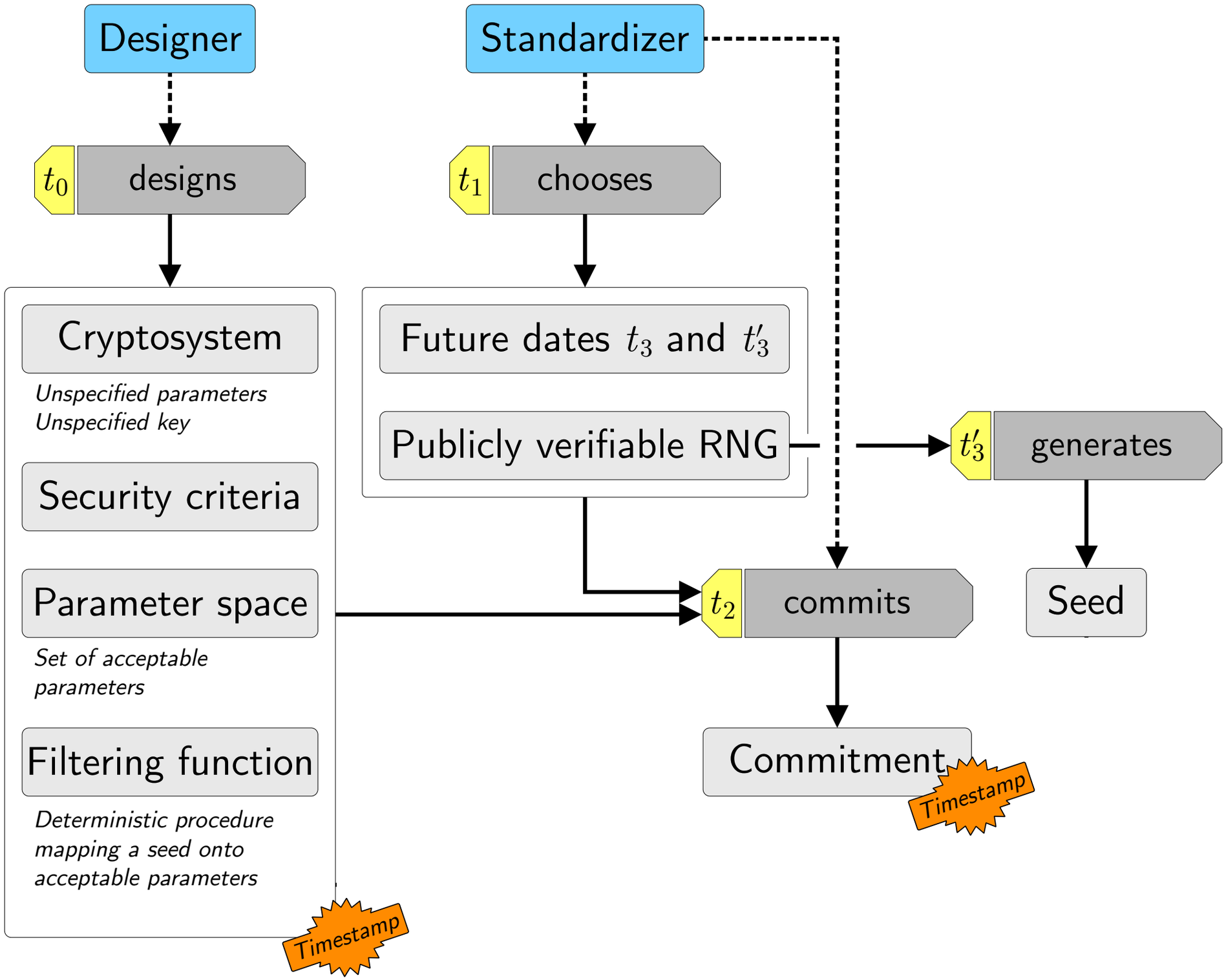
$$t_0 \leq t_1 < t_3 < t'_3$$

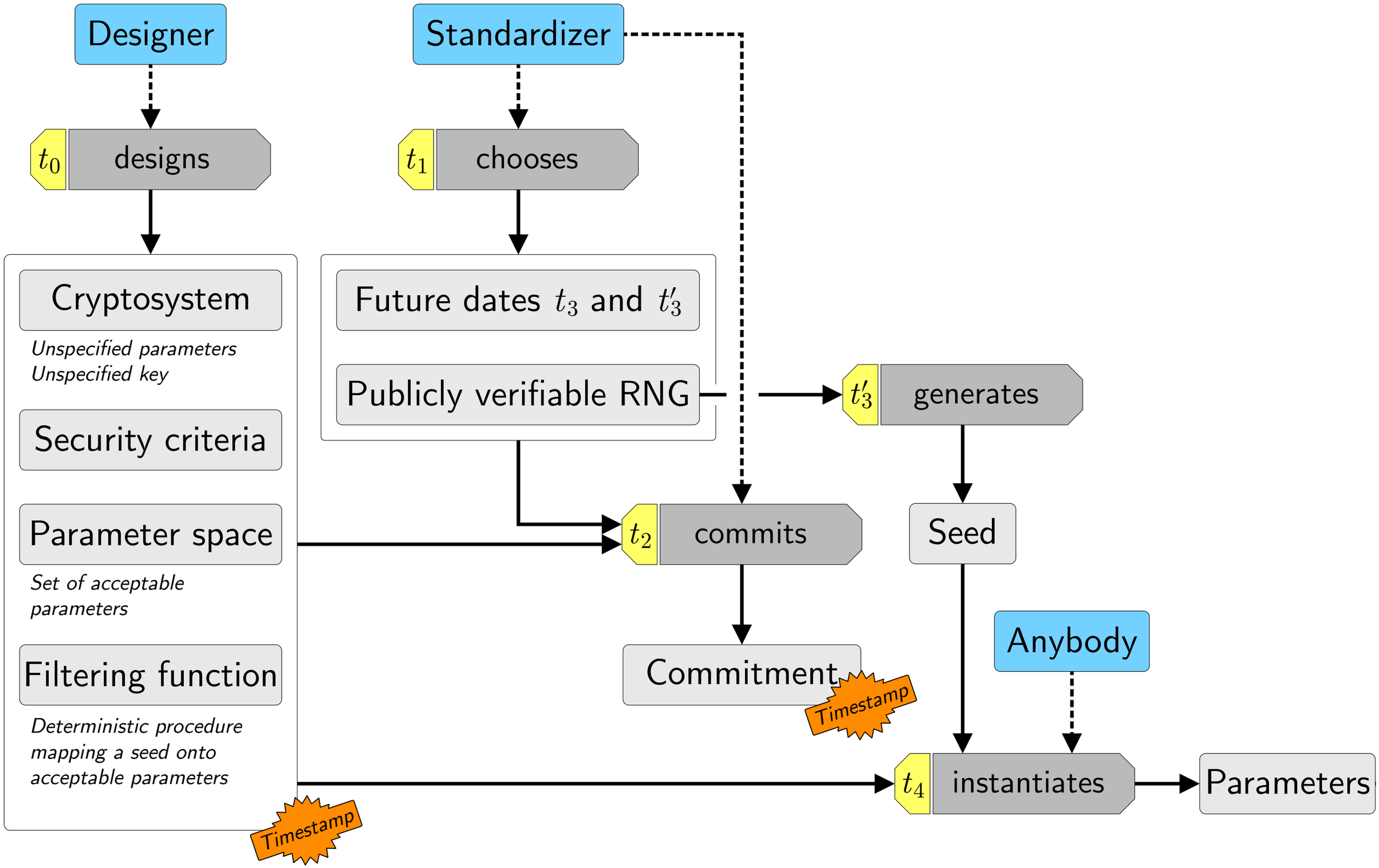


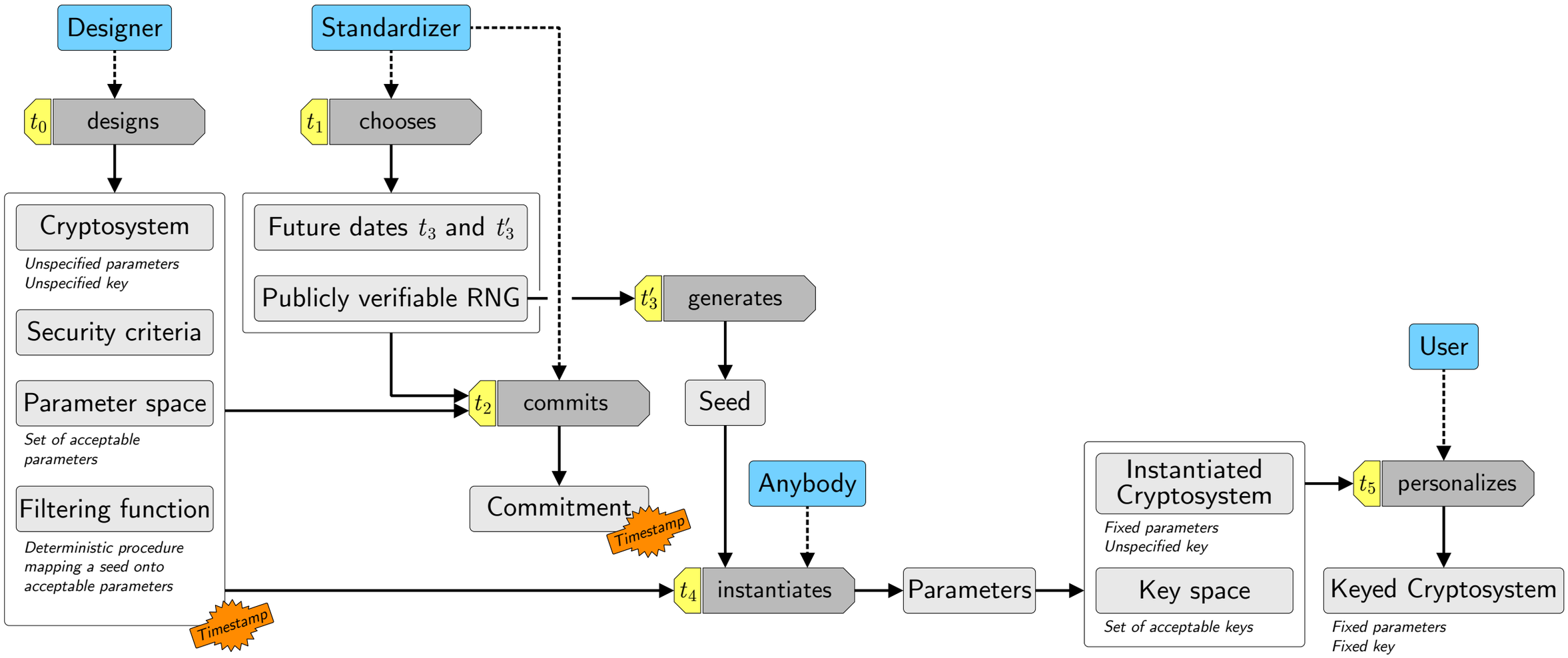
$$t_0 \leq t_1 < t_3 < t'_3$$



$$t_0 \leq t_1 \leq t_2 < t_3 < t'_3$$


Designer
Standardizer
t_0
designs
t_1
chooses
Cryptosystem
Unspecified parameters
Unspecified key
Security criteria
Parameter space
Set of acceptable parameters
Filtering function
Deterministic procedure mapping a seed onto acceptable parameters
Timestamp
Future dates t_3 and t'_3
Publicly verifiable RNG
t'_3
generates
Seed
t_2
commits
Commitment
Timestamp


Designer
Standardizer
t_0
designs
t_1
chooses
Cryptosystem
Unspecified parameters
Unspecified key
Security criteria
Parameter space
Set of acceptable parameters
Filtering function
Deterministic procedure mapping a seed onto acceptable parameters
Future dates t_3 and t'_3
Publicly verifiable RNG
t'_3
generates
t_2
commits
Seed
Commitment
Timestamp
Anybody
Timestamp
t_4
instantiates
Parameters

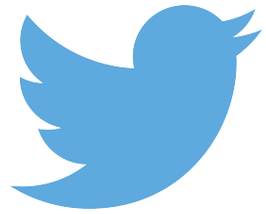

Designer
Standardizer
t_0 designs
t_1 chooses
Cryptosystem
Unspecified parameters
Unspecified key
Security criteria
Parameter space
Set of acceptable parameters
Filtering function
Deterministic procedure mapping a seed onto acceptable parameters
Future dates t_3 and t'_3
Publicly verifiable RNG
t'_3 generates
t_2 commits
Seed
Commitment
Timestamp
Timestamp
Anybody
t_4 instantiates
Parameters
Instantiated Cryptosystem
Fixed parameters
Unspecified key
Key space
Set of acceptable keys
User
t_5 personalizes
Keyed Cryptosystem
Fixed parameters
Fixed key

PVRs are hard to find

- "Classical" cryptographic RNGs are not enough
  - ➙ not publicly observable
- random.org
  - ➙ trivially manipulatable
- Stock market prices
  - ➙ possibly manipulatable (expensive)
- Weather data
  - ➙ somewhat predictable (a few days ahead)
- The Bitcoin block chain (Bonneau, Clark, and Goldfeder)
  - ➙ somewhat manipulatable? (Pierrot and Wesolowski)
- In many cases: hard to measure the exact entropy

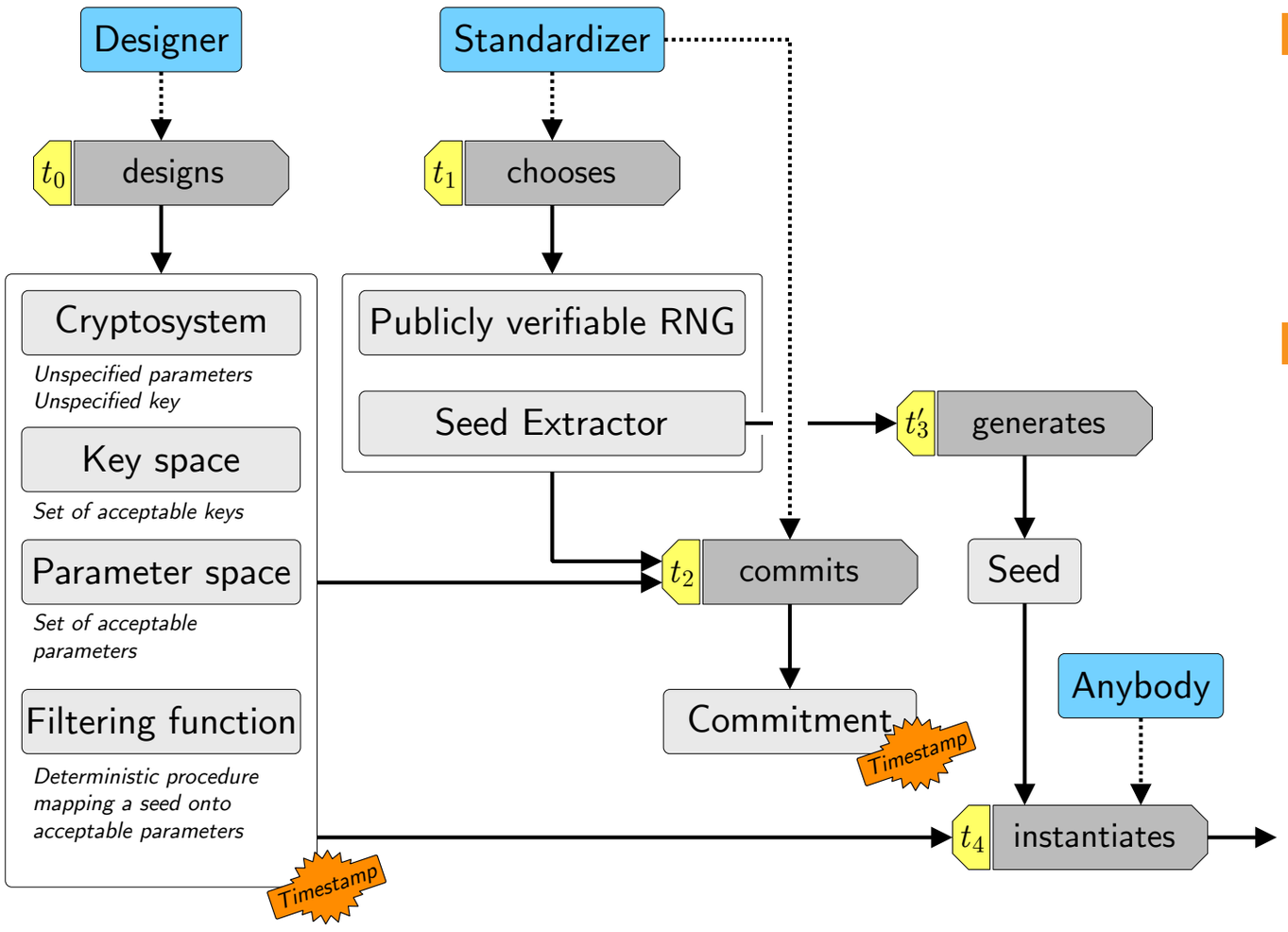
Two recent PVR constructions

- Lenstra and Wesolowski
  - Twitter
  - Slow hash function
  - Webcam



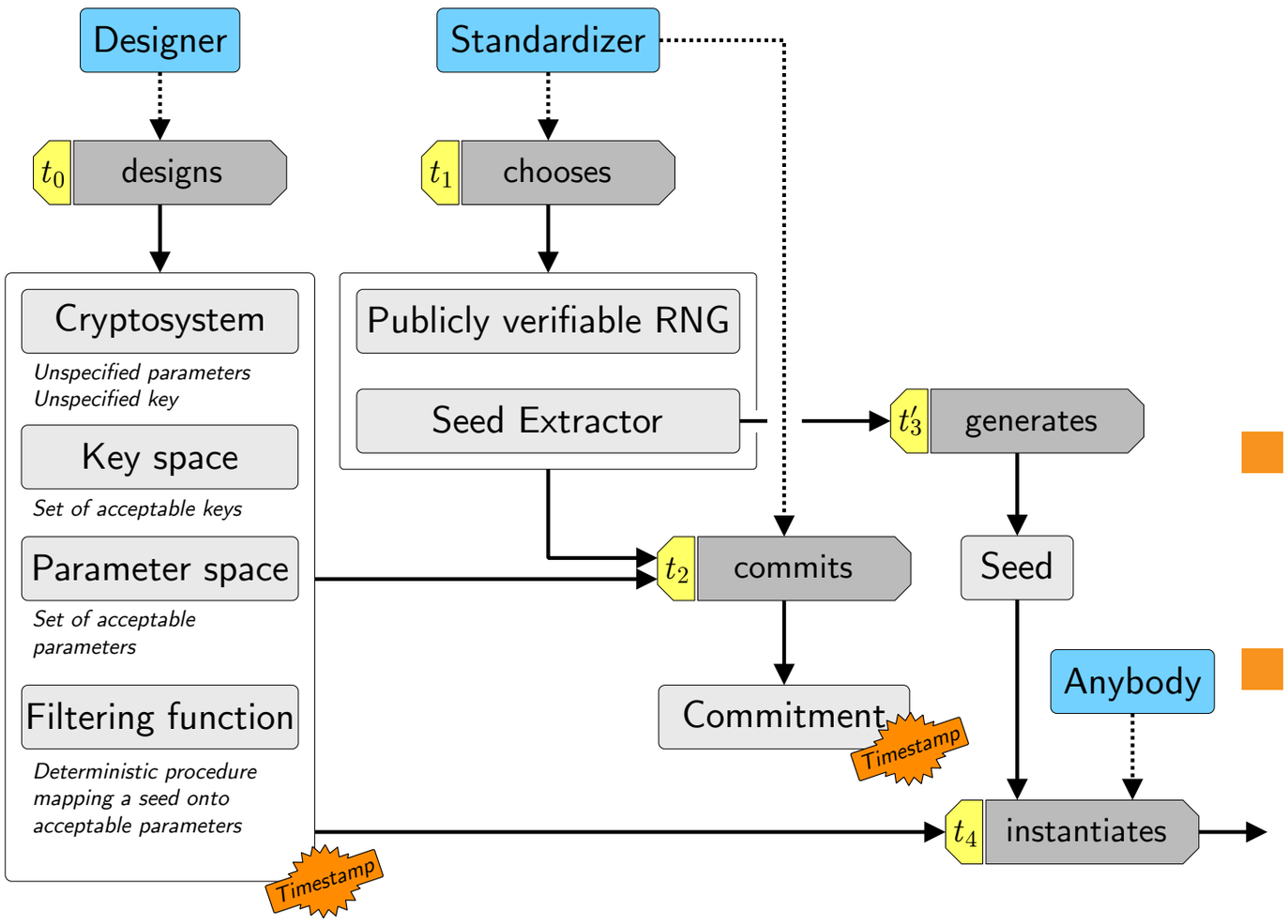
- CryptoExperts' team
  - National lotteries from around the world

Lenstra-Wesolowski PVR (Unicorn protocol)



- t_2 : Commit on t_3 and t'_3
- t_3 : Collect data from tweets containing hashtag `#trx_curve_xxxxx`
- t'_3
  - Stop collecting Twitter data → s_0
  - Webcam (EPFL parking lot) → s_1
  - Instantly publish $c = h(h(s_0,s_1))$
  - Start *SlowHash*$(h(s_0,s_1))$
- t_4 : Seed is available
- Faster verification procedure

CryptoExperts' Million Dollar PVR



- At t_2 publish…
  - An ordered list of future lottery draws happening between t_3 and t'_3
  - A Seed Extractor (turning an ordered list of draws into a seed)
- At t'_3 publish the draw results and the resulting seed
- At t_4 anybody can check the results and (re)generate the seed

Pros and Cons

Lenstra-Wesolowski PVR

- ✓ Fully automated, generates several curves every day
- ✓ Uses Twitter, so everyone can participate
- x Uses Twitter, and tweets can be deleted
- x Hard to predict/measure entropy
- x Non-trivial tasks have to be performed by the Standardizer during the process

Million Dollar PVR

- ✓ Completely decentralized
- ✓ Entropy easy to quantify
- ✓ Once the draw list has been committed, the process cannot be stopped
- xx Lotteries have a bad reputation

Conclusion

- When generating parameters, Standardizers should do it in a **fully rigid** way
- This can be done for *any* cryptographic scheme

  … provided that we have a convincing PVR

- Two PVR constructions have been proposed recently
- I believe both proposals to be convincing

  … but this feeling is not shared by everyone, making these PVRs imperfect

  In that sense, no PVR will ever be perfect

- At least, proposals like these set a minimum level of requirements for future standards.

Bonus slide

- Yesterday night…
- There might be another Dual_EC_DRBG out there
- X9.82 (June 2004 draft), section 10.3.3 : Micali-Schnorr DRBG
- Similar to BBS, except that you compute the output as
  - x^e mod n
- n is an RSA modulus
- Section 10.3.3.3.3, Selection of the M-S parameters:

  *"The application may request a private modulus, or it may use the default modulus of the appropriate size as given in Annex A.2. The implementation **shall** permit either […]"*
- Needless to say, you won't find the primes in Annex A.2, only the moduli of "appropriate size".
- Definitely not "Fully rigid"

Thank you !

Parameters of the MDC ;-)

http://cryptoexperts.github.io/million-dollar-curve/

CryptoExperts

Bonus Material:

More on CryptoExperts' PVR

CRYPTOEXPERTS

Turning a draw into a (small) seed

- We consider a lottery that draws m numbers among n.
- Each draw is converted into an integer (uniformly distributed index) between 0 and $\binom{n}{m}$

$$c_1 < c_2 < \cdots < c_m$$

$$x = \binom{c_1 - 1}{1} + \binom{c_2 - 1}{2} + \cdots + \binom{c_m - 1}{m}$$

Small seeds to one large seed

- We combine the draws from *r* loteries using a mixed radix technique
- The specific order in which the draws are chosen must be defined beforehand
- ≈5000 bits of entropy per week

Lottery name	m	n	Entropy	Drawings per week	Entropy per week
Australian OZ Lotto	7	45	25.43	1	25.43
Australian Saturday Lotto	6	45	22.95	1	22.95
Australian Monday Lotto	6	45	22.95	1	22.95
Australian Wednesday Lotto	6	45	22.95	1	22.95
Australian Powerball	6	40	21.87	1	21.87
Belgian Lotto	6	45	22.95	2	45.91
Brasilian Mega-Sena	6	60	25.58	2	51.15
Brasilian Dupla Sena	6	50	23.92	2	47.84
Brasilian Quina	5	80	24.52	2	49.04
Brasilian Lotofácil	15	25	21.64	3	64.92
Canadian Loto Max (Main Draw)	7	49	26.36	1	26.36
Canadian Loto 649 (Main Draw)	6	49	23.74	2	47.47
Canadian Daily Keno	20	70	57.17	14	800.35
Canadian Lottario	6	45	22.96	1	22.96
Dutch Lotto	6	45	22.96	1	22.96
Euromillions	5	50	21.01	2	42.03
French Loto	5	49	20.86	3	62.58
French Keno	20	70	57.17	14	800.35
German Lotto	6	49	23.74	2	47.47
German Keno	20	70	51.17	7	400.17
German Euro Jackpot	5	50	21.01	1	21.01
Italian SuperEnalotto	6	90	29.21	3	87.64
New Zealand Lotto	6	40	21.87	2	43.74
New Zealand Keno	20	80	61.62	28	1725.26
Mauritius Loto	6	40	21.87	1	21.87
Spanish La Primitiva	6	49	23.74	2	47.47
Spanish BonoLoto	6	49	23.74	6	142.42
Spanish El Gordo	5	54	21.59	1	21.59
Swiss Loto	6	42	22.32	2	44.65
UK Health Lottery (£1 Draw Game)	5	50	21.01	5	105.07
US Powerball	5	69	23.42	2	46.84
US Hot Lotto	5	47	20.55	2	41.10
US Wild Card	5	33	17.87	2	35.71
US Mega Millions	5	75	24.04	2	48.08
US NY Lotto	6	59	25.43	2	50.85
US NY Cash 4Life	5	60	22.38	2	44.76
Total					≈ 5175.77

Bonus Material:

Million Dollar Curve

CryptoExperts

Security criteria for MDCurve

- p of 256 bits, scuh that $p \equiv 3 \pmod 4$
- Edwards curve
- Cardinality of E(F_p) of the form $4q$
- q different from p (anomalous curve attack)
- Large "embedding degree" (like Brainpool)
- Large "CM field discriminants"
- "Twist secure"

CryptoExperts

Filtering function

- In less than 2 weeks, we can obtain 4×2048 bits…
- …that we use to generate BBS parameters
- We modify an algorithm introduced by Joye, Paillier, and Vaudenay in order to efficiently generate "strong strong primes" starting with a *bounded quantity of entropy*.

Filtering function

- We use BBS in order to generate the parameters p and d of an Edwards curve.
- We keep on generating curves until one satisfies all the security criteria

Want more?

- Paper on the IACR ePrint archive

https://eprint.iacr.org/2015/1249

- Script published sur GitHub
- Project web site on GitHub

http://cryptoexperts.github.io/million-dollar-curve/

- What you won't find…
- our Python script that automatically fetches draw results (python3/Beautiful Soup/json/excel/…)