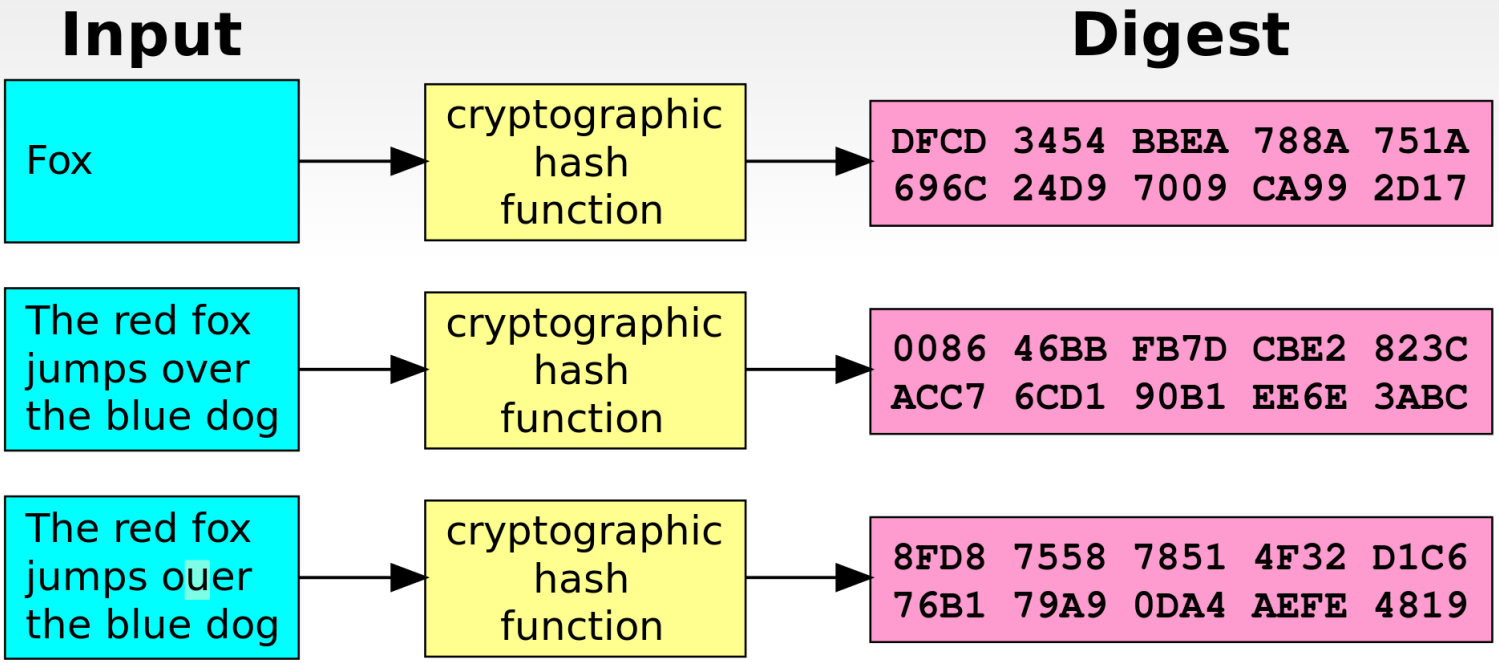
Sistemas Peer-to-Peer

6. Merkle Trees

Francisco Sant'Anna
francisco@ime.uerj.br

Função Hash Criptográfica

Função Hash Criptográfica



Função Hash Criptográfica

Função Hash Criptográfica

- Determinística:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`

Função Hash Criptográfica

- Determinística:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
- "One-way" (não inversível):
  - H^{-1}`("DFCD BBEA ... CA99 2D17") → ???`

Função Hash Criptográfica

- Determinística:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
- "One-way" (não inversível):
  - H^{-1}`("DFCD BBEA ... CA99 2D17") → ???`
- Sem colisões:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H(~~?????~~) → "DFCD BBEA ... CA99 2D17"`

Função Hash Criptográfica

- Determinística:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
- "One-way" (não inversível):
  - H^{-1}`("DFCD BBEA ... CA99 2D17") → ???`
- Sem colisões:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H(~~?????~~) → "DFCD BBEA ... CA99 2D17"`
- Caótica:
  - `H("Fox") → "DFCD BBEA ... CA99 2D17"`
  - `H("Foy") → "1200 C78A ... EF1A D99B"`

Aplicações

Aplicações

- Integridade de um recurso
  - verificar o hash após recebimento

Aplicações

- Integridade de um recurso
  - verificar o hash após recebimento
- Identificação pequena e única de um recurso
  - o hash é a própria identificação

Aplicações

- Integridade de um recurso
  - verificar o hash após recebimento
- Identificação pequena e única de um recurso
  - o hash é a própria identificação
- Assinaturas, senhas, provas de trabalhos, etc

Merkle Trees

Merkle Trees

- **Hash Lists**, Hash DAGs

Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat* ← *Hello!* ← *Welcome!* ← ...

Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat* ← *Hello!* ← *Welcome!* ← ...

- `H("#chat")` → `A95B969D…`

Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat* ← *Hello!* ← *Welcome!* ← ...

- `H("#chat")` → `A95B969D…`

Merkle Trees

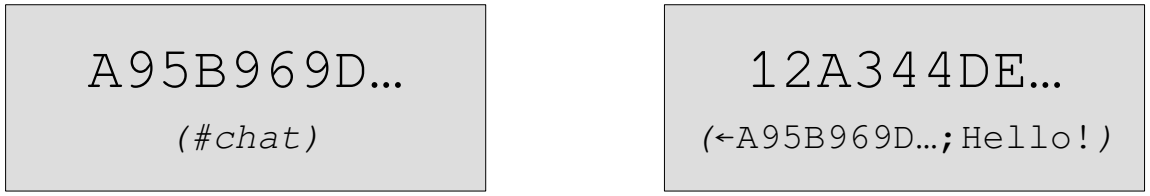
- **Hash Lists**, Hash DAGs
- *#chat* ← *Hello!* ← *Welcome!* ← ...

- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`

Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat* ← *Hello!* ← *Welcome!* ← ...

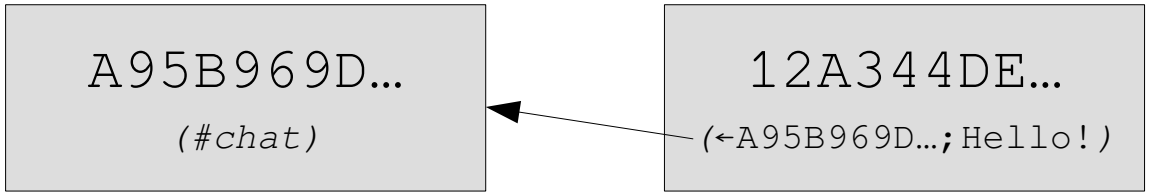
- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`



Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat ← Hello! ← Welcome! ← …*

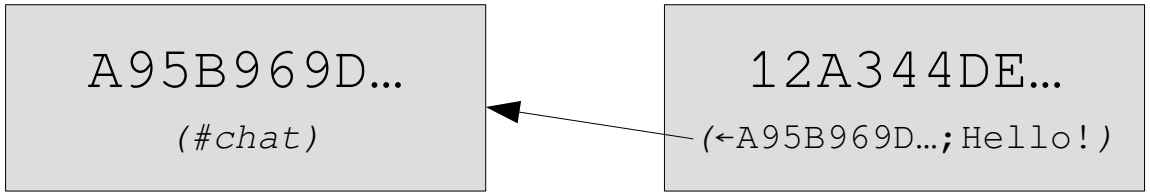
- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`



Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat ← Hello! ← Welcome! ← …*

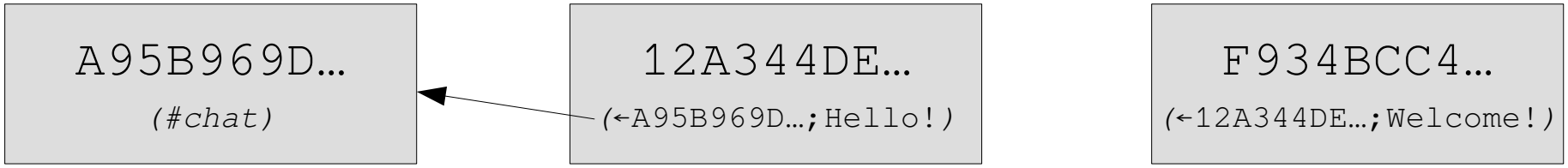
- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`
- `H("←12A344DE… ; Welcome!")` → `F934BCC4…`



Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat ← Hello! ← Welcome! ← …*

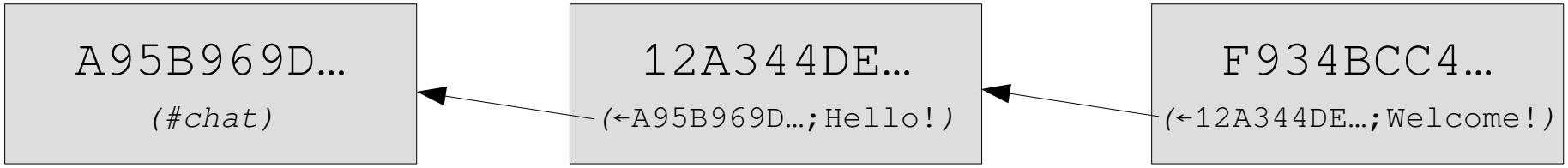
- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`
- `H("←12A344DE… ; Welcome!")` → `F934BCC4…`



Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat ← Hello! ← Welcome! ← …*

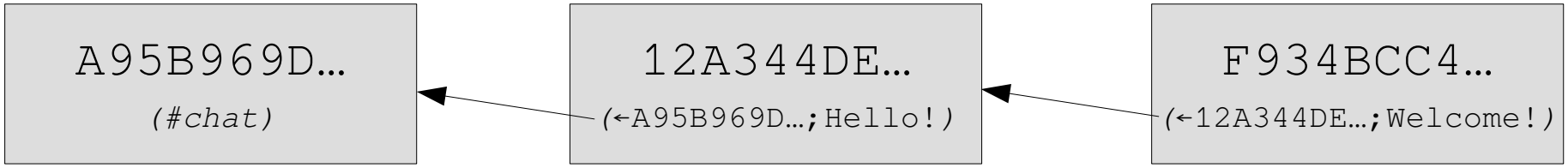
- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`
- `H("←12A344DE… ; Welcome!")` → `F934BCC4…`



Merkle Trees

- **Hash Lists**, Hash DAGs
- *#chat ← Hello! ← Welcome! ← …*

- `H("#chat")` → `A95B969D…`
- `H("←A95B969D… ; Hello!")` → `12A344DE…`
- `H("←12A344DE… ; Welcome!")` → `F934BCC4…`



O hash `F934BCC4…` representa univocamente a lista inteira: a lista é verificável e íntegra

Merkle Trees

- Sincronização de dados em sistemas P2P

Merkle Trees

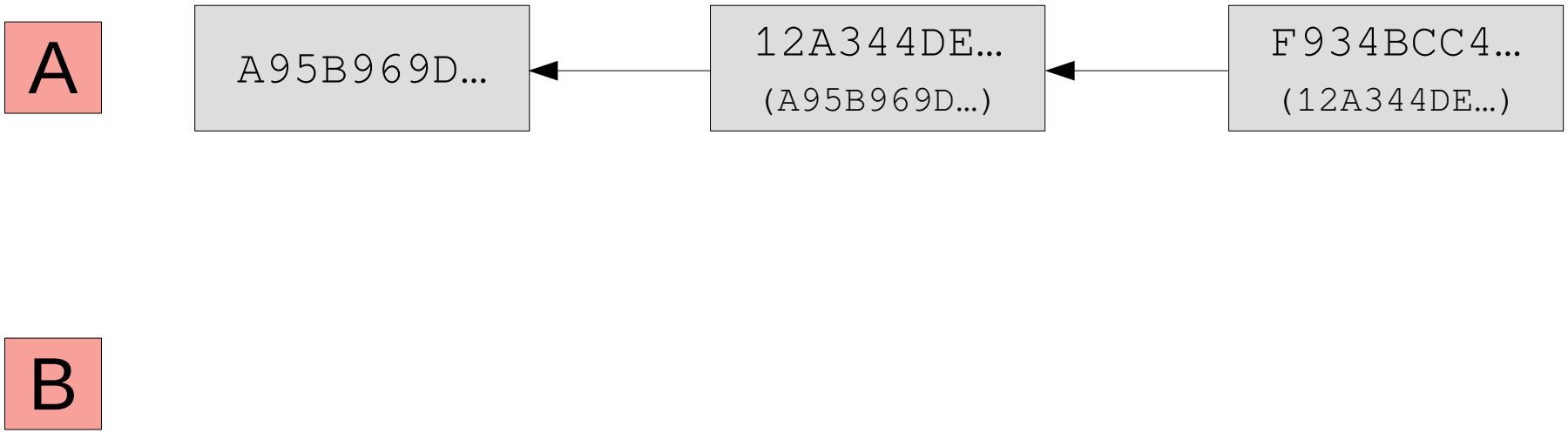
- Sincronização de dados em sistemas P2P

A

B

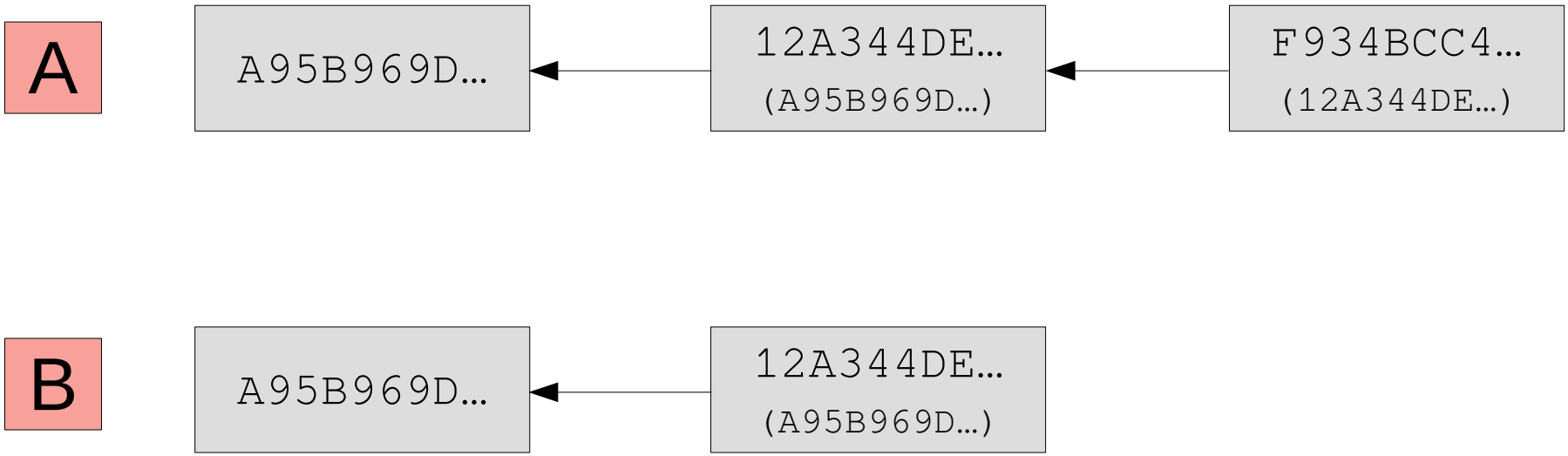
Merkle Trees

- Sincronização de dados em sistemas P2P



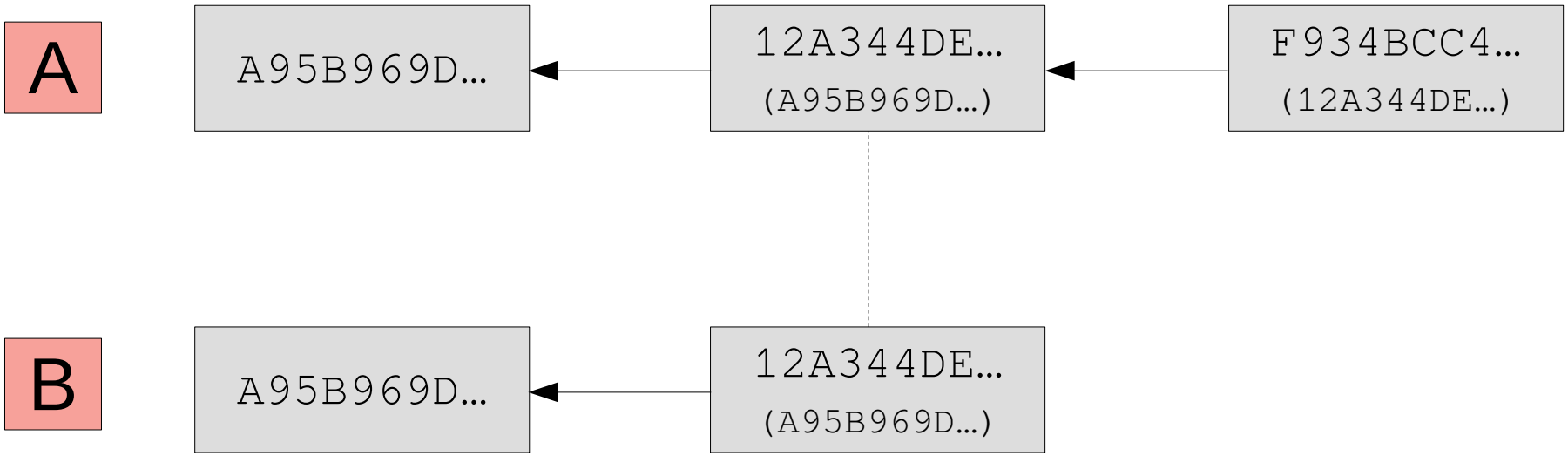
Merkle Trees

- Sincronização de dados em sistemas P2P



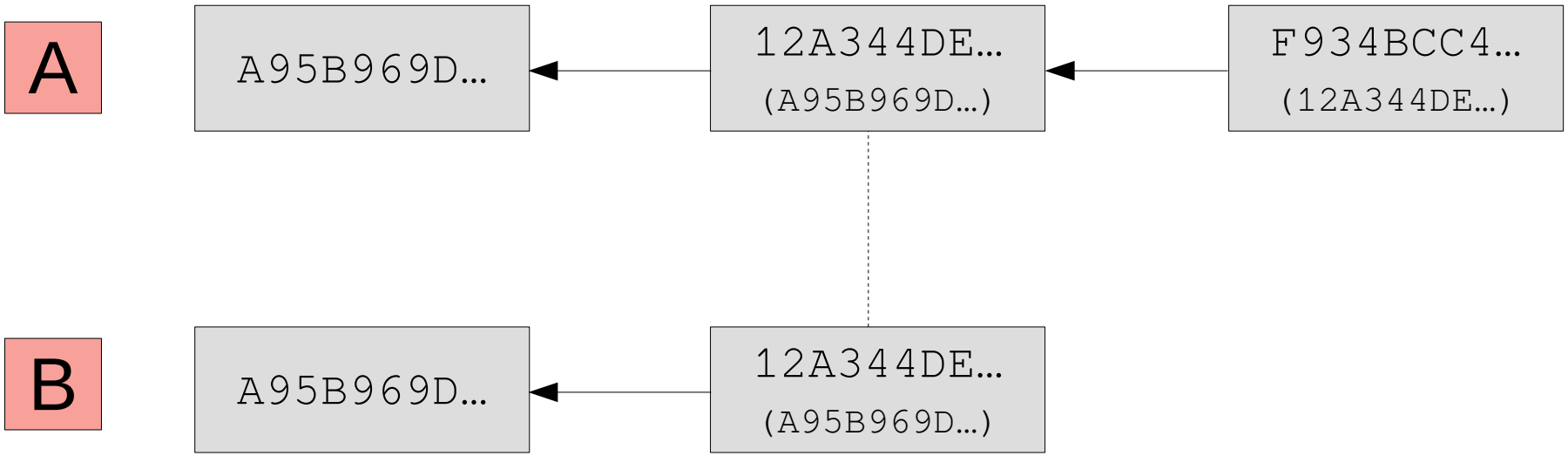
Merkle Trees

- Sincronização de dados em sistemas P2P



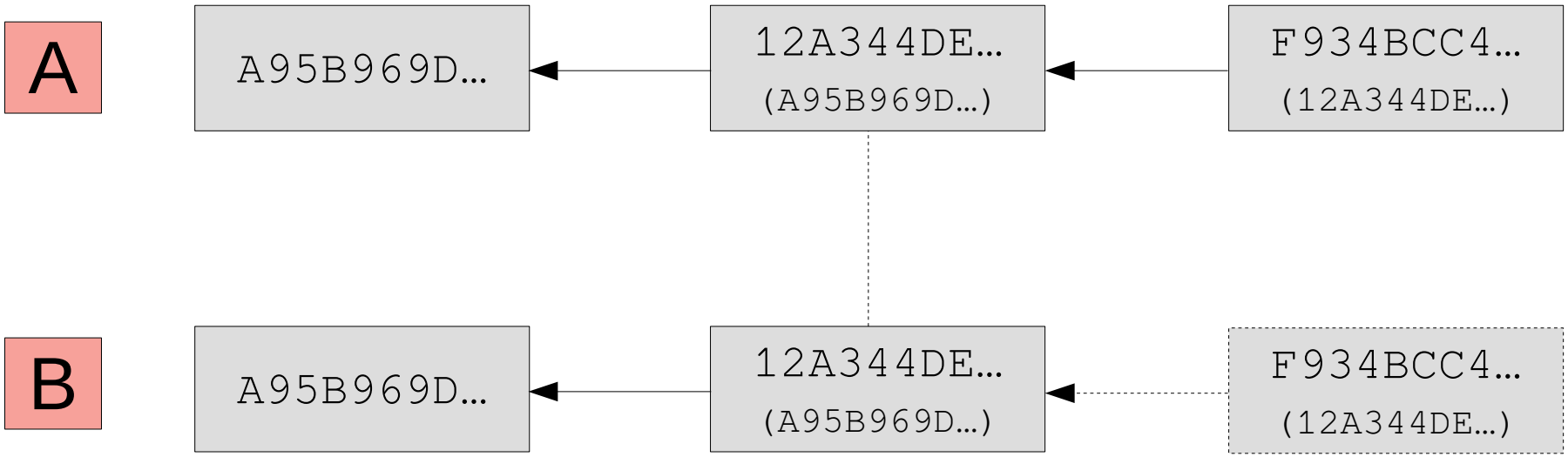
Merkle Trees

- Sincronização de dados em sistemas P2P
  - Basta achar o último em comum?



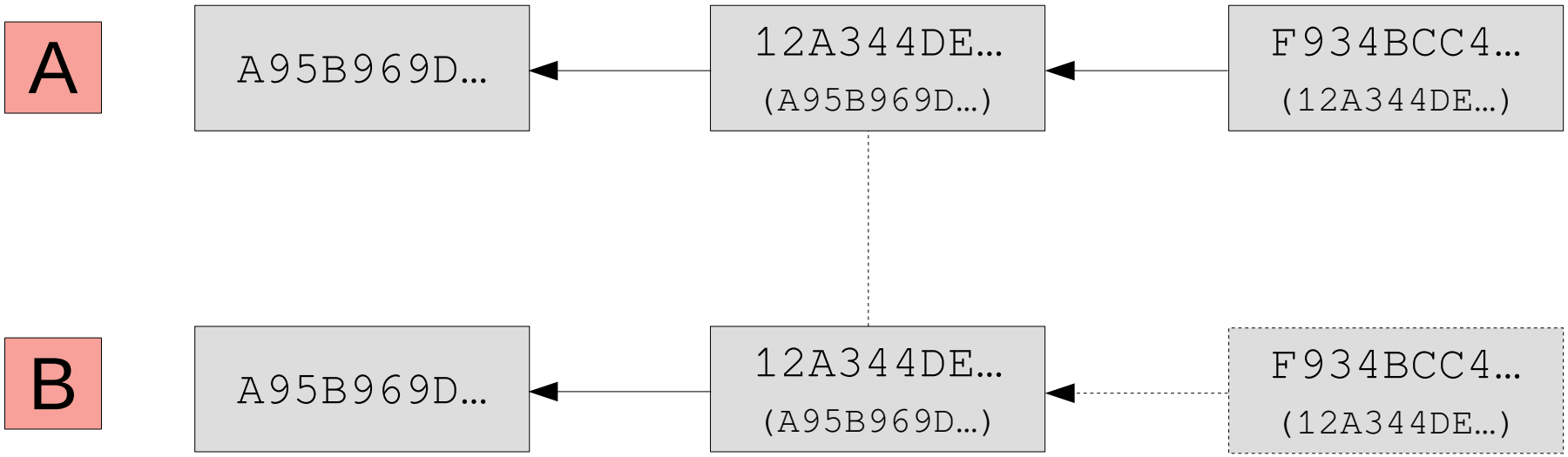
Merkle Trees

- Sincronização de dados em sistemas P2P
  - Basta achar o último em comum?



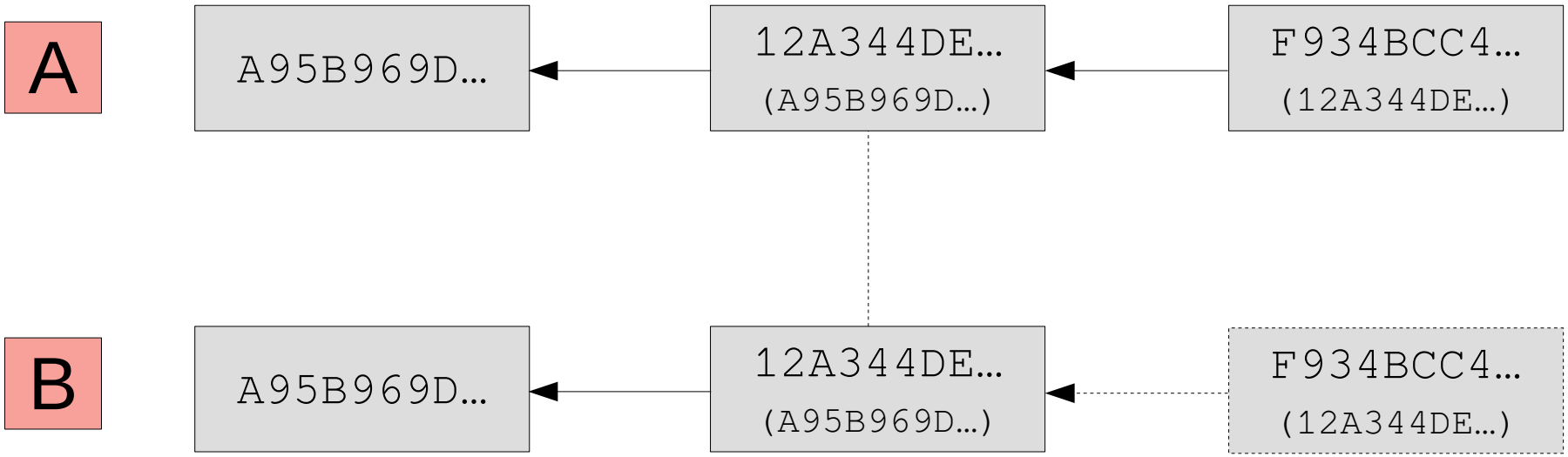
Merkle Trees

- Sincronização de dados em sistemas P2P
  - Basta achar o último em comum?
  - Basta transmitir até o último em comum?



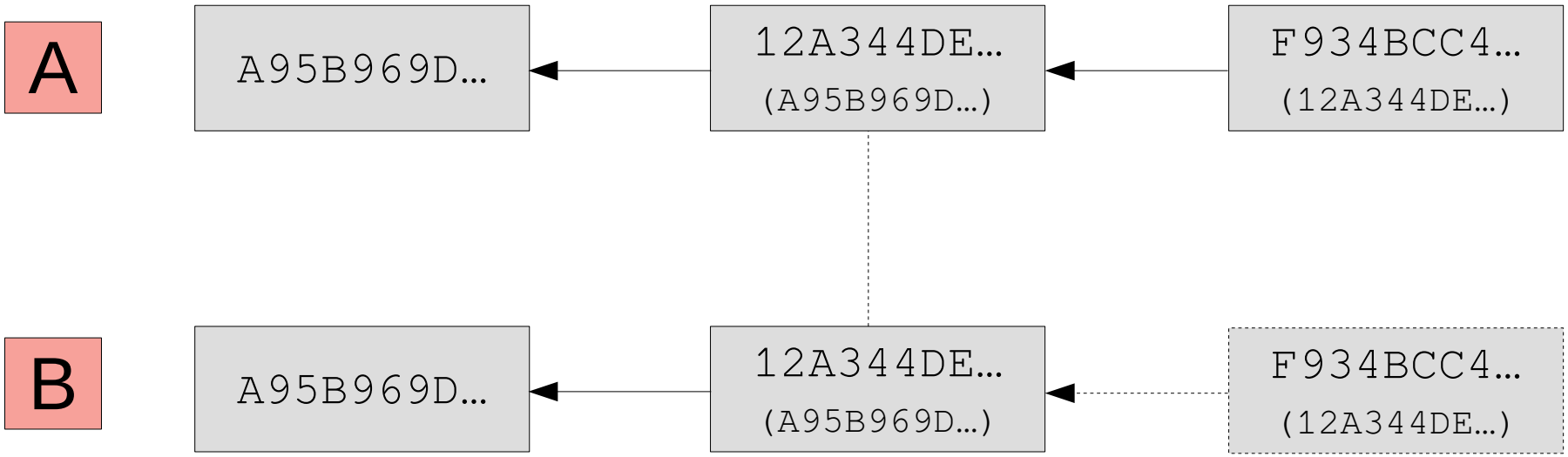
Merkle Trees

- Sincronização de dados em sistemas P2P
  - Basta achar o último em comum?
  - Basta transmitir até o último em comum?
  - É possível o nó A ter outros blocos até o gênesis?



Merkle Trees

- Sincronização de dados em sistemas P2P
  - Basta achar o último em comum?
  - Basta transmitir até o último em comum?
  - É possível o nó A ter outros blocos até o gênesis?
  - É possível o nó A alterar algum bloco?



Merkle Trees

Merkle Trees

- *"Hash trees can be used to verify any kind of data stored, handled and transferred in and between computers. They can help ensure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks."* [Wikipedia on Merkle Trees]

Merkle Trees

- *"Hash trees can be used to verify any kind of data stored, handled and transferred in and between computers. They can help ensure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks."* [Wikipedia on Merkle Trees]

- Sistemas que utilizam Merkle Trees:

Merkle Trees

- *"Hash trees can be used to verify any kind of data stored, handled and transferred in and between computers. They can help ensure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks."* [Wikipedia on Merkle Trees]

- Sistemas que utilizam Merkle Trees:
  - IPFS, Dat, Git, Bitcoin, Cassandra,

    Gnutella, Scuttlebutt, Freechains

Sistemas Peer-to-Peer

6. Merkle Trees

Francisco Sant'Anna
francisco@ime.uerj.br