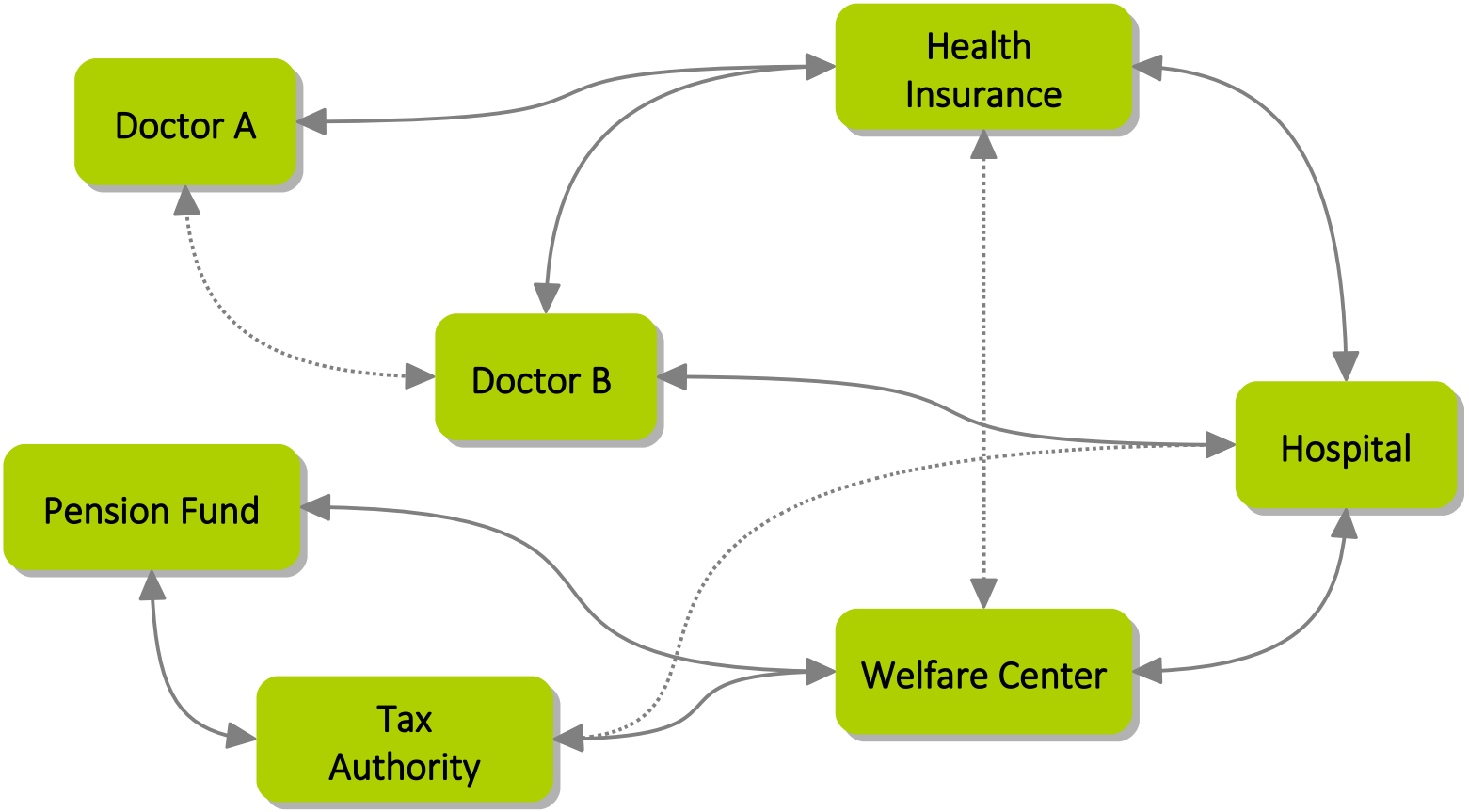
(Un)linkable Pseudonyms for Governmental Databases

Jan Camenisch and Anja Lehmann
IBM Research Zurich

How to maintain related yet distributed data?

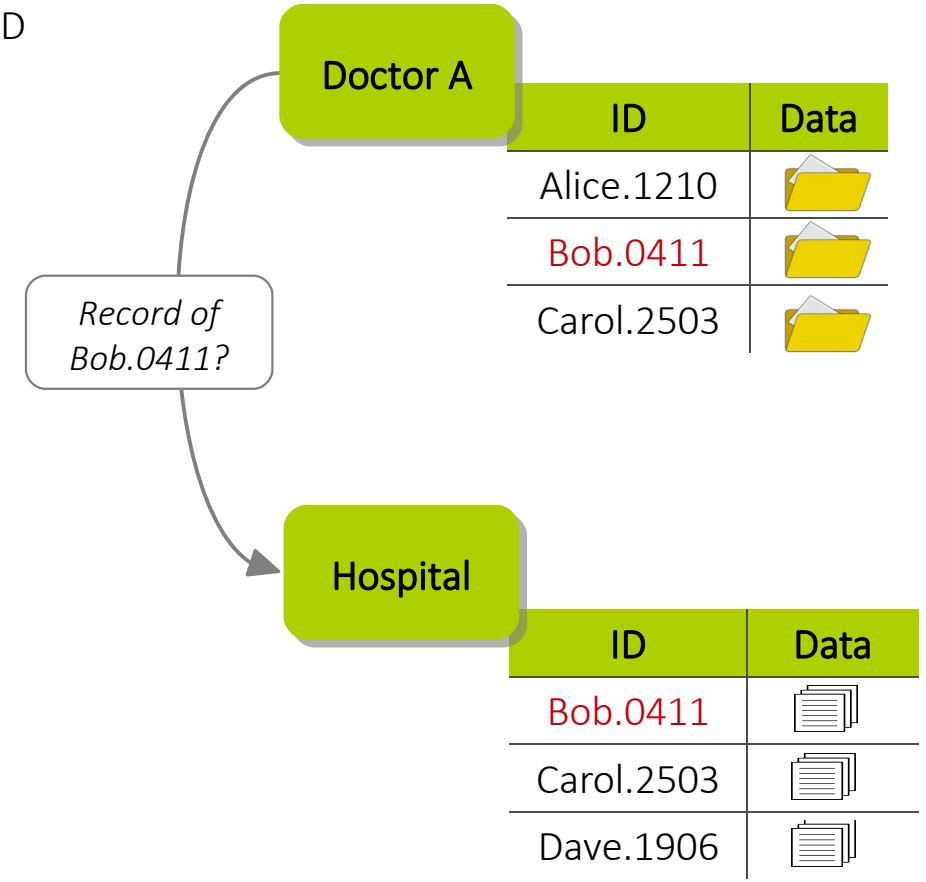
use case: social security system

- different entities maintain data of citizens
- eventually data needs to be exchanged or correlated



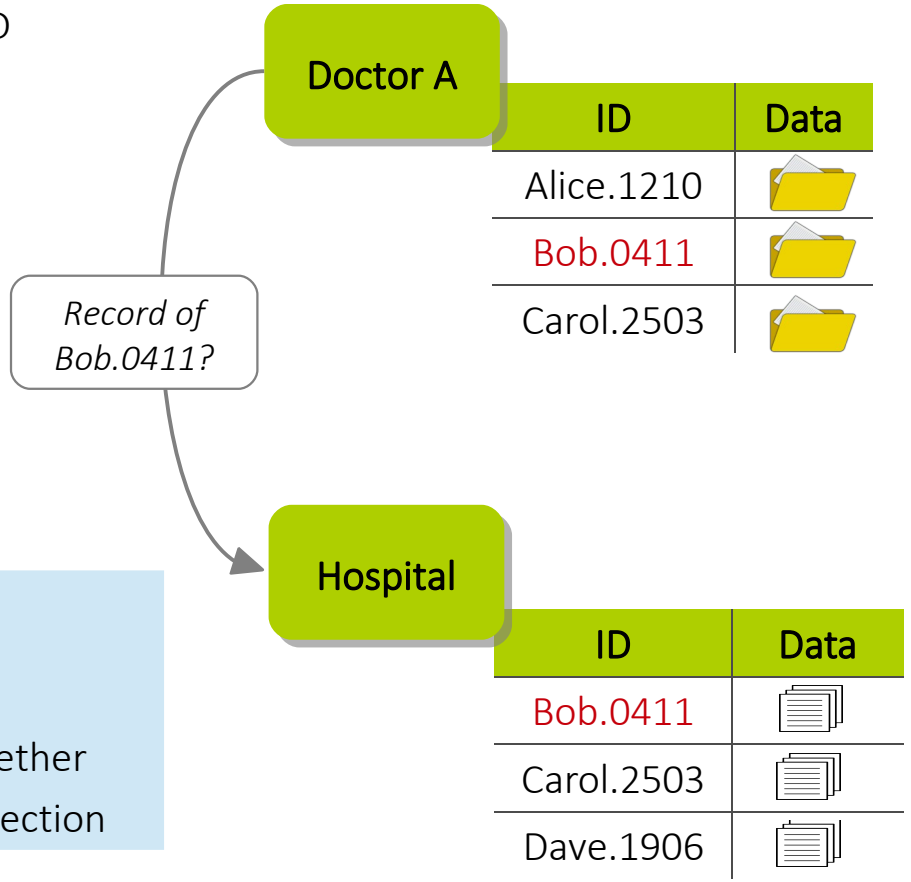
Globally Unique Identifier

- user data is associated with globally unique identifier
  - e.g., social security number, insurance ID
- different entities can easily share & link related data records



Globally Unique Identifier

- user data is associated with globally unique identifier
  - e.g., social security number, insurance ID
- different entities can easily share & link related data records

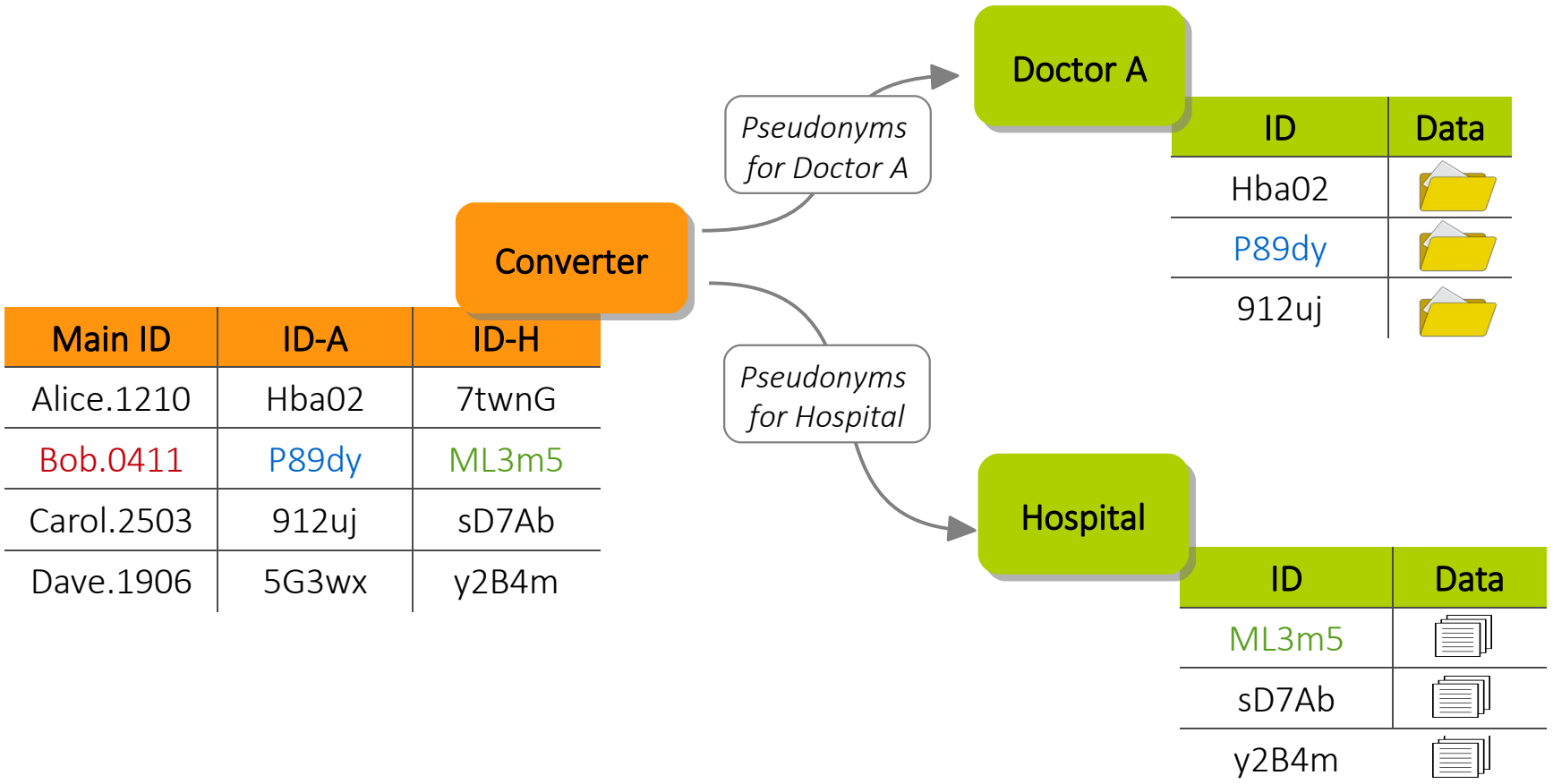


\+ simple data exchange

– no control about data exchange
– if records are lost, pieces can be linked together
– data has high-value → requires strong protection

Local Pseudonyms & *Trusted* Converter

- central converter derives independent server-local identifiers from unique identifer
- user data is associated with (unlinkable) server-local identifiers *aka "pseudonyms"*



Main ID	ID-A	ID-H
Alice.1210	Hba02	7twnG
Bob.0411	P89dy	ML3m5
Carol.2503	912uj	sD7Ab
Dave.1906	5G3wx	y2B4m

Local Pseudonyms & *Trusted* Converter

- central converter derives independent server-local identifiers from unique identifer
- user data is associated with (unlinkable) server-local identifiers *aka "pseudonyms"*
- only converter can link & convert pseudonyms
  → central hub for data exchange

Converter

Main ID	ID-A	ID-H
Alice.1210	Hba02	7twnG
Bob.0411	P89dy	ML3m5
Carol.2503	912uj	sD7Ab
Dave.1906	5G3wx	y2B4m

Doctor A

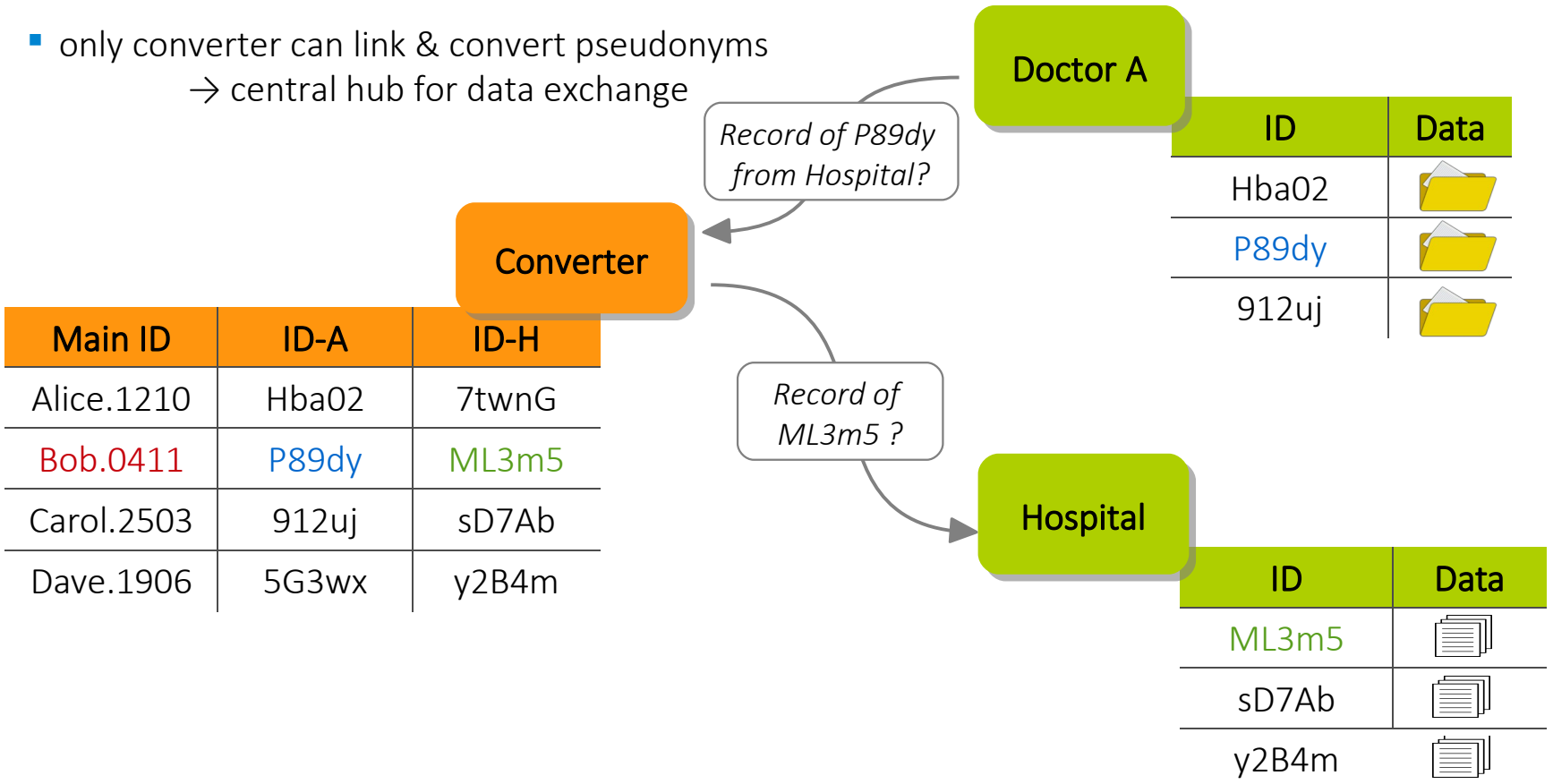
ID	Data
Hba02	
P89dy	
912uj	

Hospital

ID	Data
ML3m5	
sD7Ab	
y2B4m	

Local Pseudonyms & *Trusted* Converter

- central converter derives independent server-local identifiers from unique identifer
- user data is associated with (unlinkable) server-local identifiers *aka "pseudonyms"*
- only converter can link & convert pseudonyms
  → central hub for data exchange



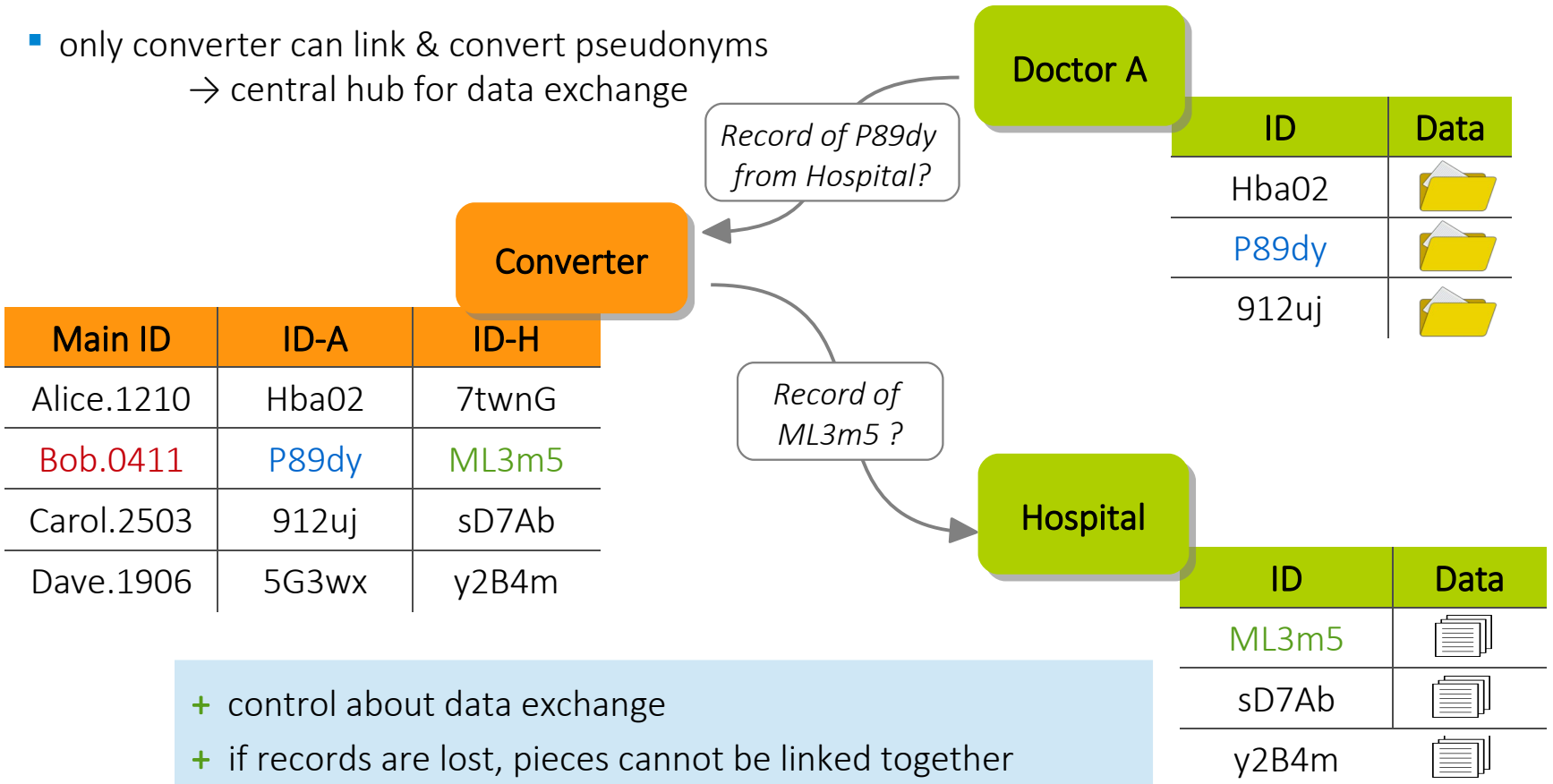
ID	Data
Hba02	
P89dy	
912uj	

Main ID	ID-A	ID-H
Alice.1210	Hba02	7twnG
Bob.0411	P89dy	ML3m5
Carol.2503	912uj	sD7Ab
Dave.1906	5G3wx	y2B4m

ID	Data
ML3m5	
sD7Ab	
y2B4m	

Local Pseudonyms & *Trusted* Converter

- central converter derives independent server-local identifiers from unique identifer
- user data is associated with (unlinkable) server-local identifiers *aka "pseudonyms"*
- only converter can link & convert pseudonyms
  → central hub for data exchange



ID	Data
Hba02	
P89dy	
912uj	

Main ID	ID-A	ID-H
Alice.1210	Hba02	7twnG
Bob.0411	P89dy	ML3m5
Carol.2503	912uj	sD7Ab
Dave.1906	5G3wx	y2B4m

ID	Data
ML3m5	
sD7Ab	
y2B4m	

\+ control about data exchange

\+ if records are lost, pieces cannot be linked together

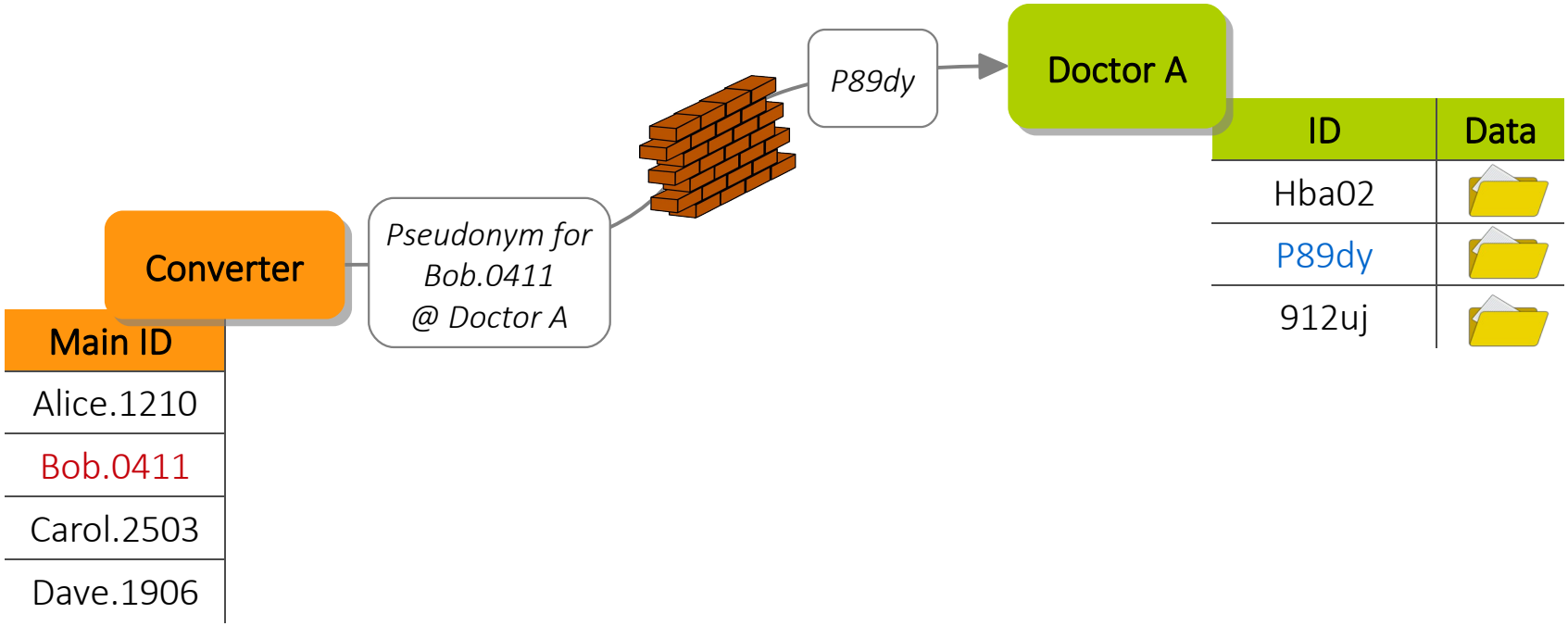
– converter learns all request & knows all correlations

our work:

(un)linkable pseudonyms but without *trusted* converter

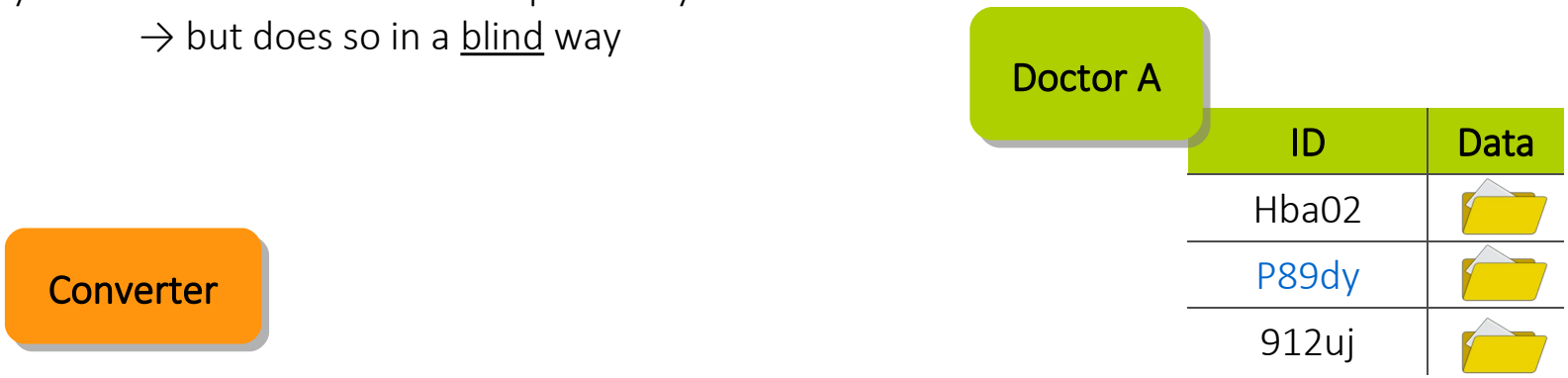
(Un)linkable Pseudonyms | Pseudonym Generation

- converter & servers jointly derive pseudonyms from unique identifers
  - servers do not learn unique identifiers, converter does not learn the pseudonyms



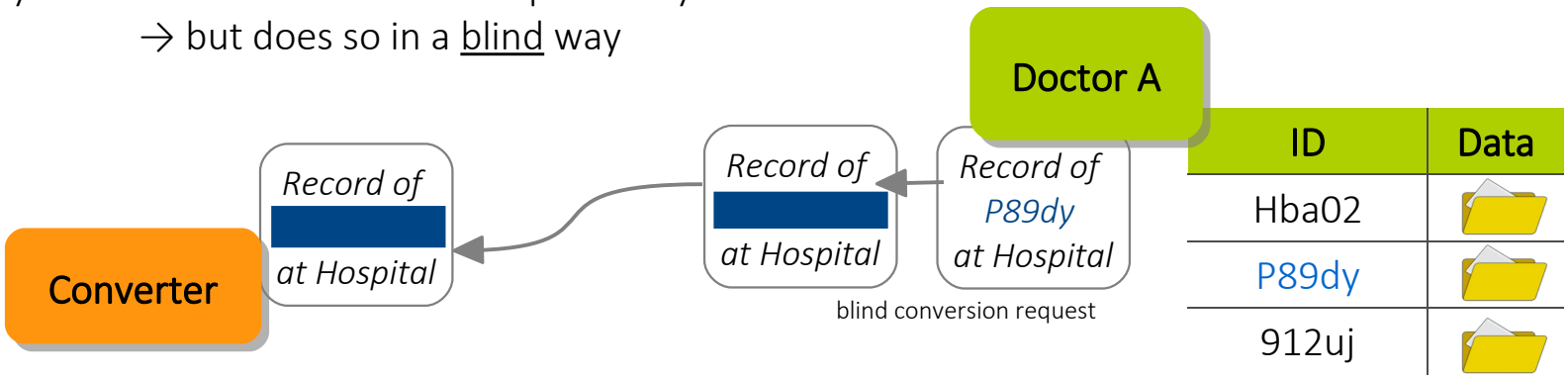
(Un)linkable Pseudonyms | Pseudonym Conversion

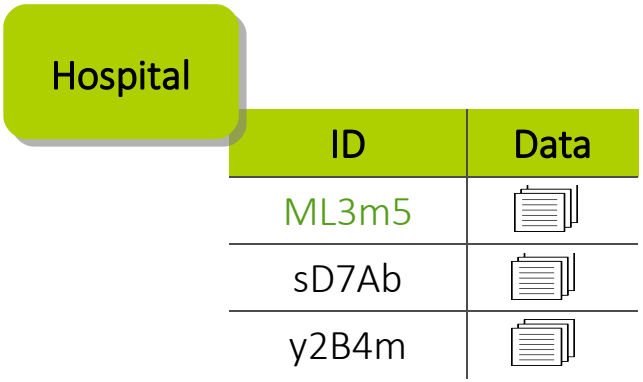
- converter & servers jointly derive pseudonyms from unique identifers
  - servers do not learn unique identifiers, converter does not learn the pseudonyms
- only converter can link & convert pseudonyms
  → but does so in a blind way



(Un)linkable Pseudonyms | Pseudonym Conversion

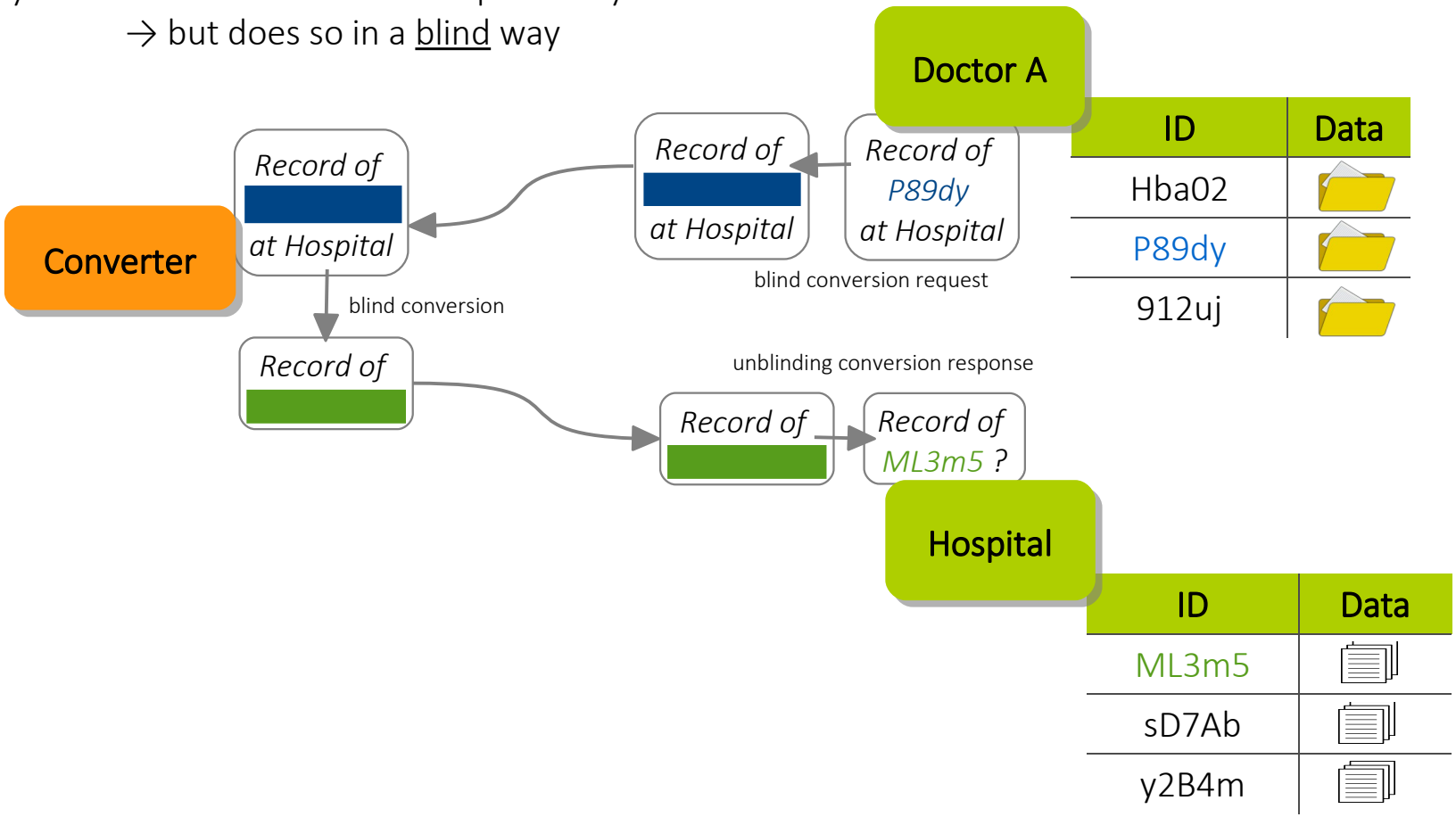
- converter & servers jointly derive pseudonyms from unique identifers
  - servers do not learn unique identifiers, converter does not learn the pseudonyms
- only converter can link & convert pseudonyms
  → but does so in a blind way





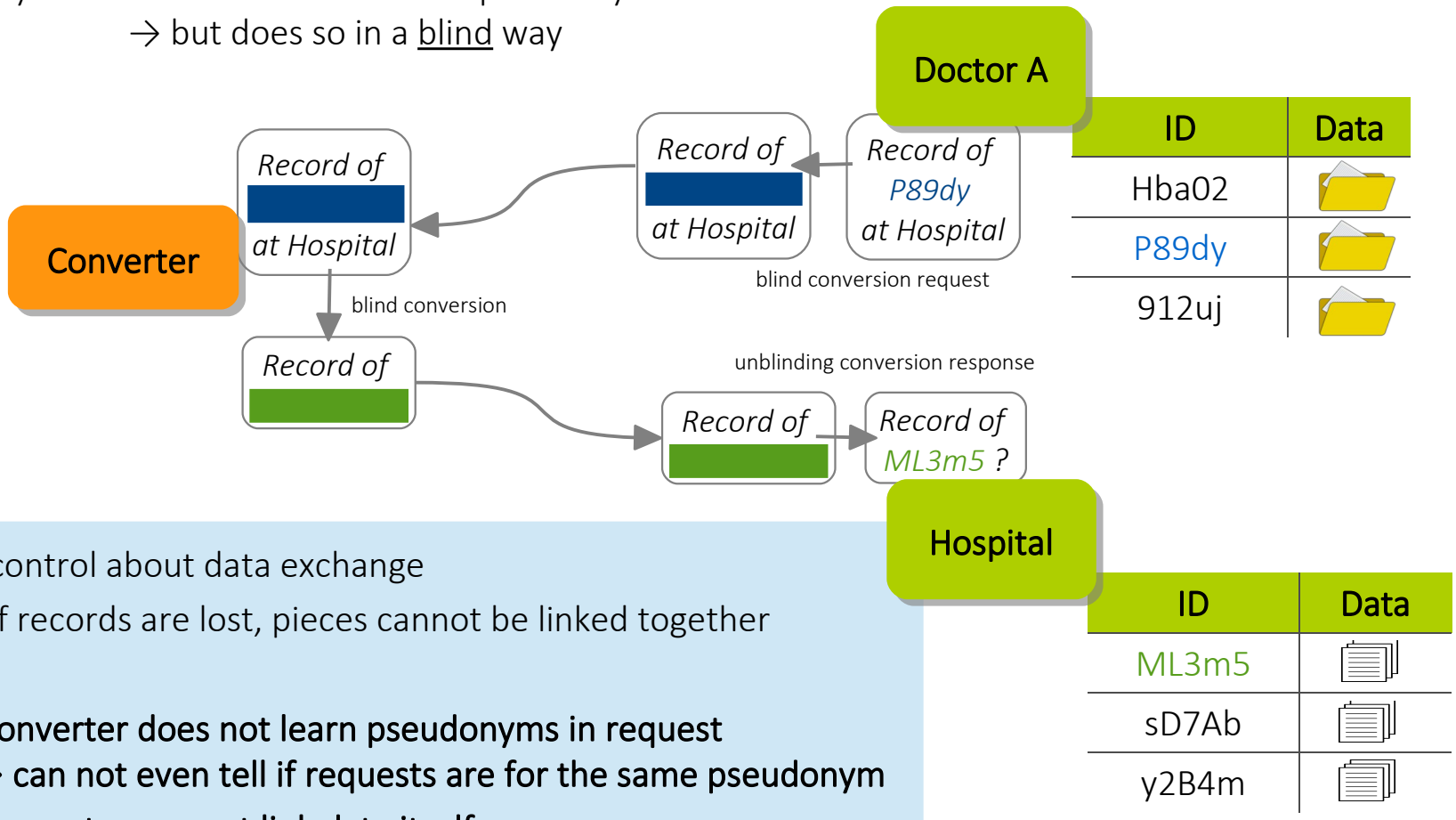
(Un)linkable Pseudonyms | Pseudonym Conversion

- converter & servers jointly derive pseudonyms from unique identifers
  - servers do not learn unique identifiers, converter does not learn the pseudonyms
- only converter can link & convert pseudonyms
  → but does so in a blind way



(Un)linkable Pseudonyms | Security

- converter & servers jointly derive pseudonyms from unique identifers
  - servers do not learn unique identifiers, converter does not learn the pseudonyms
- only converter can link & convert pseudonyms
  → but does so in a blind way



+ control about data exchange
+ if records are lost, pieces cannot be linked together

+ converter does not learn pseudonyms in request
  → can not even tell if requests are for the same pseudonym
+ converter can not link data itself

(Un)linkable Pseudonyms | Construction

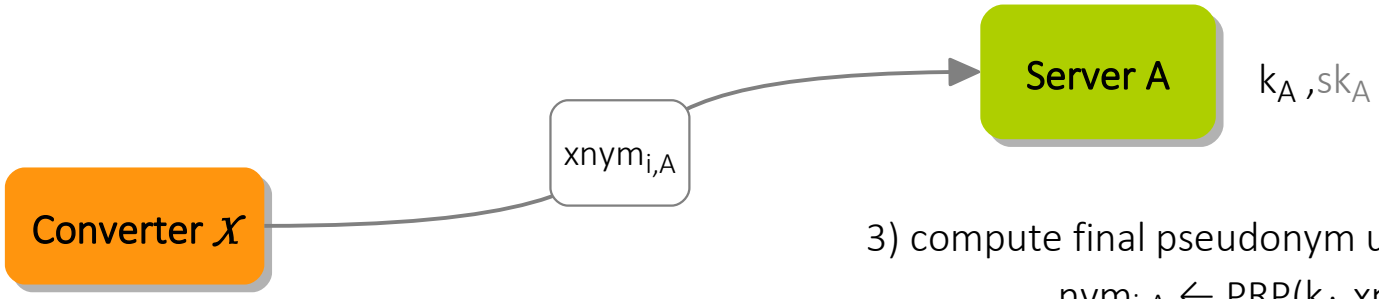
- security formally defined in the Universal Composability (UC) framework
  - ideal functionality describing the optimal behaviour of such a system
  - converter and servers can be fully corrupt

- provably secure construction based on
  - homomorphic encryption scheme (ElGamal encryption)
  - verifiable pseudorandom function (Dodis-Yampolskiy-PRF)
  - pseudorandom permutation (“lazy sampling”)
  - dual-mode and standard signature schemes (AGOT+, Schnorr signatures)
  - zero-knowledge proofs (Fiat-Shamir NIZKs with trapdoored ElGamal)

High-level Idea | Pseudonym Generation

- converter $\mathcal{X}$ and server S_A jointly compute a pseudonym $nym_{i,A}$ for user uid_i

  $\mathcal{X}$'s input: unique user-id uid_i and server ID S_A



k, $sk_{\mathcal{X}}$, for each server: x_A, x_B, x_C, …

3) compute final pseudonym using a secret key k_A

$nym_{i,A} \leftarrow PRP(k_A, xnym_{i,A})$

1) compute global core identifier using secret key k

$z_i \leftarrow PRF(k, uid_i)$

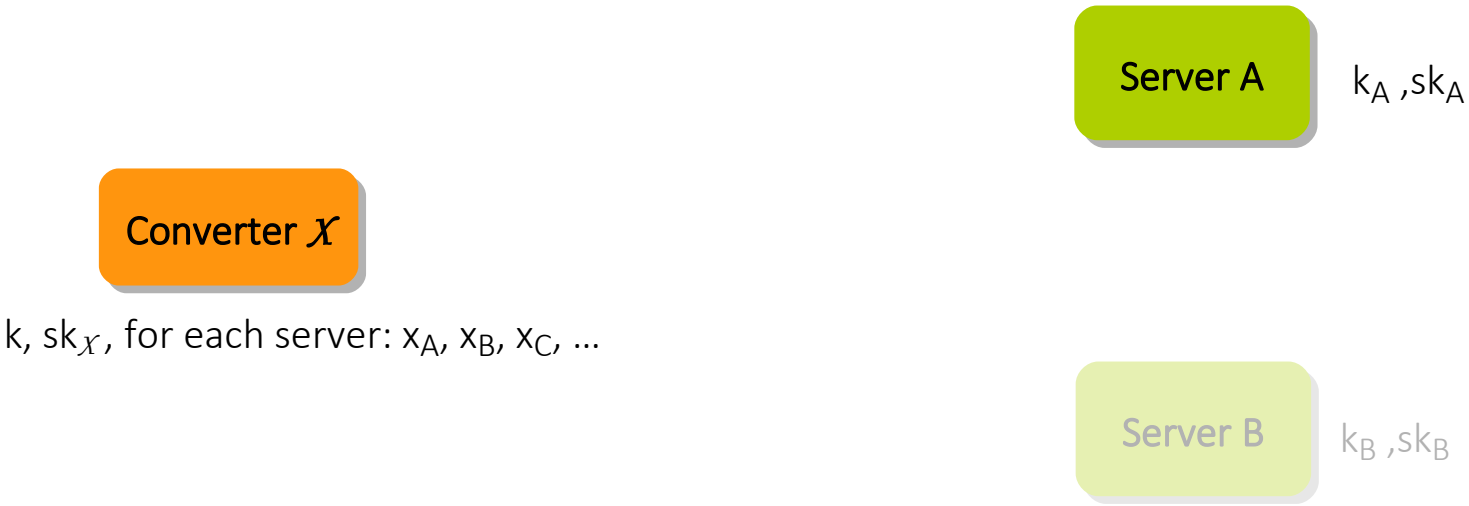
2) compute server-local "inner" pseudonym using server-specific secret key x_A

$xnym_{i,A} \leftarrow z_i^{x_A}$

High-level Idea | Pseudonym Conversion

- server S_A wishes to convert a pseudonym $nym_{i,A}$ for server S_B

  S_A's input: $nym_{i,A}$, S_B, qid



High-level Idea | Pseudonym Conversion

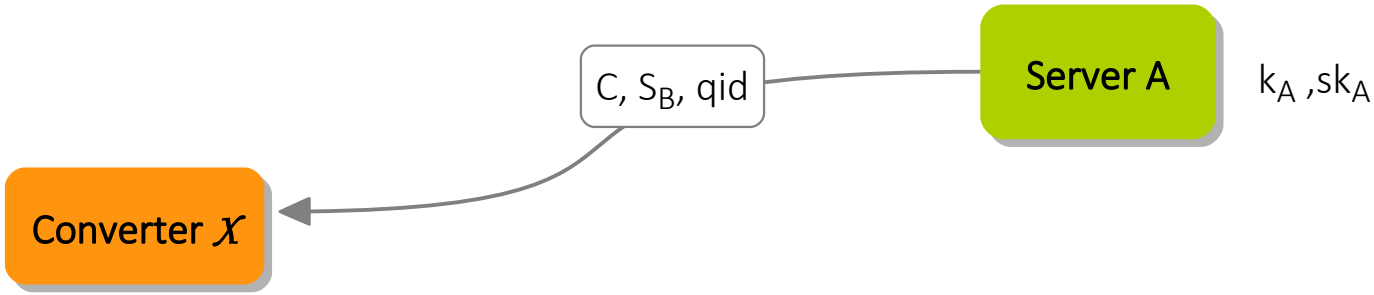
- server S_A wishes to convert a pseudonym $nym_{i,A}$ for server S_B

  S_A's input: $nym_{i,A}$, S_B, qid

1) reobtain $xnym_{i,A} \leftarrow PRP^{-1}(k_A, nym_{i,A})$

2) encrypt $xnym_{i,A}$ under S_B's and Converter $\mathcal{X}$'skey

$C \leftarrow Enc(pk_{\mathcal{X}}, (Enc(pk_B, xnym_{i,A}))$



k, $sk_{\mathcal{X}}$, for each server: x_A, x_B, x_C, …

Server B k_B ,sk_B

High-level Idea | Pseudonym Conversion

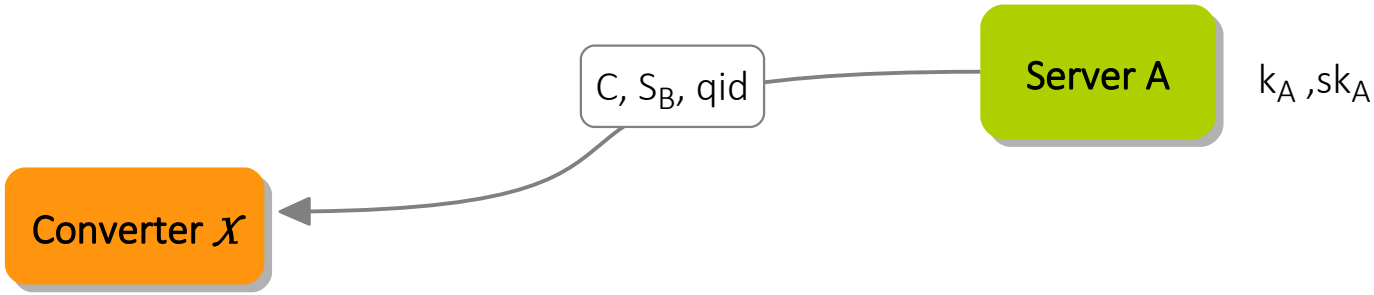
- server S_A wishes to convert a pseudonym $nym_{i,A}$ for server S_B

  S_A's input: $nym_{i,A}$, S_B, qid

1) reobtain $xnym_{i,A} \leftarrow PRP^{-1}(k_A, nym_{i,A})$

2) encrypt $xnym_{i,A}$ under S_B's and Converter $\mathcal{X}$'skey

$C \leftarrow Enc(pk_{\mathcal{X}}, (Enc(pk_B, xnym_{i,A}))$



3) decrypt first layer as

$C' \leftarrow Dec(sk_{\mathcal{X}}, C)$

4) blindly transform encrypted pseudonym

$C'' \leftarrow C'^{\Delta}$ with $\Delta = x_B / x_A$

$C'' = Enc(pk_B, \mathbf{xnym_{i,A}})^{x_B / x_A}$

$= Enc(pk_B, \mathbf{PRF(k,uid_i)^{x_A}})^{x_B / x_A}$

$= Enc(pk_B, \mathbf{PRF(k,uid_i)^{x_B}})$

$= Enc(pk_B, \mathbf{xnym_{i,B}})$

High-level Idea | Pseudonym Conversion

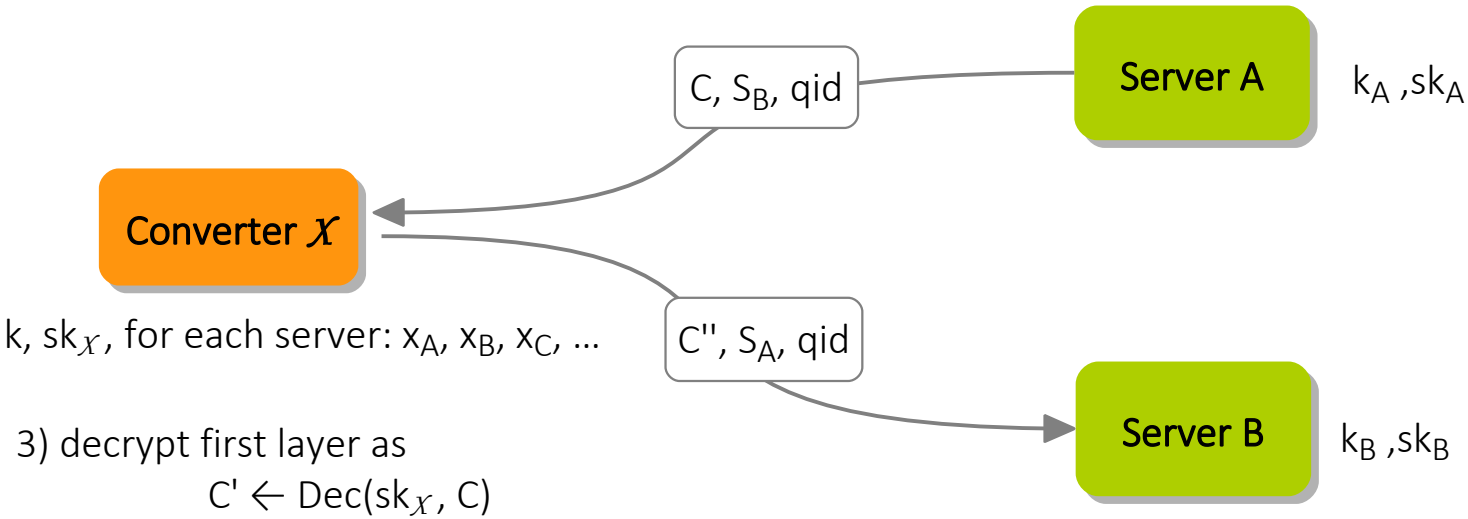
- server S_A wishes to convert a pseudonym $nym_{i,A}$ for server S_B

  S_A's input: $nym_{i,A}$, S_B, qid

1) reobtain $xnym_{i,A} \leftarrow PRP^{-1}(k_A, nym_{i,A})$

2) encrypt $xnym_{i,A}$ under S_B's and Converter $\mathcal{X}$'skey

$C \leftarrow Enc(pk_{\mathcal{X}}, (Enc(pk_B, xnym_{i,A}))$



3) decrypt first layer as

$C' \leftarrow Dec(sk_{\mathcal{X}}, C)$

4) blindly transform encrypted pseudonym

$C'' \leftarrow C'^{\Delta}$ with $\Delta = x_B / x_A$

$C'' = Enc(pk_B, xnym_{i,A})^{x_B / x_A}$

$= Enc(pk_B, PRF(k,uid_i)^{x_A})^{x_B / x_A}$

$= Enc(pk_B, PRF(k,uid_i)^{x_B})$

$= Enc(pk_B, xnym_{i,B})$

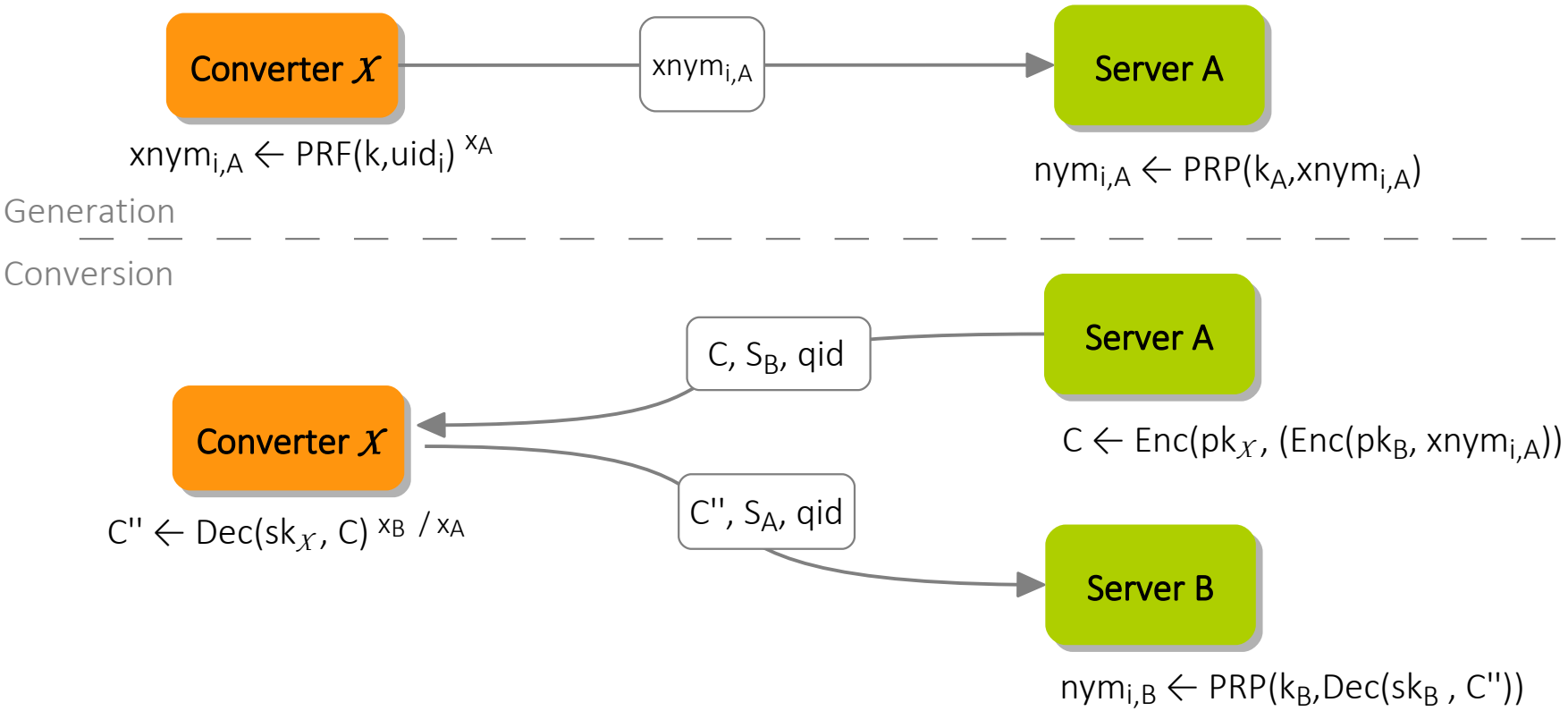
5) decrypt inner pseudonym

$xnym_{i,B} \leftarrow Dec(sk_B, C'')$

6) compute final pseudonym as

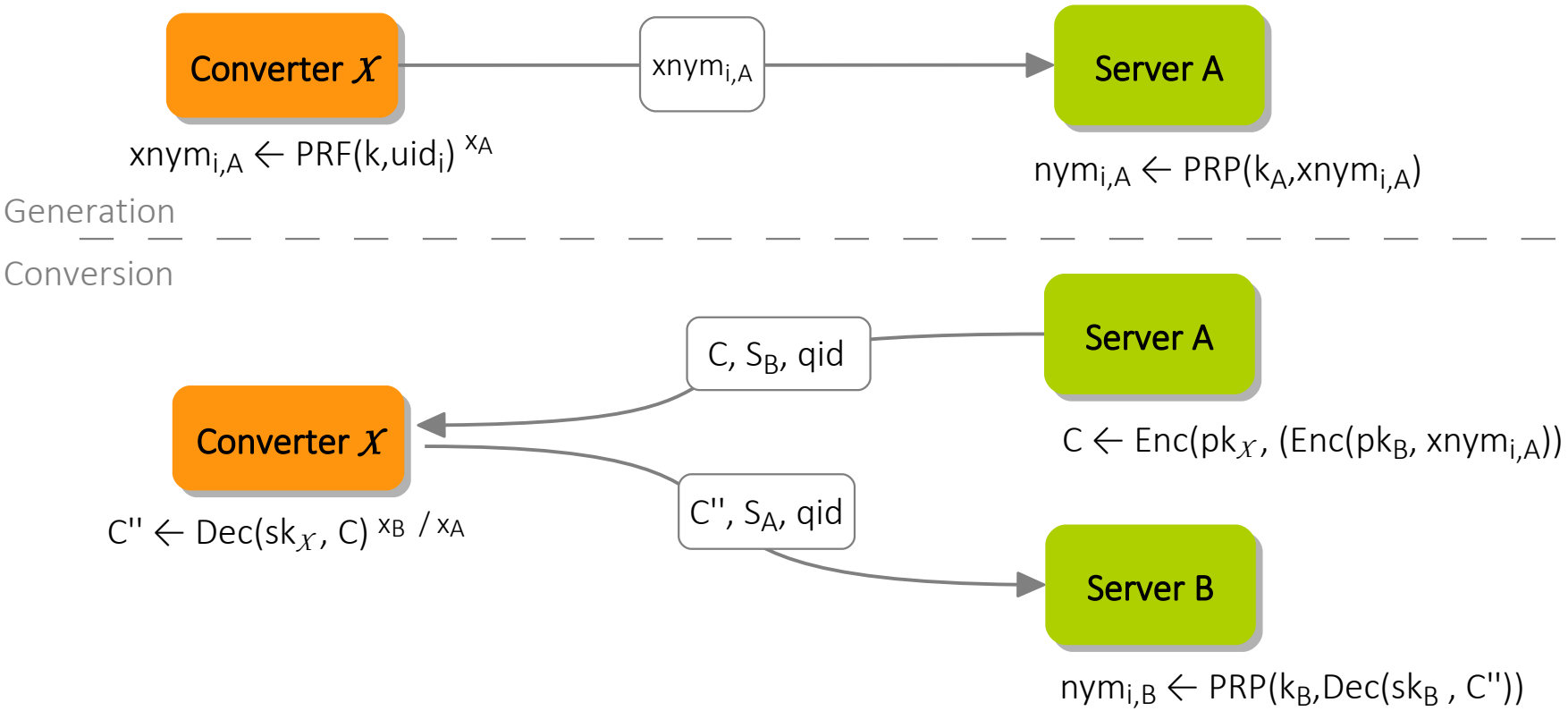
$nym_{i,B} \leftarrow PRP(k_B, xnym_{i,B})$

High-level Idea | Active Corruptions



High-level Idea | Active Corruptions (Server)

- ensure that servers can convert only *their* pseudonyms

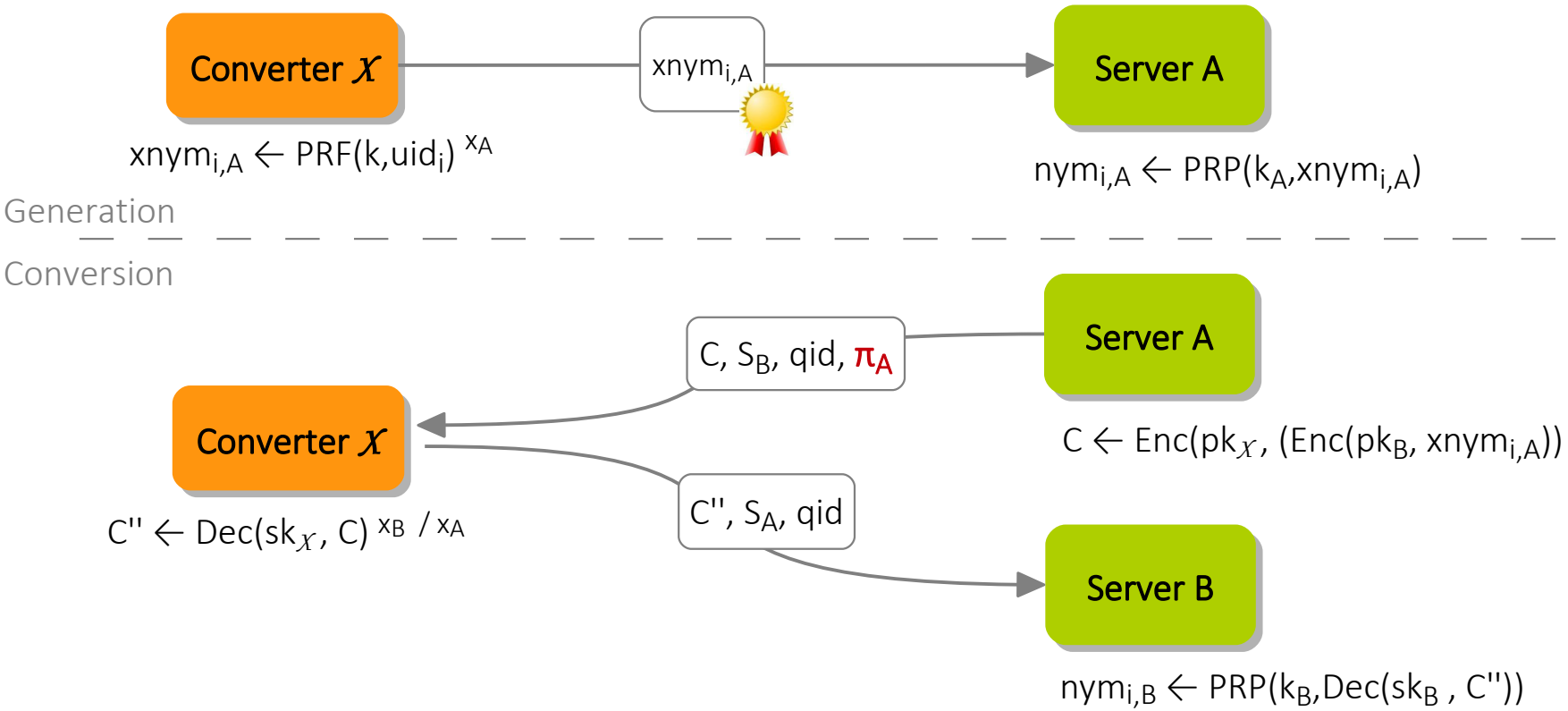


High-level Idea | Active Corruptions (Server)

- ensure that servers can convert only *their* pseudonyms

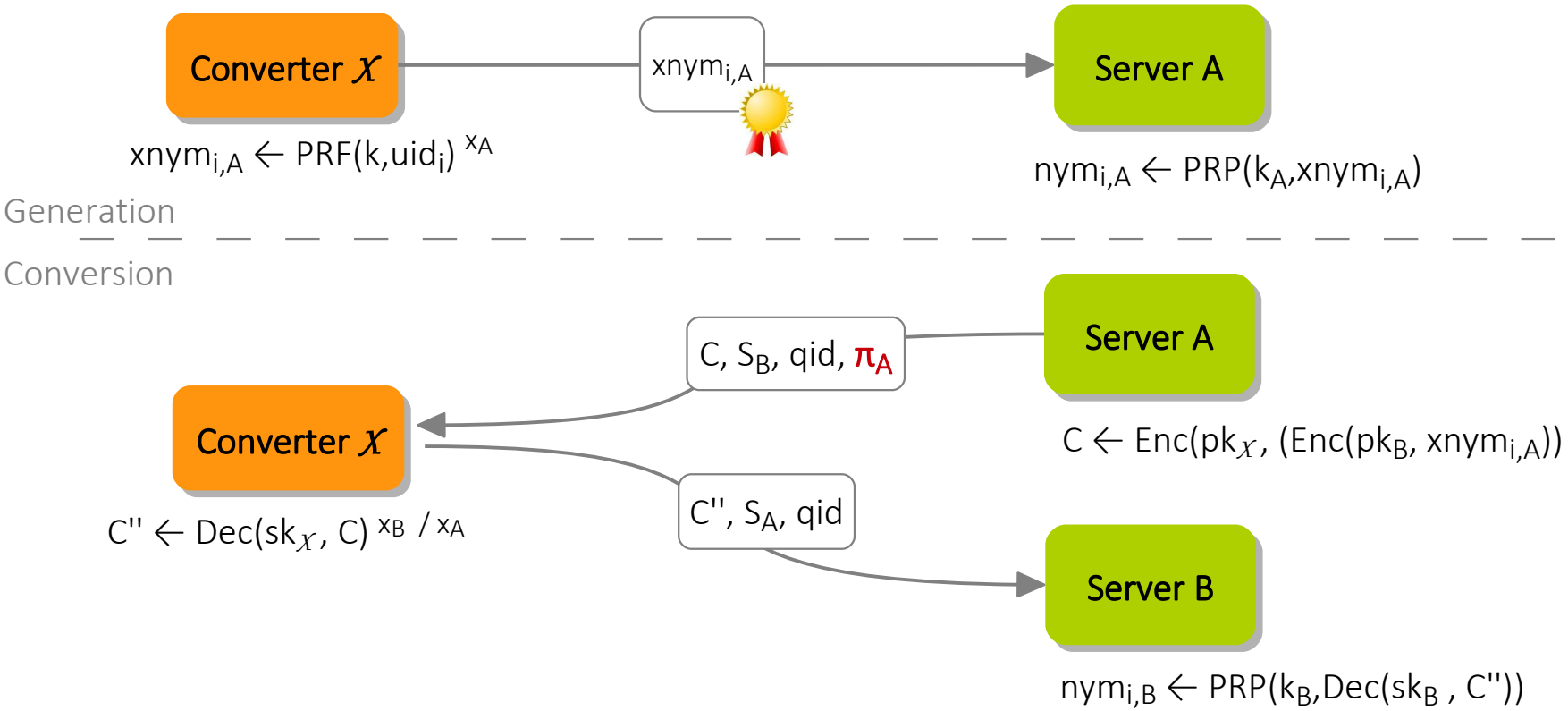
  → generation: "bind" pseudonym $nym_{i,A}$ to server S_A via server-specific signature

  conversion: S_A proves that C contains a correctly signed pseudonym



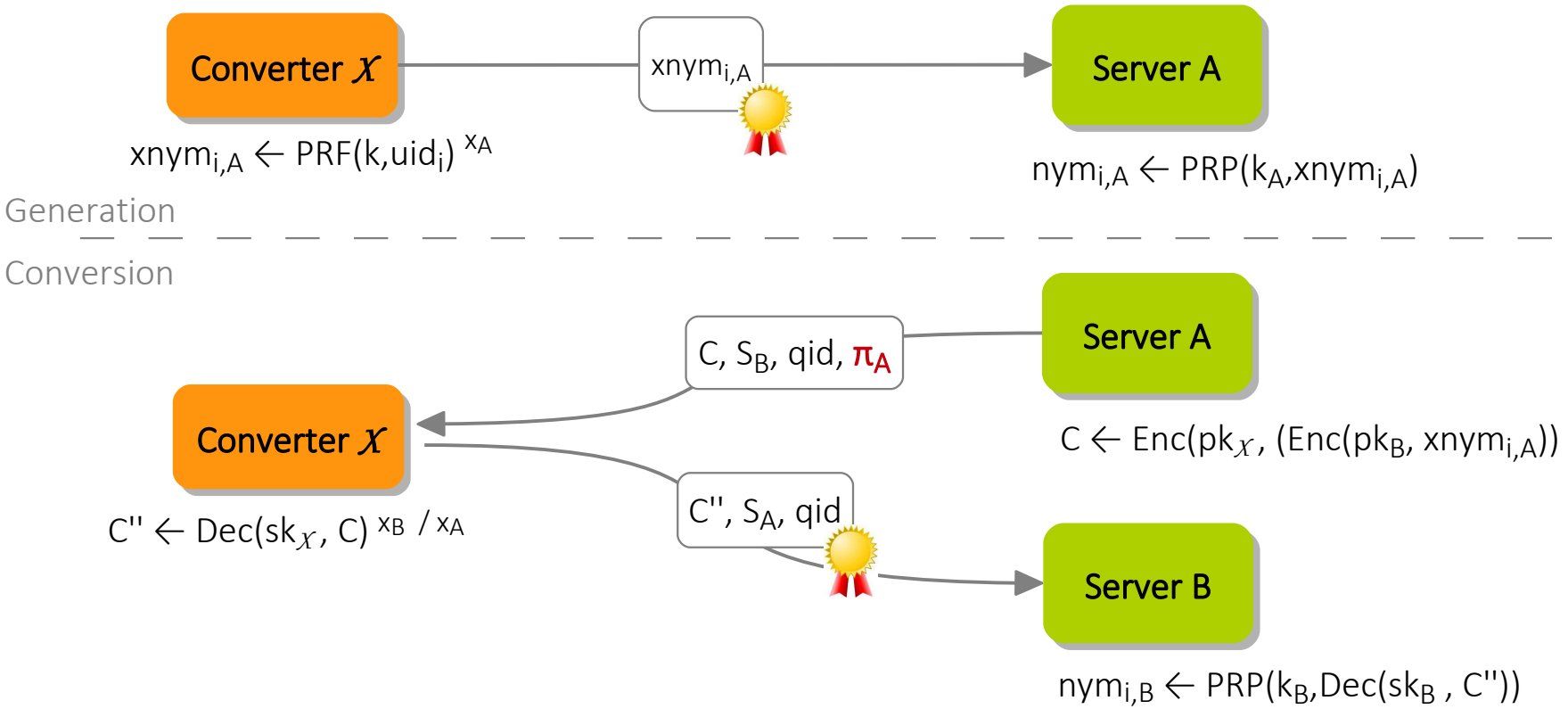
High-level Idea | Active Corruptions (Server)

- ensure that servers can convert only *their* pseudonyms
  - → generation: "bind" pseudonym $nym_{i,A}$ to server S_A via server-specific signature
    conversion: S_A proves that C contains a correctly signed pseudonym
- challenge: how to sign pseudonyms in a blind conversion?



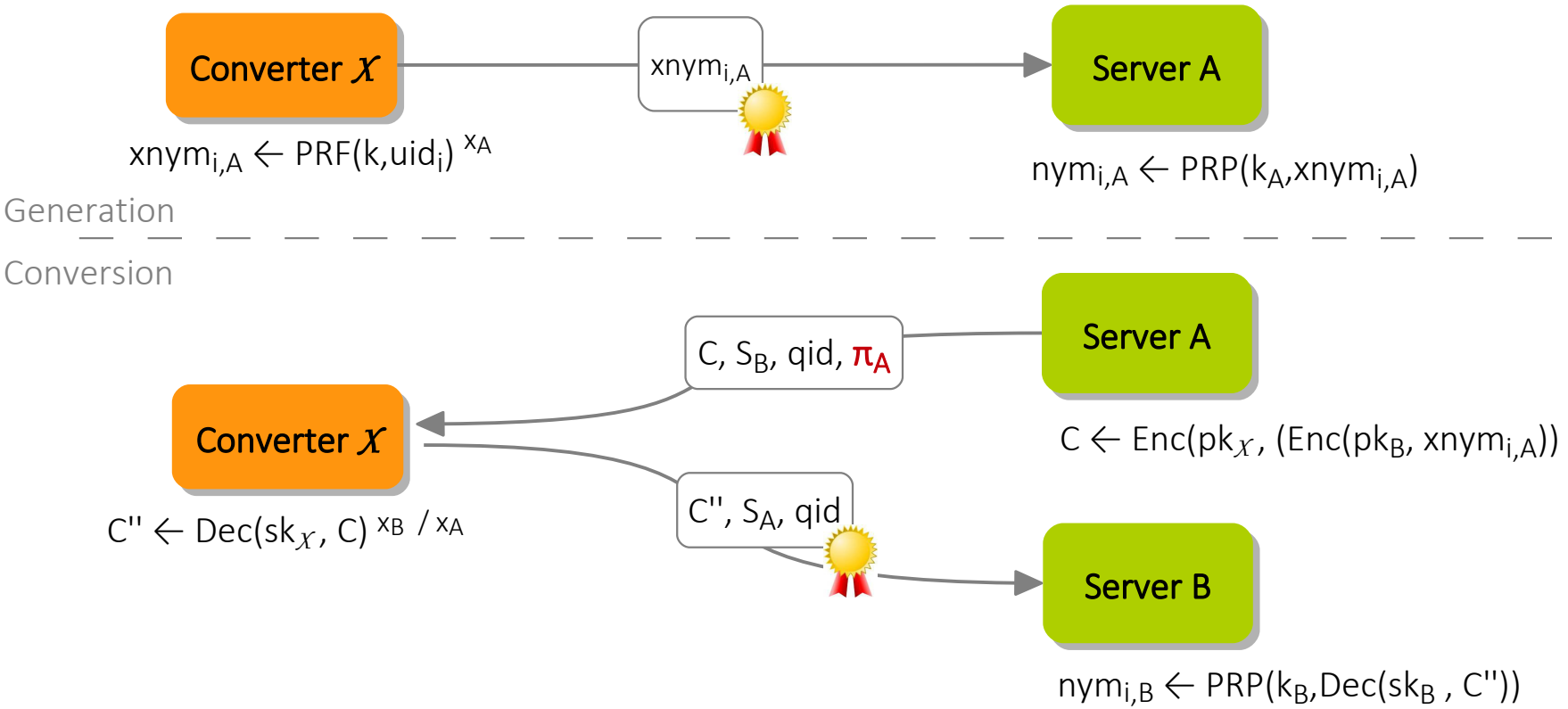
High-level Idea | Active Corruptions (Server)

- ensure that servers can convert only *their* pseudonyms
  - → generation: "bind" pseudonym $nym_{i,A}$ to server S_A via server-specific signature
    conversion: S_A proves that C contains a correctly signed pseudonym
- challenge: how to sign pseudonyms in a blind conversion?
  - → "dual-mode" signatures: signature on ciphertext, can be "decrypted" to signature on plaintext



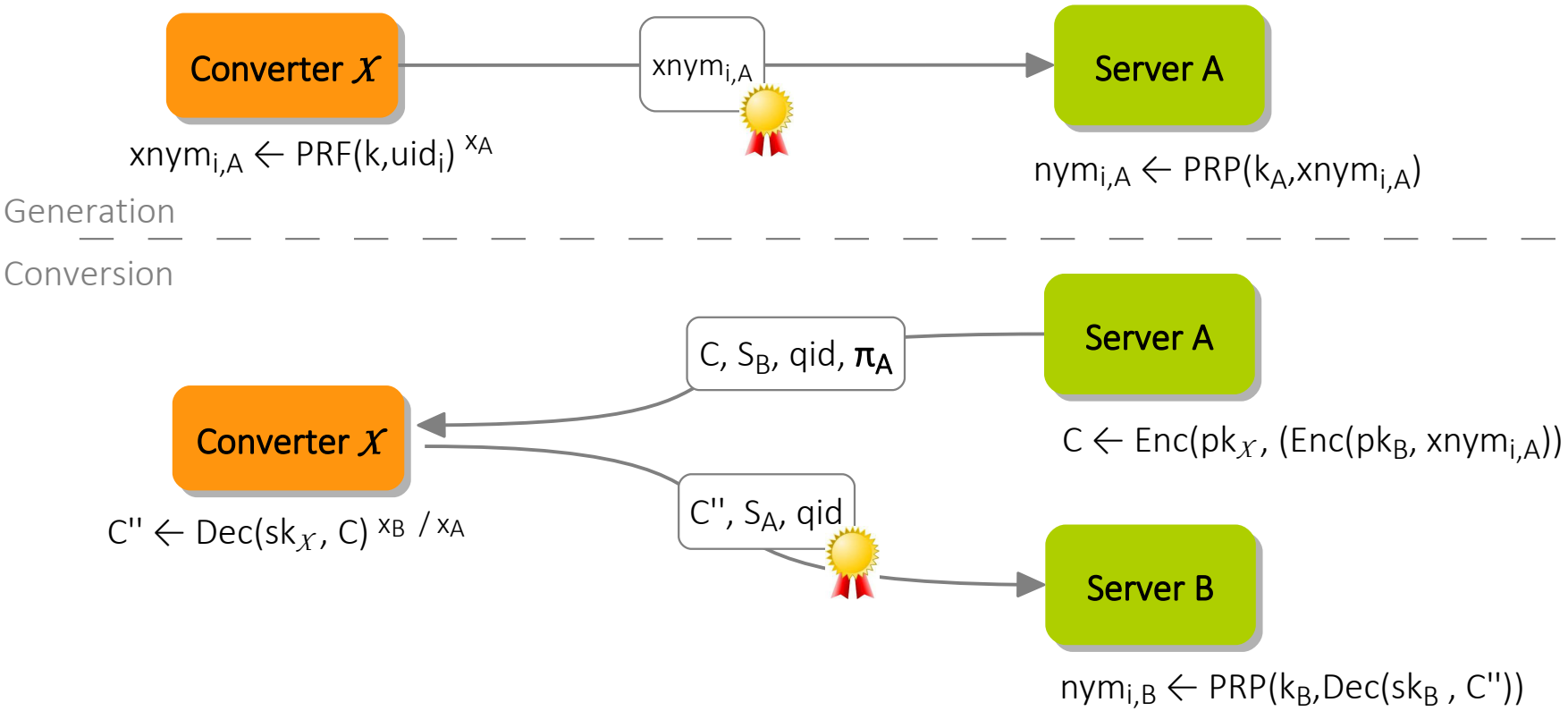
High-level Idea | Active Corruptions (Server)

- ensure that servers can convert only *their* pseudonyms
  - → generation: "bind" pseudonym $nym_{i,A}$ to server S_A via server-specific signature
    conversion: S_A proves that C contains a correctly signed pseudonym
- challenge: how to sign pseudonyms in a blind conversion?
  - → "dual-mode" signatures: signature on ciphertext, can be "decrypted" to signature on plaintext



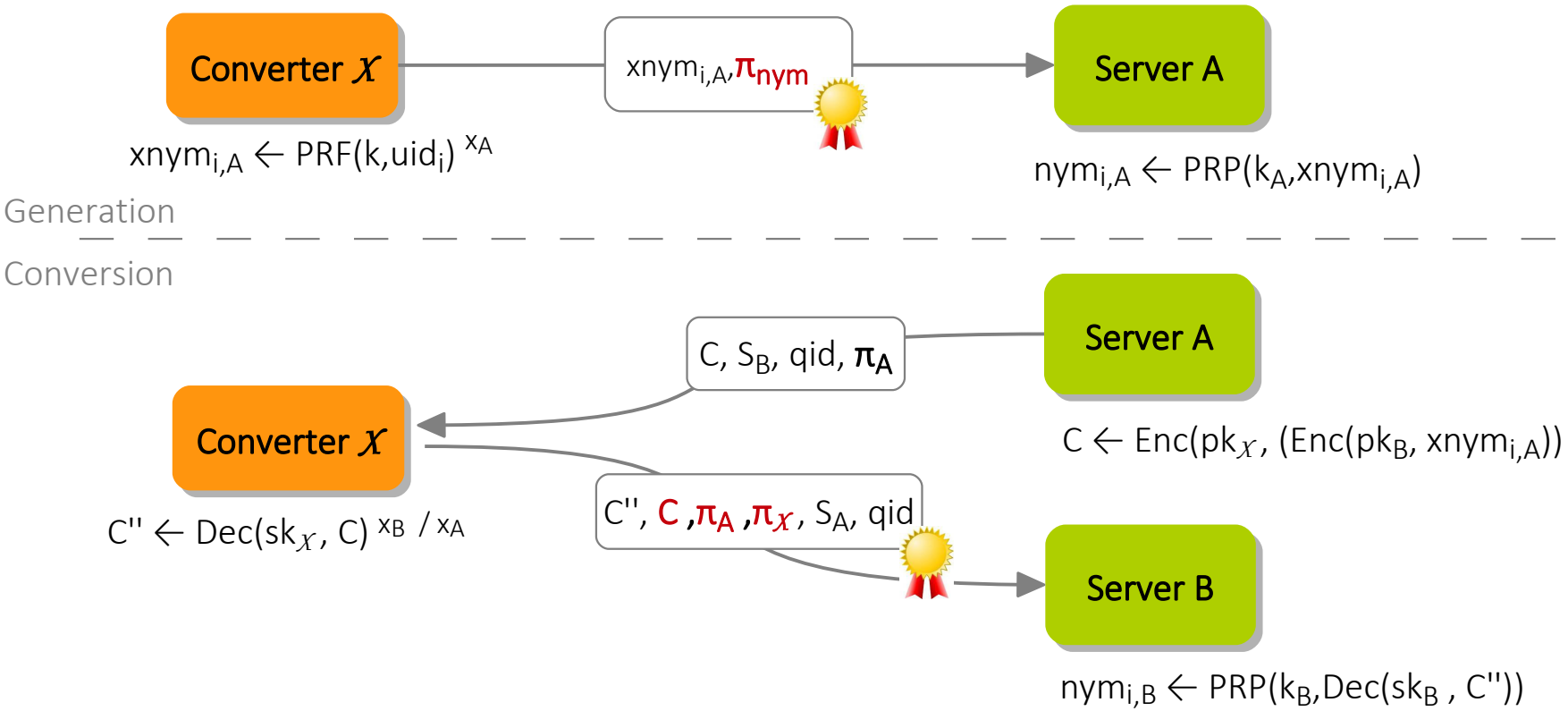
High-level Idea | Active Corruptions (Converter)

- ensure consistency of pseudonyms and conversions even in the presence of a corrupt converter
  → let converter $\mathcal{X}$ prove correctness of his computations via NIZKs



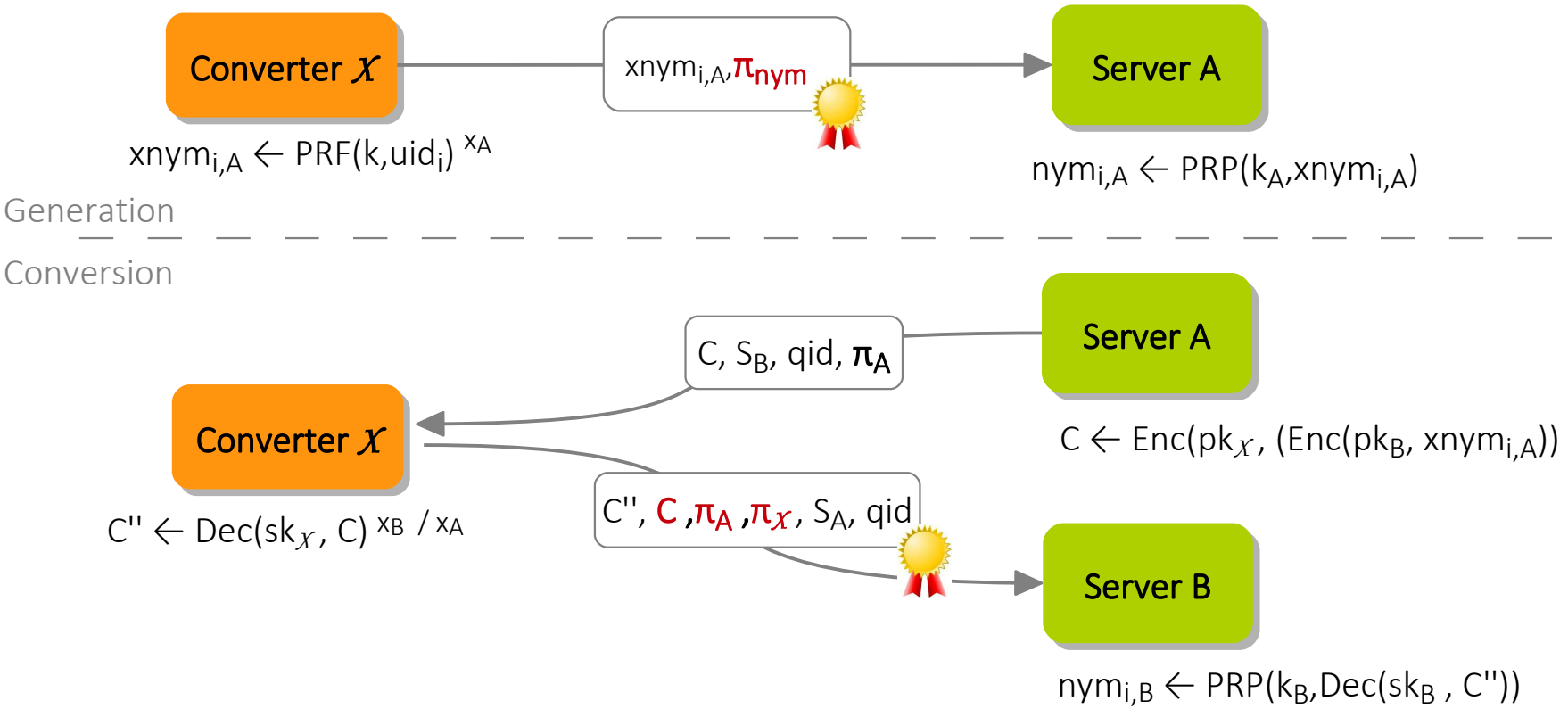
High-level Idea | Active Corruptions (Converter)

- ensure consistency of pseudonyms and conversions even in the presence of a corrupt converter
  → let converter $\mathcal{X}$ prove correctness of his computations via NIZKs



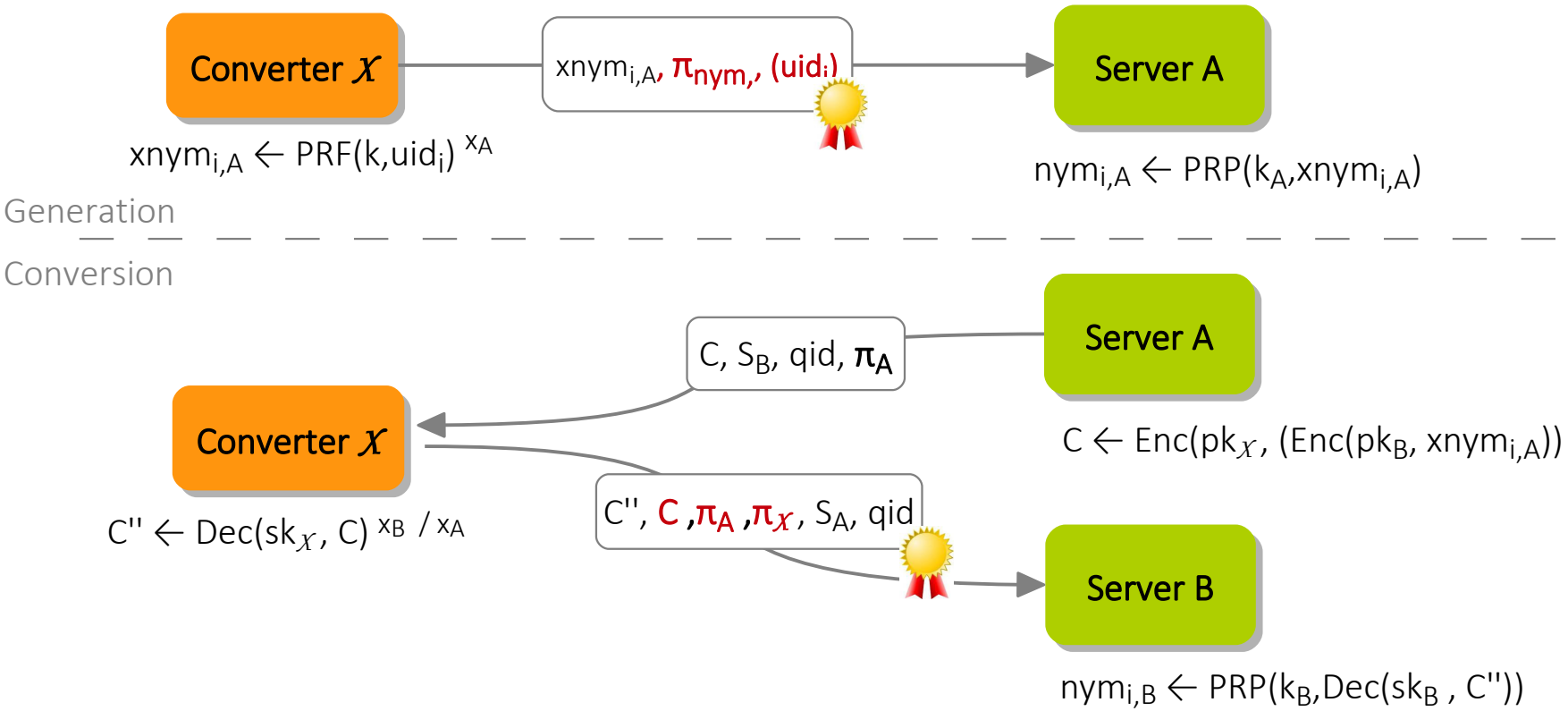
High-level Idea | Active Corruptions (Converter)

- ensure consistency of pseudonyms and conversions even in the presence of a corrupt converter
  → let converter $\mathcal{X}$ prove correctness of his computations via NIZKs



High-level Idea | Active Corruptions (Converter)

- ensure consistency of pseudonyms and conversions even in the presence of a corrupt converter
  → let converter $\mathcal{X}$ prove correctness of his computations via NIZKs
- pseudonym generation can be anonymous or not
  → non-anon: Server S_A can verify that $xnym_{i,A}$ was correctly derived from uid_i
  option important for bootstrapping / migration



(Un)linkable Pseudonyms | Efficiency & Summary

efficiency

- security against corrupt converter and corrupt servers:
  - generation ($\mathcal{X}$+S_A): 15 exponentiations + 8 pairings
  - conversion ($\mathcal{X}$+S_A+S_B): 84 exponentiations + 30 pairings

(most can be merged into multi-exponentiations)

- more efficient variant if converter is *honest-but-curious* (but servers fully corrupt)
  - generation ($\mathcal{X}$+S_A): 7 exponentiations
  - conversion ($\mathcal{X}$+S_A+S_B): 40 exponentiations + 16 pairings

(un)linkable pseudonyms without *trusted* converter

- unlinkable data storage with controlled data exchange
  - servers maintain data w.r.t. local, random-looking pseudonyms
  - pseudonyms can only be linked via a central converter
- conversions done in a blind way → converter must not be a trusted entity
- efficient and provably secure protocol

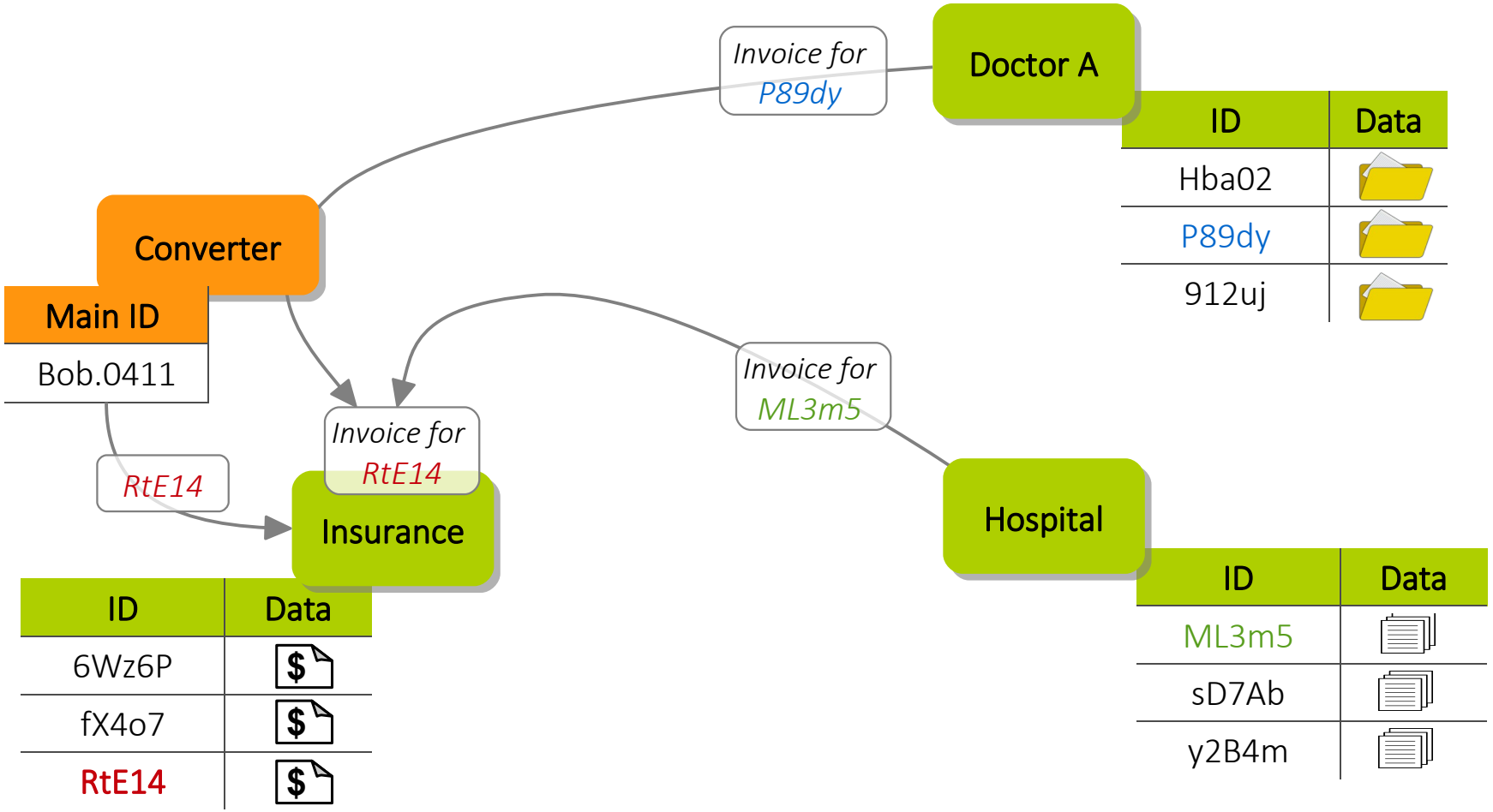
→ paradigm shift: unlinkable as default, linkable only when necessary

(Un)linkable Pseudonyms for Governmental Databases

Jan Camenisch and Anja Lehmann
IBM Research Zurich

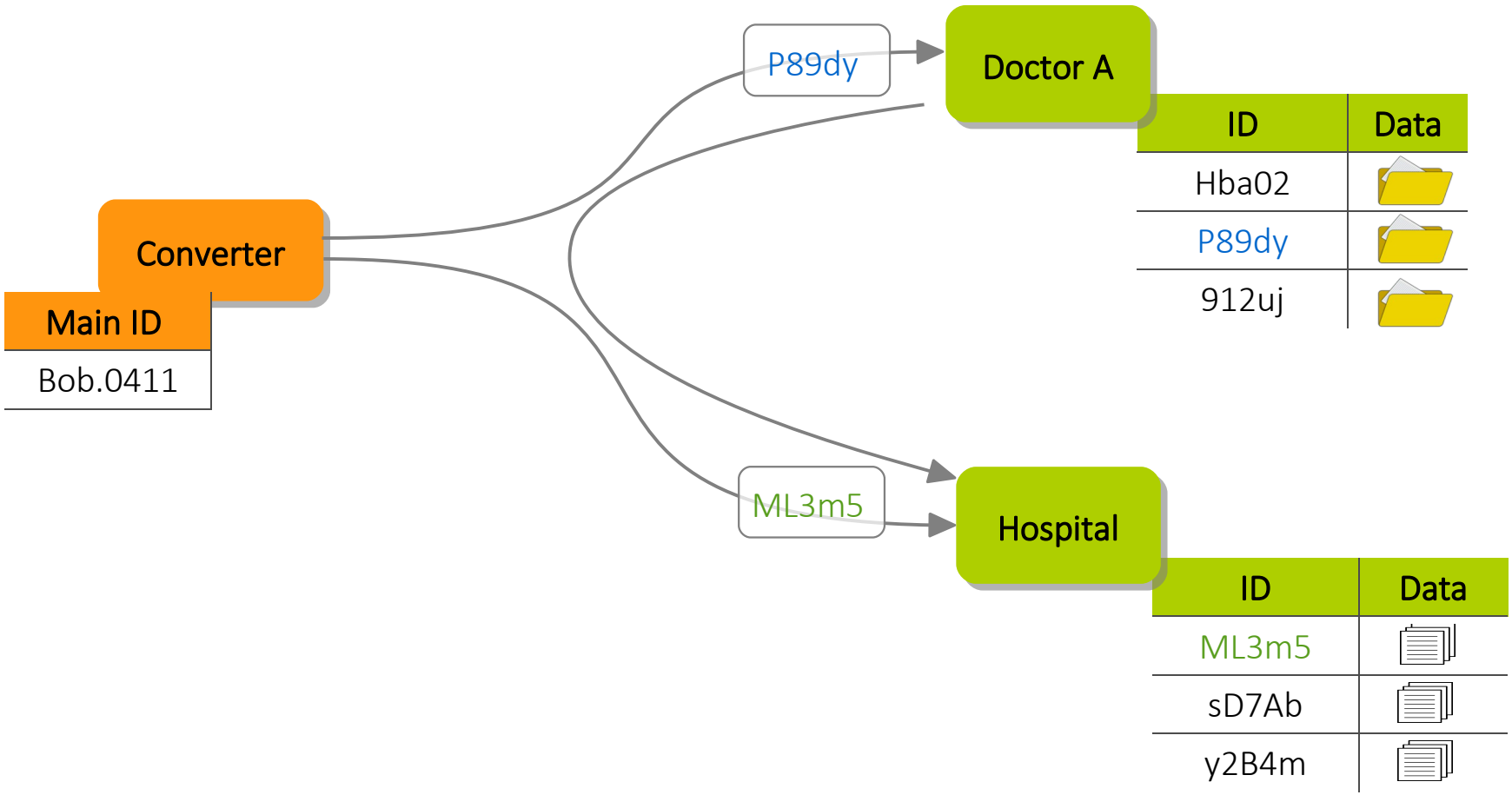
(Un)linkable Pseudonyms | Consistency

- pseudonyms are unique & consistent
  - generation is deterministic, injective and consistent with blind conversion
  - conversions are consistent and transitive



(Un)linkable Pseudonyms | Consistency

- pseudonyms are unique & consistent
  - generation is deterministic, injective and consistent with blind conversion



(Un)linkable Pseudonyms | Consistency

- pseudonyms are unique & consistent
  - generation is deterministic, injective and consistent with blind conversion
  - conversions are consistent and transitive