DLT2026 – 8th Distributed Ledger Technology Workshop, Pula, Italy, June 2026

Mobile Mining for Proof of Space

Fausto Spoto, Dipartimento di Informatica
Verona, Italy
fausto.spoto@univr.it

Ana Lucia Zandomeneghi & José Renato de Oliveira Lima, UFMA
São Luís do Maranhão, Brazil

Wo decides the next block?

Many alternatives

- Proof of Work [PoW] (who commits more work)
- Proof of Stake [PoS] (who commits more money)
- Proof of Authority [PoA] (who has more authority)
- Proof of Space [PoSp] (who commits more disk space)
- ...

Proof of Space is an interesting, niche technology

- fully decentralized
- no CPU cost
- no *special* nodes (no validators)
- miners join and leave anonymously

Blockchain mining as a competitive puzzle

Bitcoin's Proof of Work:

Blockchain mining as a competitive puzzle

Bitcoin's Proof of Work:

challenge: find a **next** such that $hash(\mathbf{next}) > difficulty$

Blockchain mining as a competitive puzzle

Bitcoin's Proof of Work:

challenge: find a **next** such that $hash(\mathbf{next}) > difficulty$

answer: reply with a **next** obtained by rotating all possible values of an unused field *nonce*, until $hash(\mathbf{next}) > difficulty$

Blockchain mining as a competitive puzzle

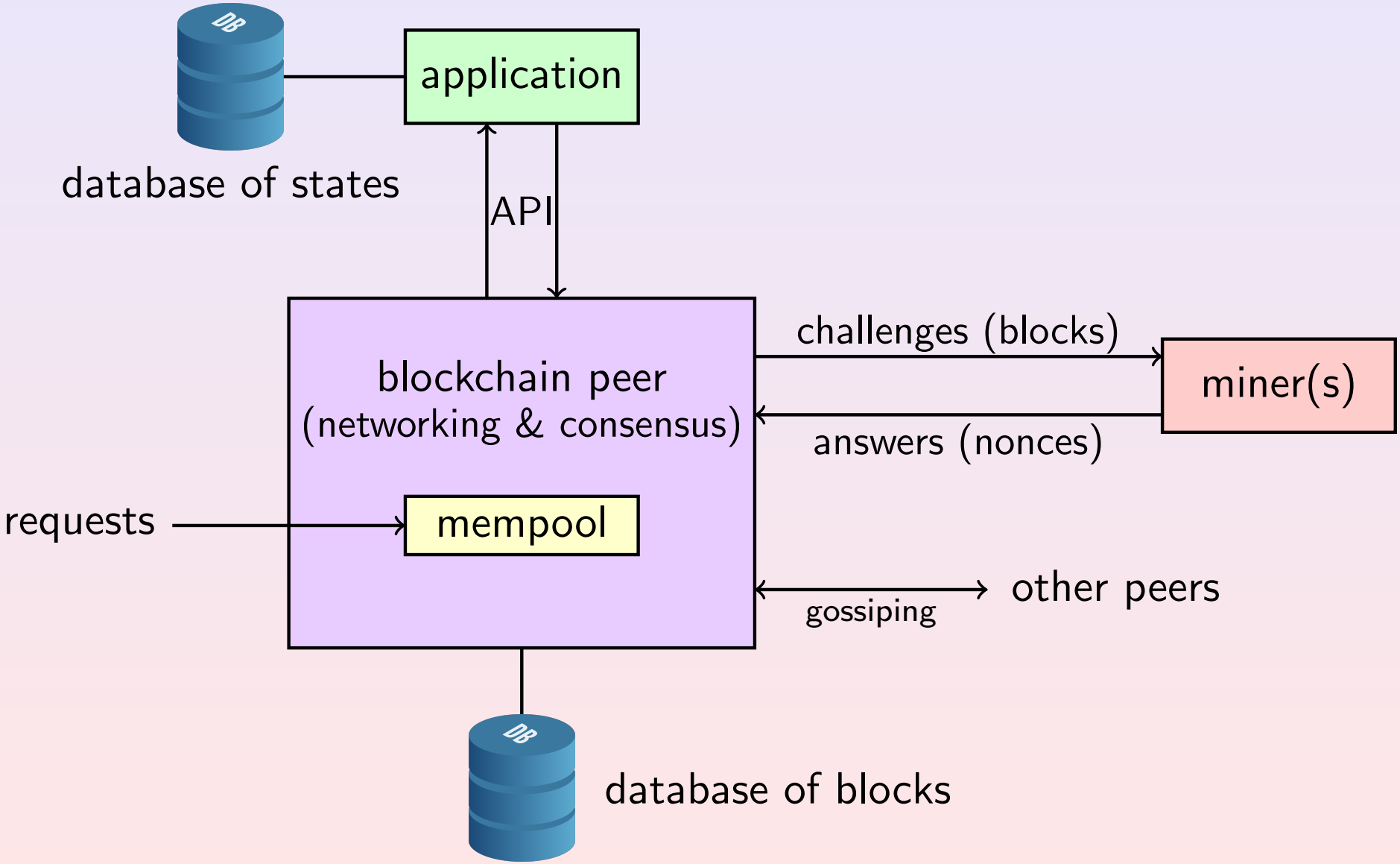
Bitcoin's Proof of Work:

challenge: find a **next** such that $hash(\mathbf{next}) > difficulty$

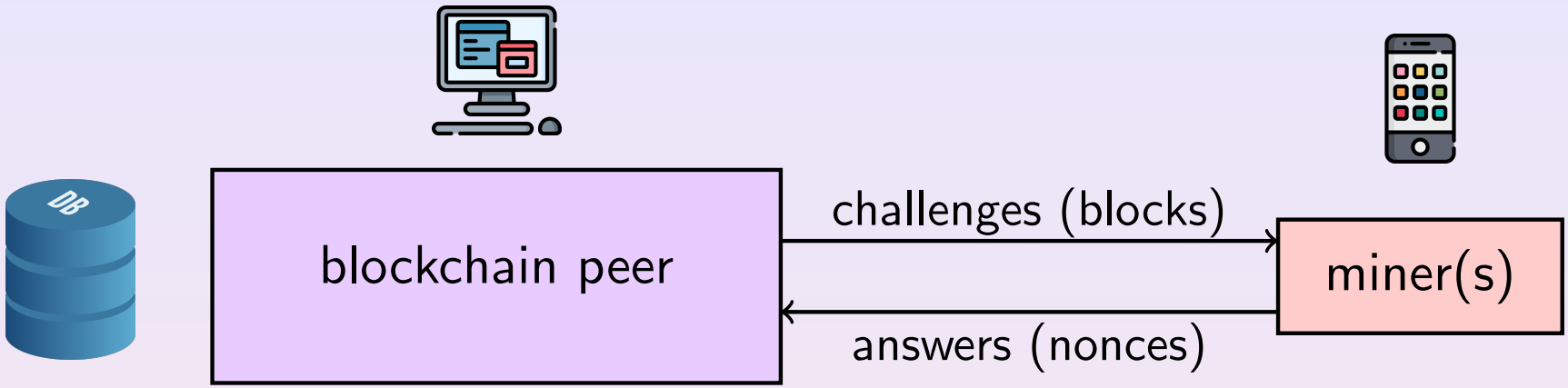
answer: reply with a **next** obtained by rotating all possible values of an unused field *nonce*, until $hash(\mathbf{next}) > difficulty$

competition: in case of more **next**, select the one with the largest hash

A Bitcoin (or old Ethereum) node and its miner(s)

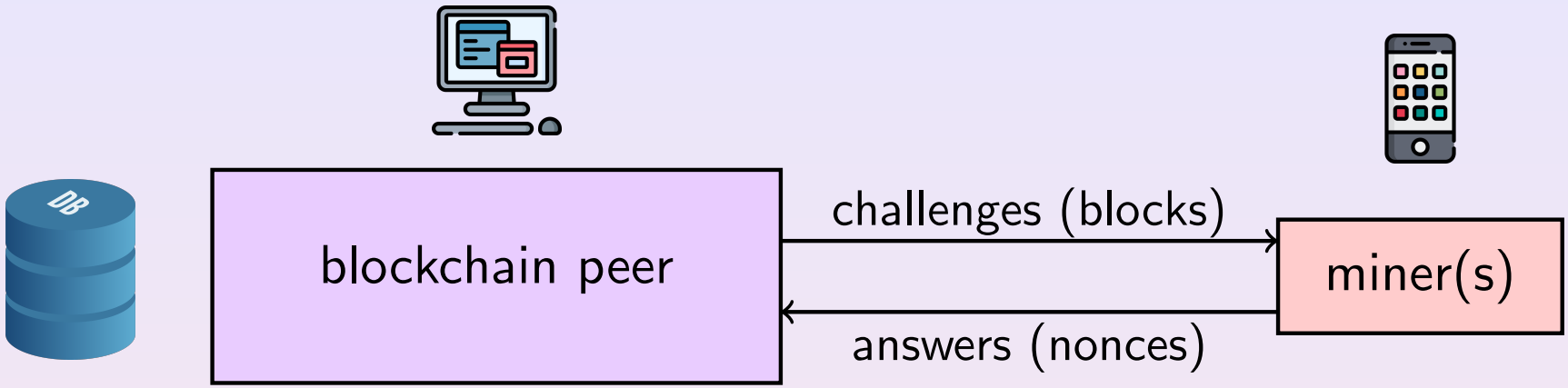


Miners as mobile applications



Why a mobile application?

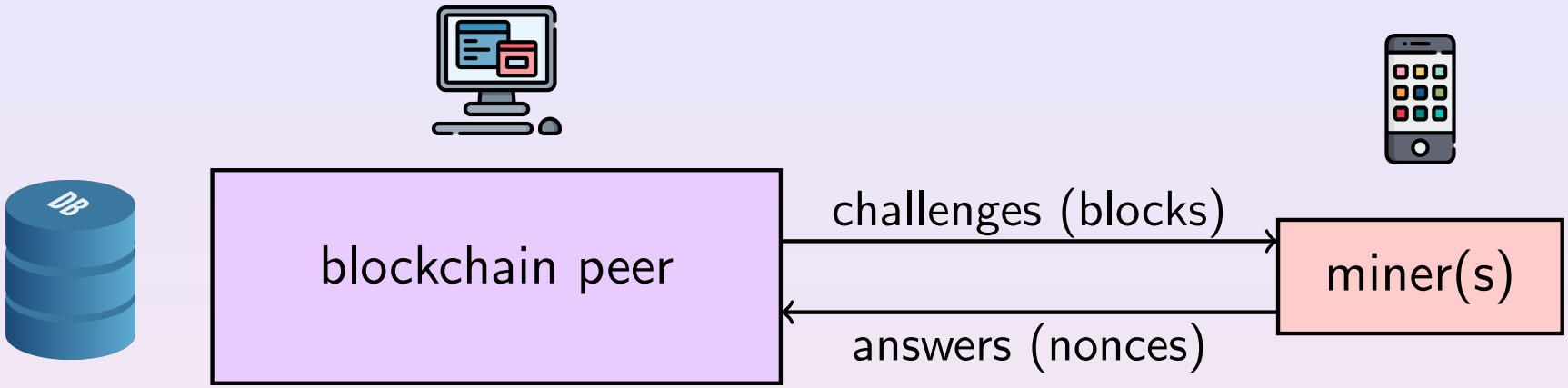
Miners as mobile applications



Why a mobile application?

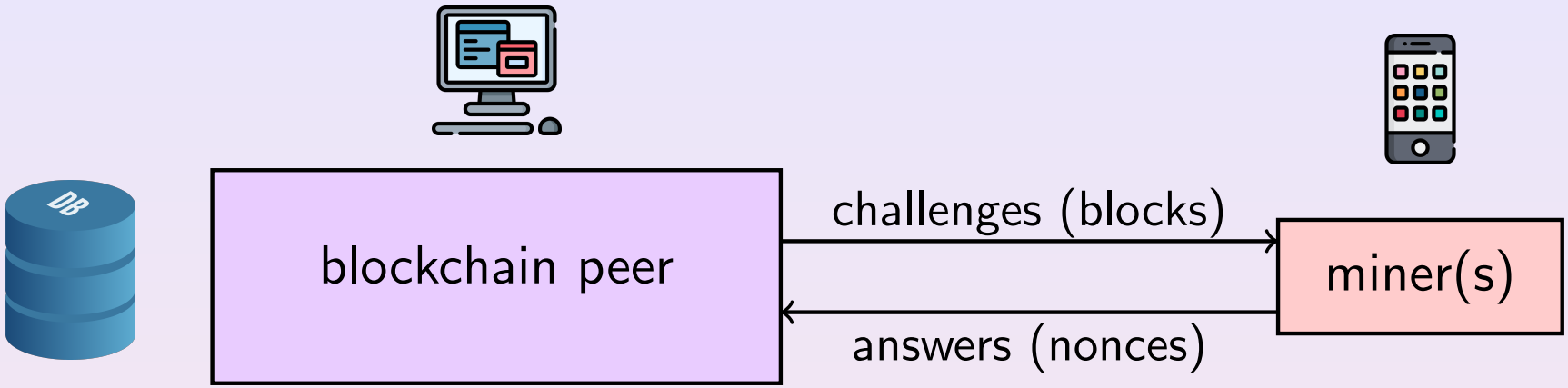
- mobile devices are a huge already existing hardware base
- a mobile application could introduce mining to a large portion of humanity, who owns a phone but not a powerful connected computer
- it can be easily and automatically updated
- it simplifies the start-up of new blockchains
- it has educational and marketing relevance

Miners as mobile applications



Can we do it for PoW (Bitcoin or old Ethereum) ?

Miners as mobile applications



Can we do it for PoW (Bitcoin or old Ethereum) ?

- no, since PoW is a CPU intensive computation
  - mobile devices are not competitive for that
  - their battery would immediately die
- no, since blocks are too large for being streamed to a mobile
  - mobile networks are relatively slow
  - data packages are still very limited for that

Enter Proof of Space

- Ateniese, Bonacina, Faonio, Galesi: *Proofs of Space: When Space Is of the Essence*, 2014 (DAG pebbling)
- Dziembowski, Faust, Kolmogorov, Pietrzak: *Proofs of Space*, 2015 (DAG pebbling) ⇒ SpaceMint: `https://github.com/kwonalbert/spacemint`
- Abusalah, Alwen, Cohen, Khilko, Pietrzak, Reyzin: *Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space*, 2017 (proof of sequential work + proof of space) ⇒ Chia: `https://www.chia.net/wp-content/uploads/2022/07/ChiaGreenPaper.pdf`
- *Signum Community Website and Documentation Project* (plot files) ⇒ Signum: `https://wiki.signum.network`

tool	consensus	network	formalized	since	status
SpaceMint	no	no	yes	2015	discontinued
Chia	yes	yes	yes	2017	active
Signum	yes	yes	no	2014	active

From Signum to Mokamint

- possibly the first ever smart contracts (before Ethereum!)
- commercial web page: `https://signum.network/`
- very informal documentation: `https://wiki.signum.network`
- chaotic, uncommented, monolithic code: `https://github.com/signum-network/signum-node`

Fausto Spoto: *A Formalization of Signum's Consensus*, 9th Workshop on Trusted Smart Contracts, Miyakojima, Japan, April 2025, LNCS 15753

Fausto Spoto: *Mokamint: Generic Energy-Efficient Consensus without Validators*, submitted for publication, 2026

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p \quad (\bowtie \text{ is byte array concatenation})$$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p$ ($\bowtie$ is byte array concatenation)

$h_0 = hash(seed)$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p$ ($\bowtie$ is byte array concatenation)

$h_0 = hash(seed)$

$h_1 = hash(h_0 \bowtie seed)$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p$ ($\bowtie$ is byte array concatenation)

$h_0 = hash(seed)$

$h_1 = hash(h_0 \bowtie seed)$

$h_2 = hash(h_1 \bowtie h_0 \bowtie seed)$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p \quad (\bowtie \text{ is byte array concatenation})$$
$$h_0 = hash(seed)$$
$$h_1 = hash(h_0 \bowtie seed)$$
$$h_2 = hash(h_1 \bowtie h_0 \bowtie seed)$$
$$\vdots$$
$$h_{8191} = hash(h_{8190} \bowtie h_{8189} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p \quad (\bowtie \text{ is byte array concatenation})$$
$$h_0 = hash(seed)$$
$$h_1 = hash(h_0 \bowtie seed)$$
$$h_2 = hash(h_1 \bowtie h_0 \bowtie seed)$$
$$\vdots$$
$$h_{8191} = hash(h_{8190} \bowtie h_{8189} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$$
$$h_{final} = hash(h_{8191} \bowtie h_{8190} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p$ ($\bowtie$ is byte array concatenation)

$h_0 = hash(seed)$

$h_1 = hash(h_0 \bowtie seed)$

$h_2 = hash(h_1 \bowtie h_0 \bowtie seed)$

$\vdots$

$h_{8191} = hash(h_{8190} \bowtie h_{8189} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$

$h_{final} = hash(h_{8191} \bowtie h_{8190} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$

reassign $h_i = h_i \oplus h_{final}$ for every $0 \leq i < 8192$ ($\oplus$ is byte array xor)

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p$ ($\bowtie$ is byte array concatenation)

$h_0 = hash(seed)$

$h_1 = hash(h_0 \bowtie seed)$

$h_2 = hash(h_1 \bowtie h_0 \bowtie seed)$

$\vdots$

$h_{8191} = hash(h_{8190} \bowtie h_{8189} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$

$h_{final} = hash(h_{8191} \bowtie h_{8190} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$

reassign $h_i = h_i \oplus h_{final}$ for every $0 \leq i < 8192$ ($\oplus$ is byte array xor)

$$nonce_p = \underbrace{h_0, h_{8191}}_{scoop_0}, \underbrace{h_2, h_{8189}}_{scoop_1}, \ldots, \underbrace{h_{8188}, h_3}_{scoop_{n-2}}, \underbrace{h_{8190}, h_1}_{scoop_{n-1}}$$

Mokamint's plotting phase

It builds a plot $nonce_0, \ldots, nonce_{n-1}$ for a peer, a miner and a chain

it must be slow and expensive

$seed = key^{peer}_{public} \bowtie key^{miner}_{public} \bowtie chainId \bowtie p$ ($\bowtie$ is byte array concatenation)

$h_0 = hash(seed)$

$h_1 = hash(h_0 \bowtie seed)$

$h_2 = hash(h_1 \bowtie h_0 \bowtie seed)$

$\vdots$

$h_{8191} = hash(h_{8190} \bowtie h_{8189} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$

$h_{final} = hash(h_{8191} \bowtie h_{8190} \bowtie \cdots \bowtie h_1 \bowtie h_0 \bowtie seed)$

reassign $h_i = h_i \oplus h_{final}$ for every $0 \leq i < 8192$ ($\oplus$ is byte array xor)

$$nonce_p = \underbrace{h_0, h_{8191}}_{scoop_0}, \underbrace{h_2, h_{8189}}_{scoop_1}, \ldots, \underbrace{h_{8188}, h_3}_{scoop_{n-2}}, \underbrace{h_{8190}, h_1}_{scoop_{n-1}}$$

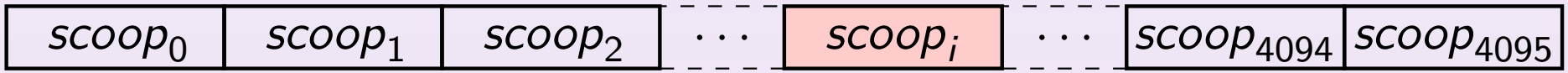
hash must be expensive (for instance: shabal256)

Mokamint's mining phase

It answers a challenge $\langle i, \gamma \rangle$ derived from the topmost block of the chain

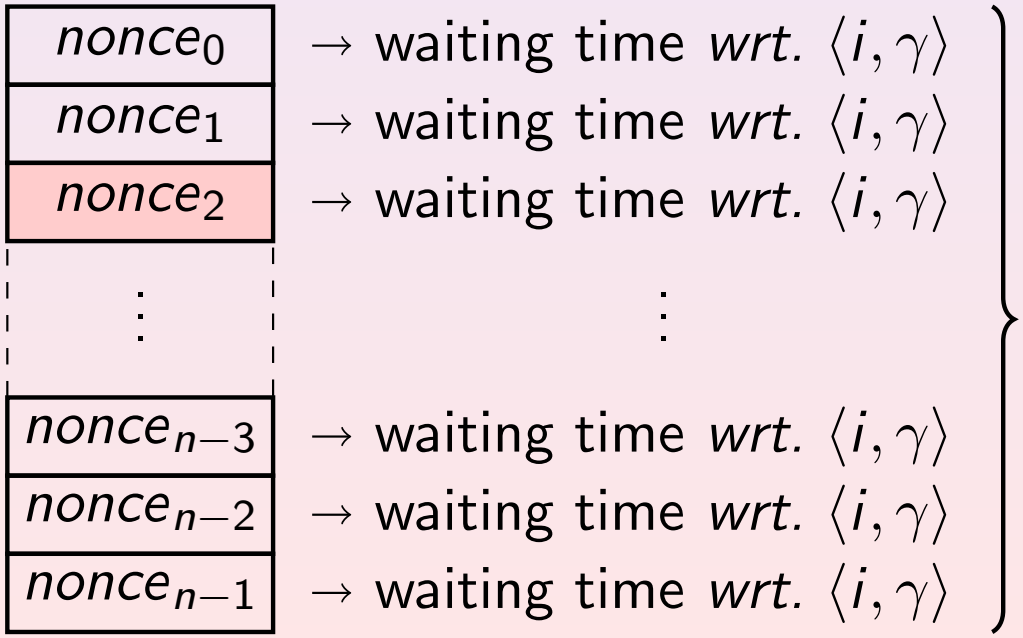
it must be cheap and quick

nonce



its waiting time *wrt.* a challenge $\langle i, \gamma \rangle$ is $hash(scoop_i \bowtie \gamma)$,

plot



min

Mokamint's Proof of Space

A large plot file is preallocated on disk (only once!)

Mokamint's Proof of Space

A large plot file is preallocated on disk (only once!)

Blocks are enriched with a *nonce*

Mokamint's Proof of Space

A large plot file is preallocated on disk (only once!)

Blocks are enriched with a *nonce*

challenge: given a *challenge* $= \langle i, \gamma \rangle$ derived from **block** find a block **next** with a *nonce* whose waiting time *wrt. challenge* is as small as possible

Mokamint's Proof of Space

A large plot file is preallocated on disk (only once!)

Blocks are enriched with a *nonce*

challenge: given a *challenge* $= \langle i, \gamma \rangle$ derived from **block** find a block **next** with a *nonce* whose waiting time *wrt. challenge* is as small as possible

answer: reply with a **next** that contains the *nonce* from the plot file that minimizes its waiting time *wrt. challenge*

Mokamint's Proof of Space

A large plot file is preallocated on disk (only once!)

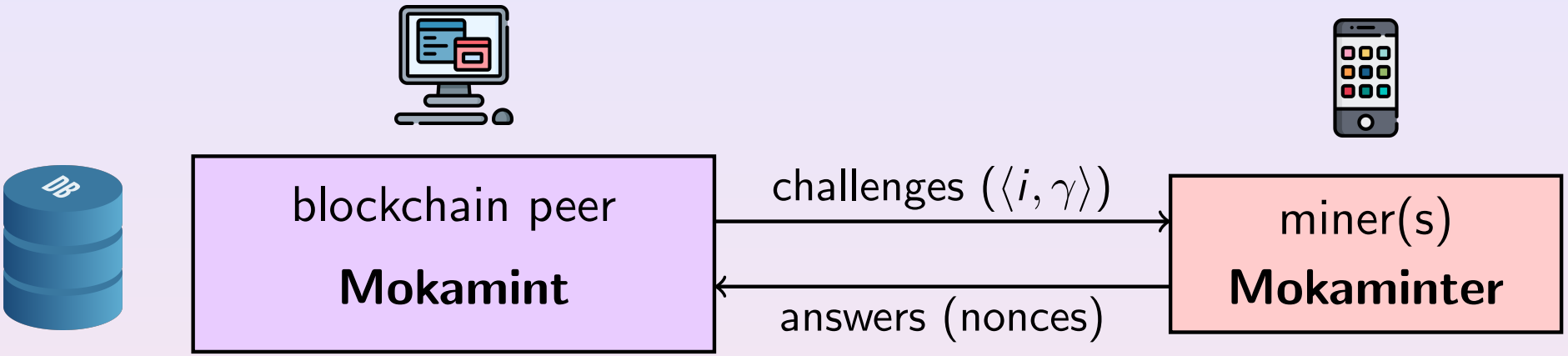
Blocks are enriched with a *nonce*

challenge: given a *challenge* $= \langle i, \gamma \rangle$ derived from **block** find a block **next** with a *nonce* whose waiting time *wrt. challenge* is as small as possible

answer: reply with a **next** that contains the *nonce* from the plot file that minimizes its waiting time *wrt. challenge*

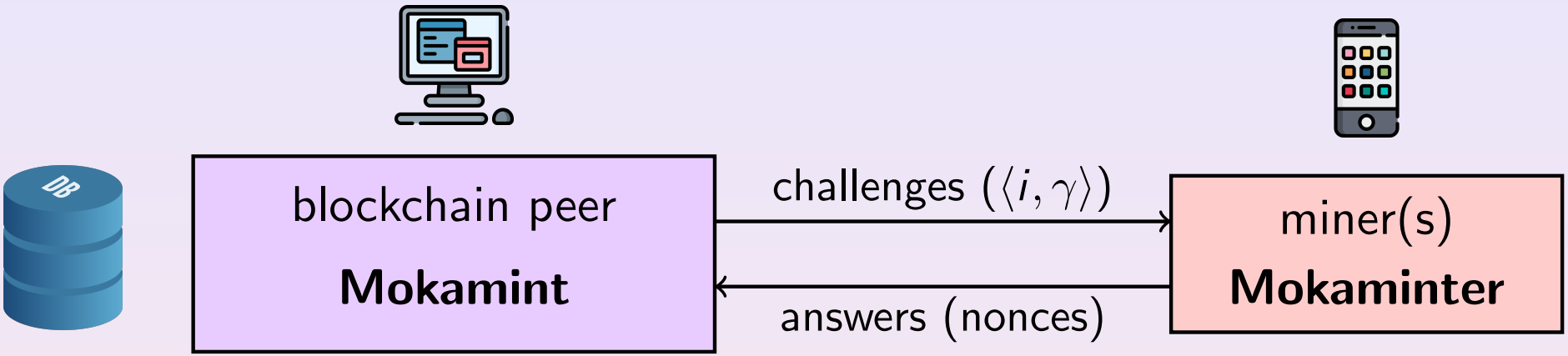
competition: in case of more **next**, select the one whose nonce has the smallest waiting time *wrt. challenge*

Miners as mobile applications, again



Can we do it for PoSp (Mokamint, at least) ?

Miners as mobile applications, again



Can we do it for PoSp (Mokamint, at least) ?

- yes, since PoSp uses almost no CPU during the mining phase
  - this preserves the battery of the device
- yes, since challenges are a around 64 bytes long and nonces can be compacted into *deadlines* of around 200 bytes
  - this limits data throughput during mining
- yes, average mobile devices have around 256 *GB* of SSD, compared to an average of 1000 *GB* of SSD of desktop computers
  - mining can be competitive on a mobile device

Experiments: plotting phase

it must be slow and expensive

device	type	nonces	size	time
Xiaomi 11 Lite 5G NE	phone	1000	0.26 GB	1m13s
Xiaomi 11 Lite 5G NE	phone	5000	1.28 GB	6m03s
Xiaomi 11 Lite 5G NE	phone	10000	2.56 GB	12m05s
Samsung Galaxy Tab S6 Lite	tablet	1000	0.26 GB	2m15s
Samsung Galaxy Tab S6 Lite	tablet	5000	1.28 GB	11m12s
Samsung Galaxy Tab S6 Lite	tablet	10000	2.56 GB	22m21s
Intel NUC8i5BEH	desktop	1000	0.26 GB	20s
Intel NUC8i5BEH	desktop	5000	1.28 GB	1m40s
Intel NUC8i5BEH	desktop	10000	2.56 GB	3m24s

(today, the average plot size in the Hotmoka mainnet is 2257675 nonces)

Experiments: mining phase

it must be cheap and quick

type	CPU	RAM	network	answer	earned crypto
phone	0.56%	2.4% (192 MB)	3.36 MB	0.207s	373.30 mokas
tablet	0.23%	3.5% (140 MB)	3.38 MB	0.225s	266.63 mokas
desktop	0.09%	1.4% (224 MB)	3.39 MB	0.019s	319.95 mokas

(one day of mining with a plot of 10000 nonces)

In the remaining (*blockrate* – **answer**) seconds the miner does nothing:

- no CPU activity
- no disk activity

Conclusion

Mokaminter is the only existing miner *in* a mobile device

- it works for any blockchain built over the Mokamint generic consensus engine based on PoSp, such as Hotmoka
- experiments with UFMA students in Brazil show that it increases the battery usage of just 0.27% on average
- it works even with the free on board wifi of an international flight
- currently only for Android

Install Mokaminter from Google Play and start mining for Hotmoka:
`https://play.google.com/store/apps/details?id=io.mokamint.android.mokaminter`