Blockchain-based Decentralized Public Key Management for Named Data Networking

Kan Yang, Lan Wang

Dept. of Computer Science, University of Memphis, USA

Jobin J. Sunny

St. Jude Children's Research Hospital

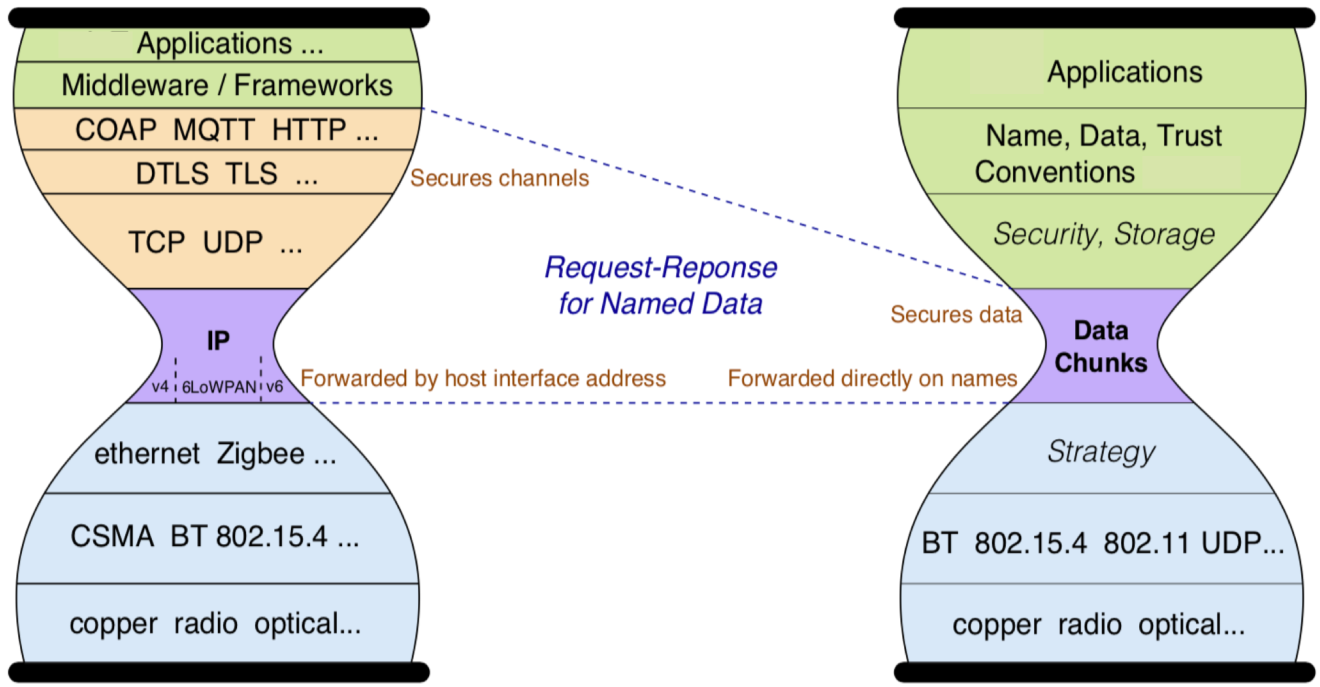
Paper in Proceedings of *IEEE ICCCN 2018,* Hangzhou, China

Outline

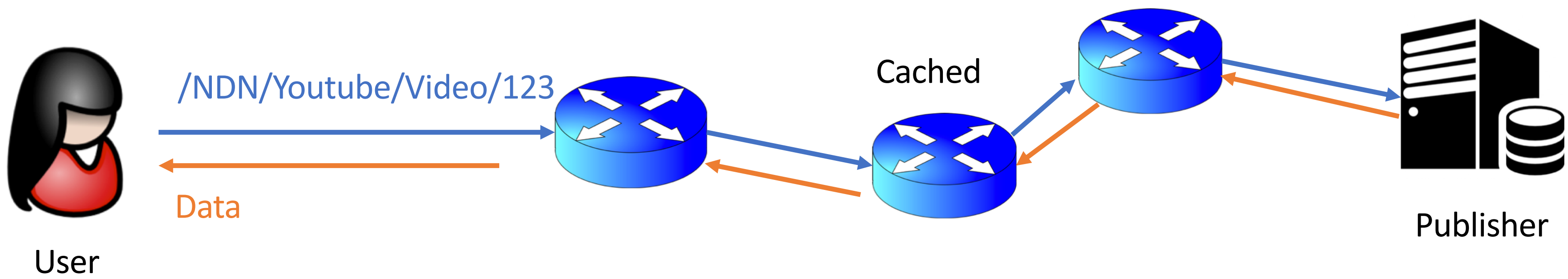
- Data-Centric Security in NDN
- Public Key Management and Compromised CA Problem
- BC-PKM: Blockchain-based Decentralized Public Key Management
  - Framework
  - Concrete Design
  - Prototype
- Conclusion and Future Work

Named Data Networking

- Bind data with a name
- Retrieve data by its name
- Forward data interest directly on names
- Forward data along the interest path



Named Data Networking



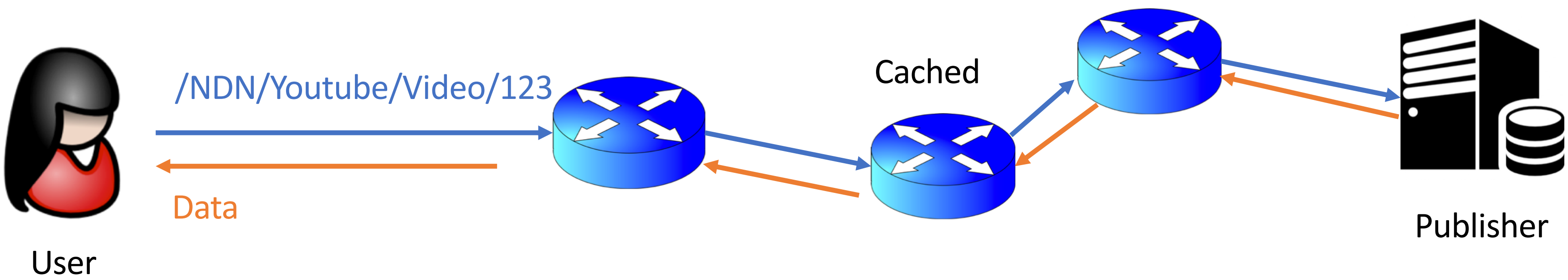
Interest Packet

Content Name
Nonce
Guiders (scope, lifetime)

Data Packet

Name
MetaInfo
Content
Signature

Data Centric Security



- All the content must be signed!
- Routers may verify the signature.
- Users must verify the signature.

Data Packet

Name
MetaInfo
Content
Signature

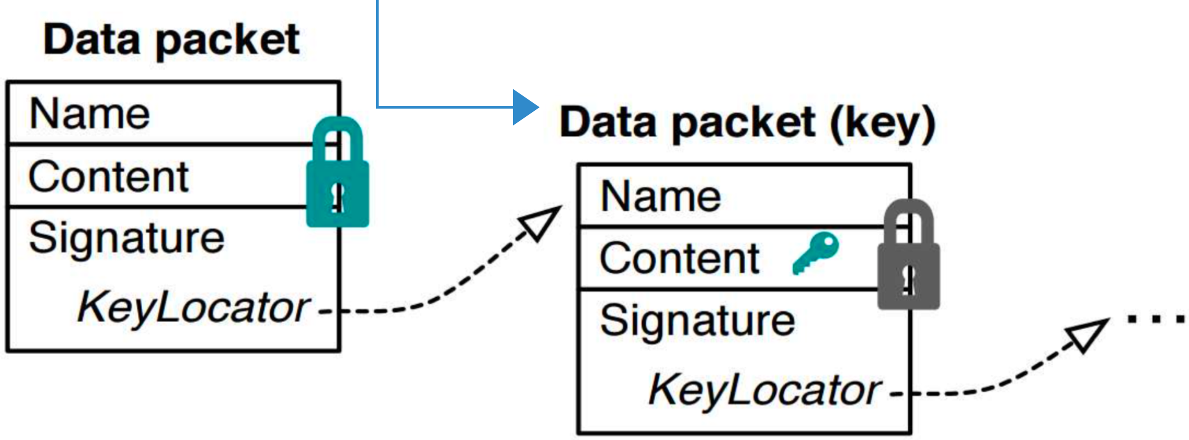
Signature from Public Key Cryptography

- Data signature is usually generated by the secret key of the producer

$$\sigma = Sign(sk, data)$$

- The signature can be validated by the public key of the producer

$$Verify(pk, data, \sigma) \rightarrow Success/Fail$$



Public Key Management (PKM)

- Data signature is usually generated by the secret key of the producer

$$\sigma = Sign(sk, data)$$

- The signature can be validated by the public key of the producer

$$Verify(pk, data, \sigma) \rightarrow Success/Fail$$

(sk_{eve}, pk_{eve})

Impersonate Alice by signing the data with his secret key:

$$\sigma_{alice} = Sign(sk_{eve}, data)$$

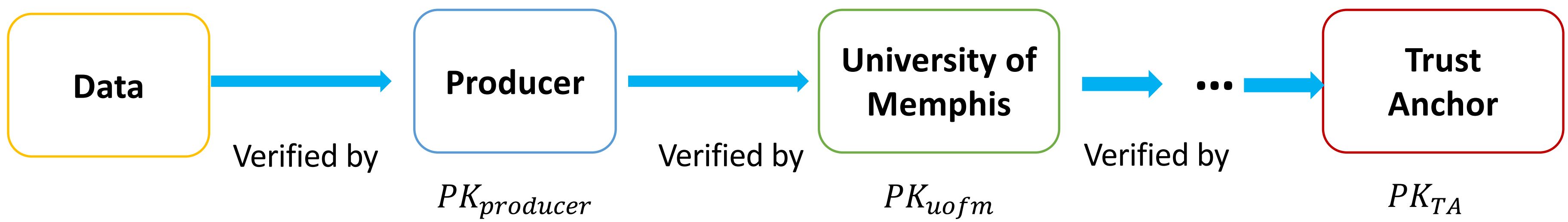
and claiming that Alice's public key is pk_{eve}.

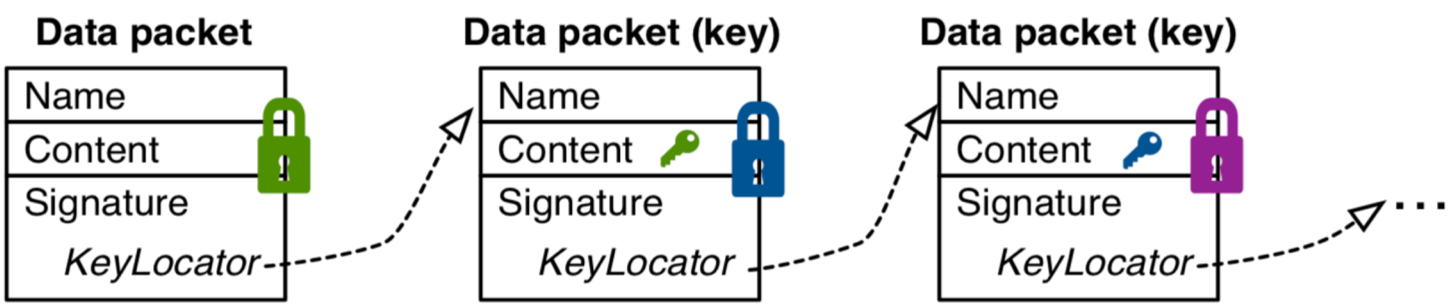
$$Verify(pk_{eve}, data, \sigma_{alice}) \rightarrow Success/Fail$$

Public Key Management is the foundation of the data-centric security!

Trust Schema: Current PKM in NDN

- Trust Relationship
- Recursively validate the signature until it reaches the trust anchor



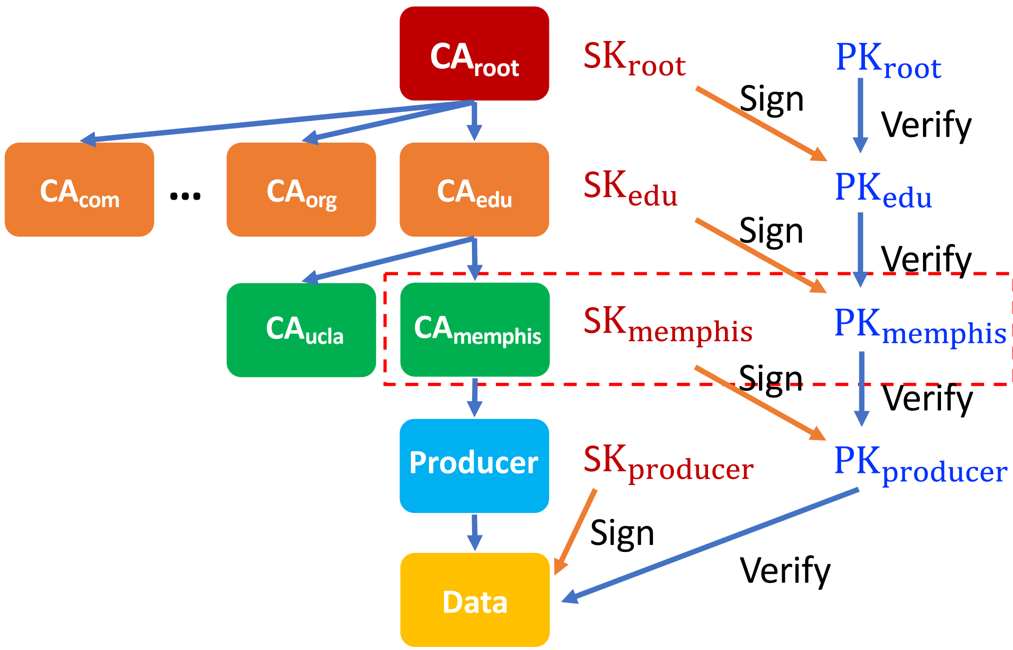


The public key of Trust Anchor is wired into the hardware or preconfigured in the software

Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, Lixia Zhang. "Schematizing and Automating Trust in Named Data Networking". *ICN'2015,* ACM, 2015, pp. 177– 186.

Traditional CA-based PKI

- Certificate Authority (CA) issues a certificate to prove that the public key is associated with a name.



Compromised CA Problem

An attacker can compromise a CA to bind a name to an unauthorized public key and produce false data using the fake certificate.

Compromised CA Incidents

Year	Incidents
2001	• VeriSign issues Microsoft Corporation code signing certificate to a non-Microsoft employee.
2008	• Thawte issues certificate for Live.com to non-Microsoft employee • Comodo issues mozilla.org certificate to Startcom • Organization forges VeriSign RapidSSL certificates
2011	• Comodo issues nine counterfeit certificates (Google, Yahoo, Live, etc.) when registration authority is compromised. • StartSSL CA compromised • DigiNotar compromised. 531 fraudulent certificates issued. • Boeing CA compromised
2012	• Microsoft CA certificates forged by exploiting MD5 (Flame)
2013	• Fraudulent certificates on Google domains issued by the French Ministry of Finance CA (ANSSI)
2014	• Intermediate CA in India compromised
2015	• Dell notebooks with rogue root CA
2016	• One CA attacked another by attempting to trademark the brands used by the second CA

Sources: https://csrc.nist.gov/csrc/media/projects/forum/documents/2012/october-2012_fcsm_pturner.pdf
http://wiki.cacert.org/Risk/History

Attacks from Compromised CA

Once the CA has been compromised, the CA has superpower to

- Register public keys for illegitimate principals
- Update public keys for existing principals
- Revoke public keys for legitimate principals

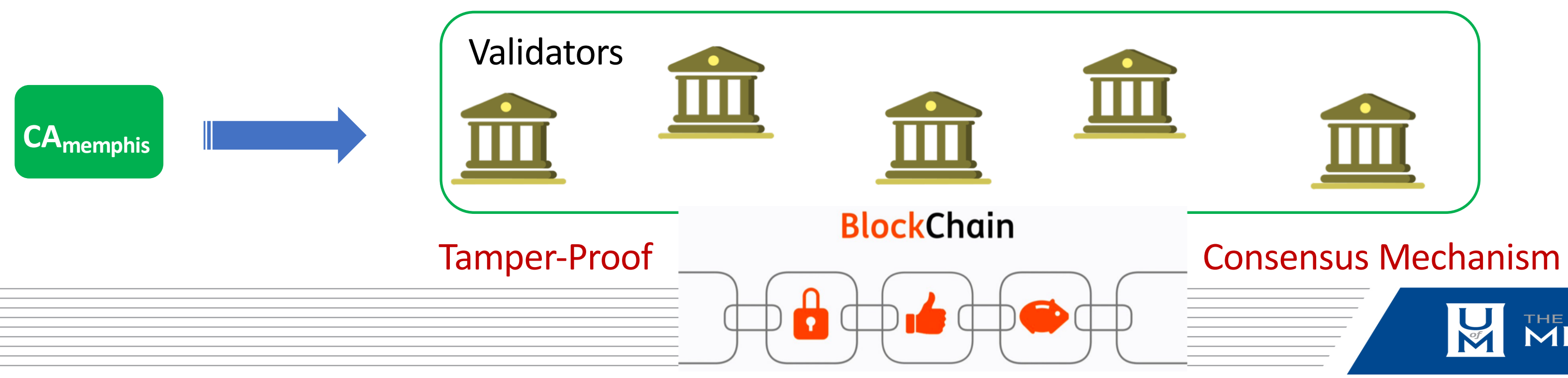
Our Idea: Reduce the superpower of single CA

Once the CA has been compromised, the CA has superpower to

- Register public keys for illegitimate principals
- Update public keys for existing principals
- Revoke public keys for legitimate principals

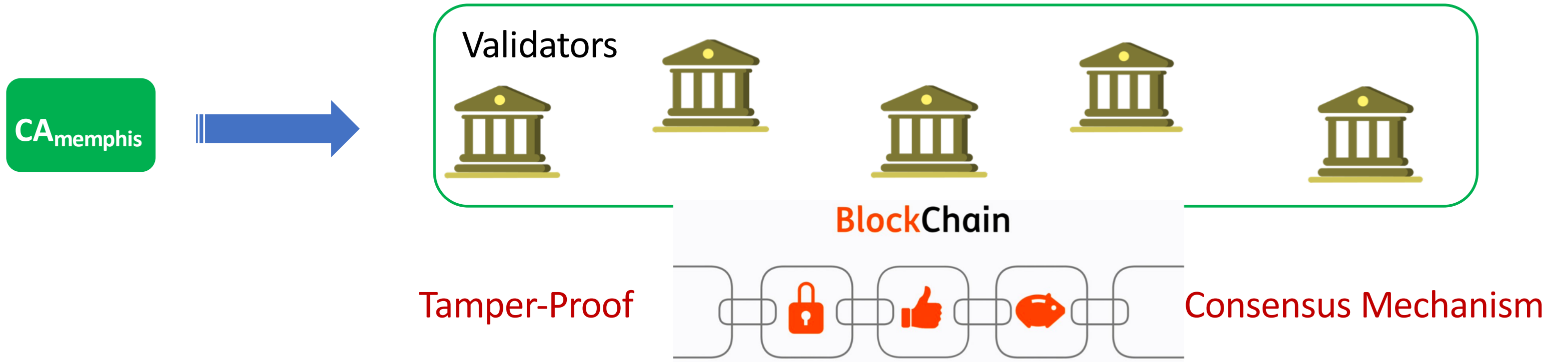
Our idea: Replace single CA with a set of Validators with lower privileges

- do the name-principal validation
- follow the majority principle to implement the public key management functions

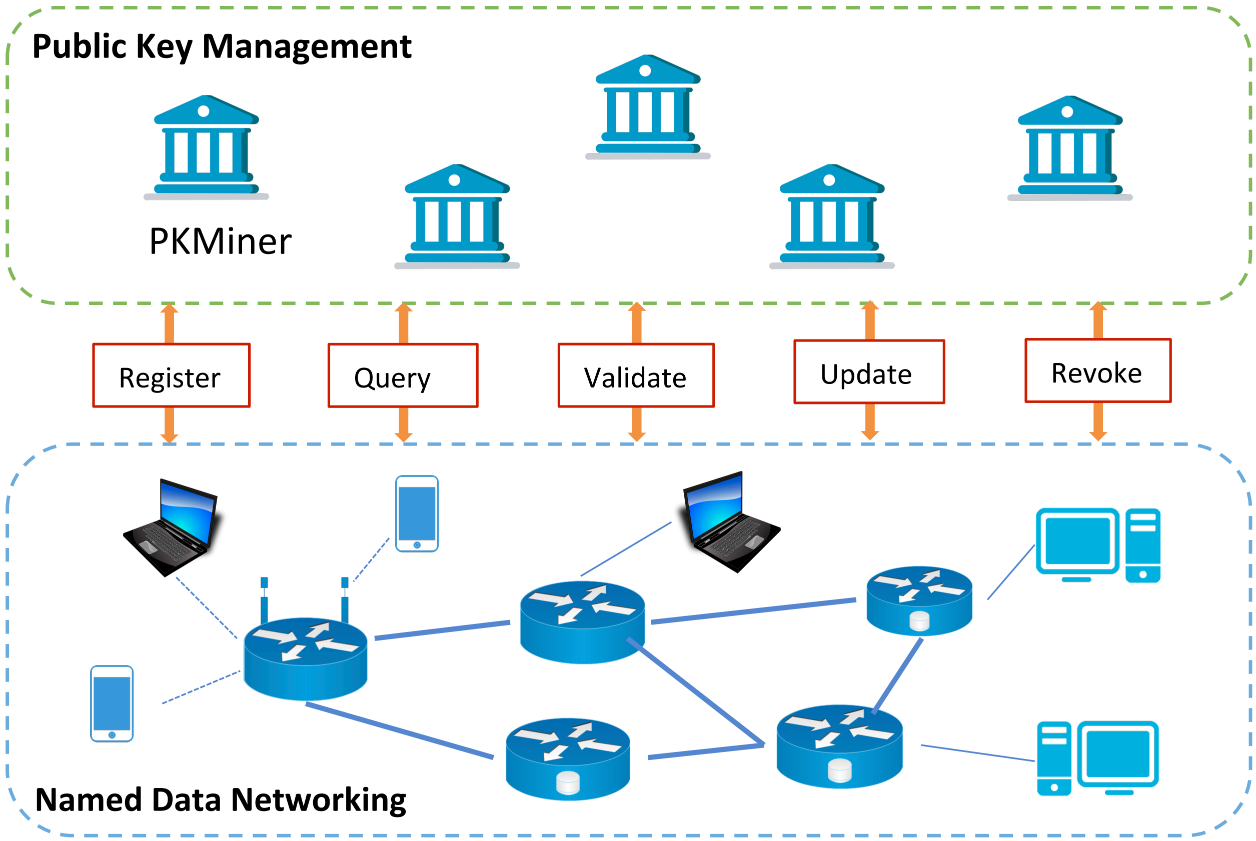


Solution: Decentralize CAs and Publish Their Actions

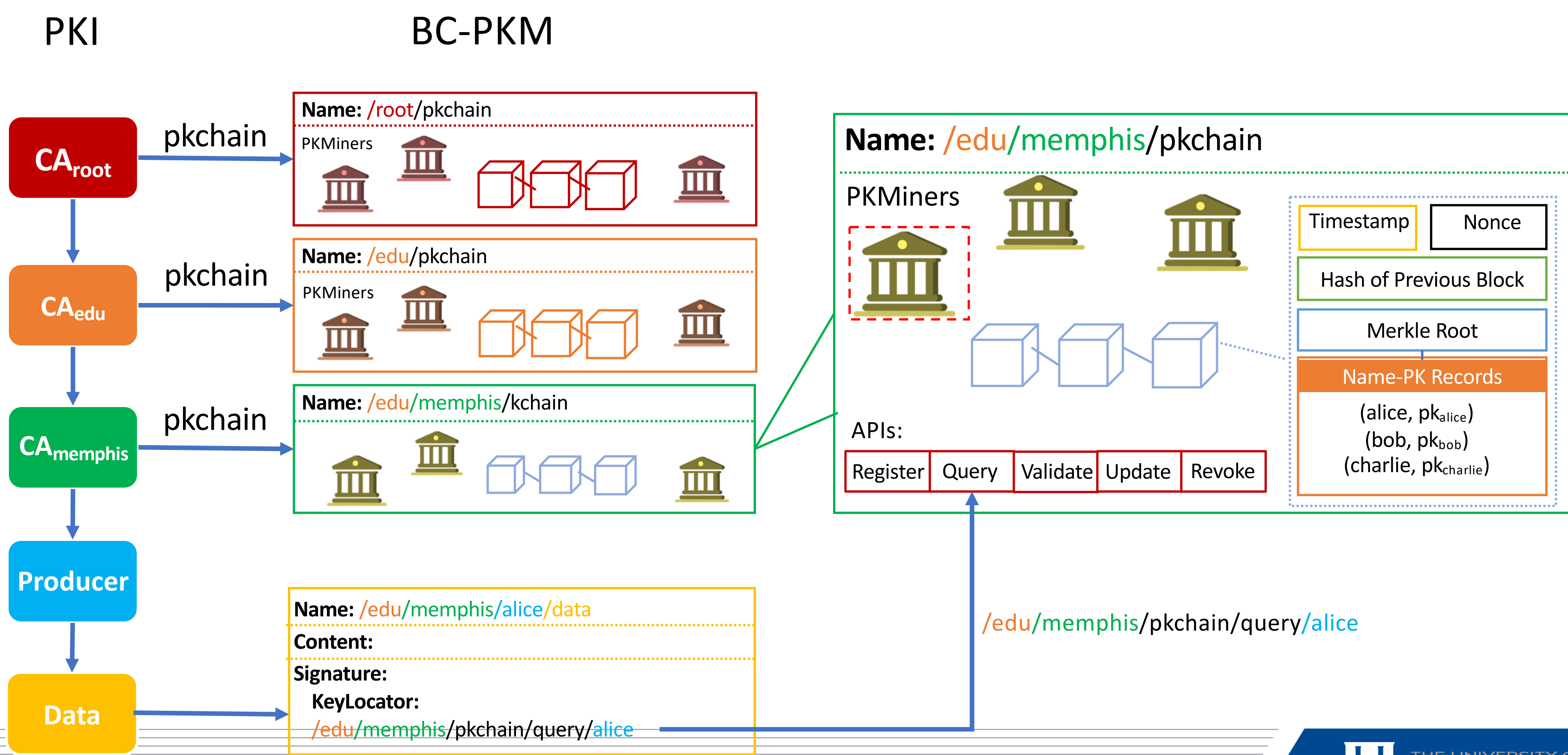
- Replace the CA at each level with a set of Validators with lower privileges
  - A validator can publish a public key record only if a **majority** agrees.
- Publish every public key record in a **tamper-proof blockchain**
- Majority rule → As long as the attacker cannot compromise half or more of the validators, an invalid public key record will not be issued.
- Tamperproofness → Even if a validator misbehaves and publishes an invalid public key record, this can be detected by other validators through the blockchain.



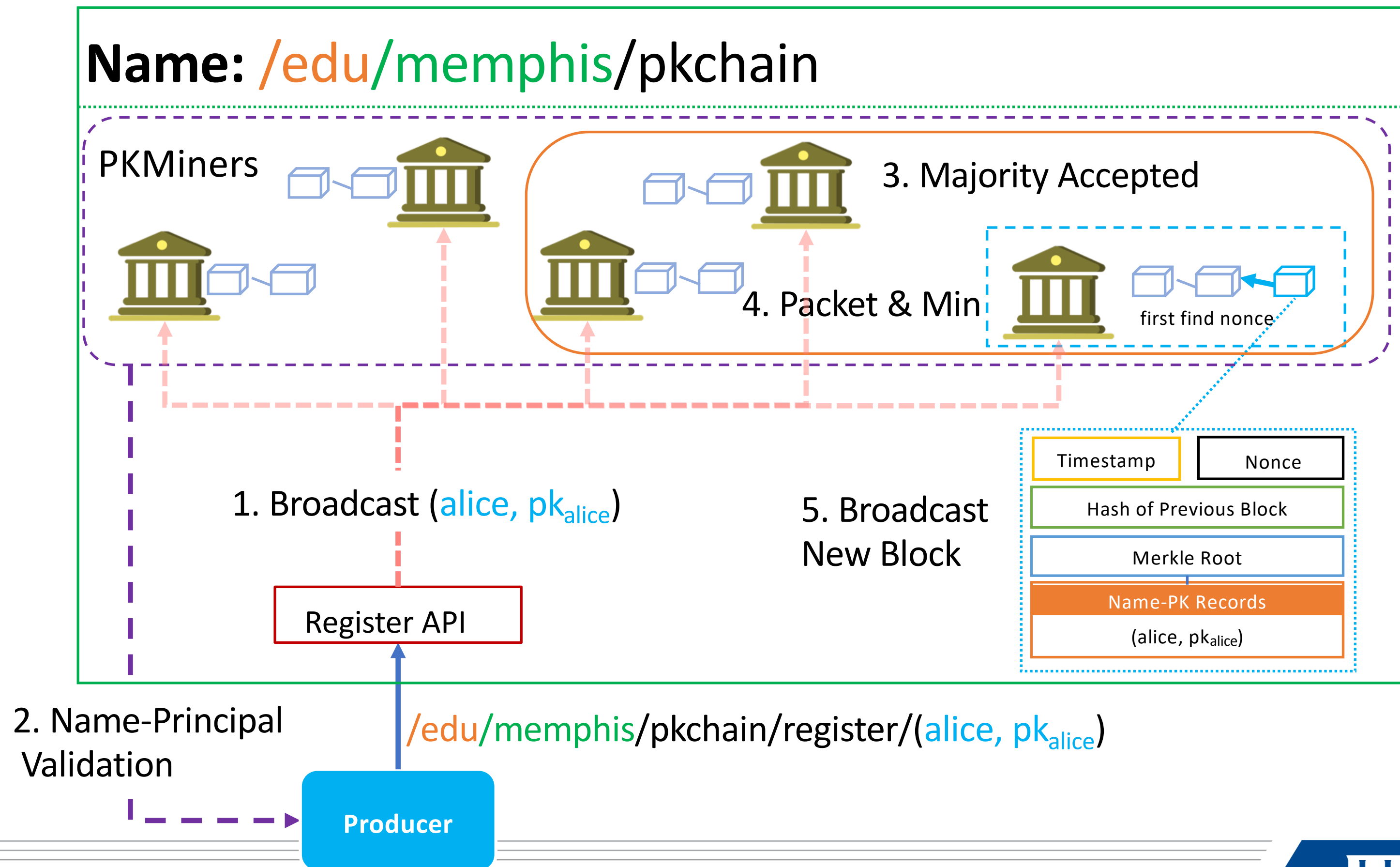
Overview of Public Key Management in NDN



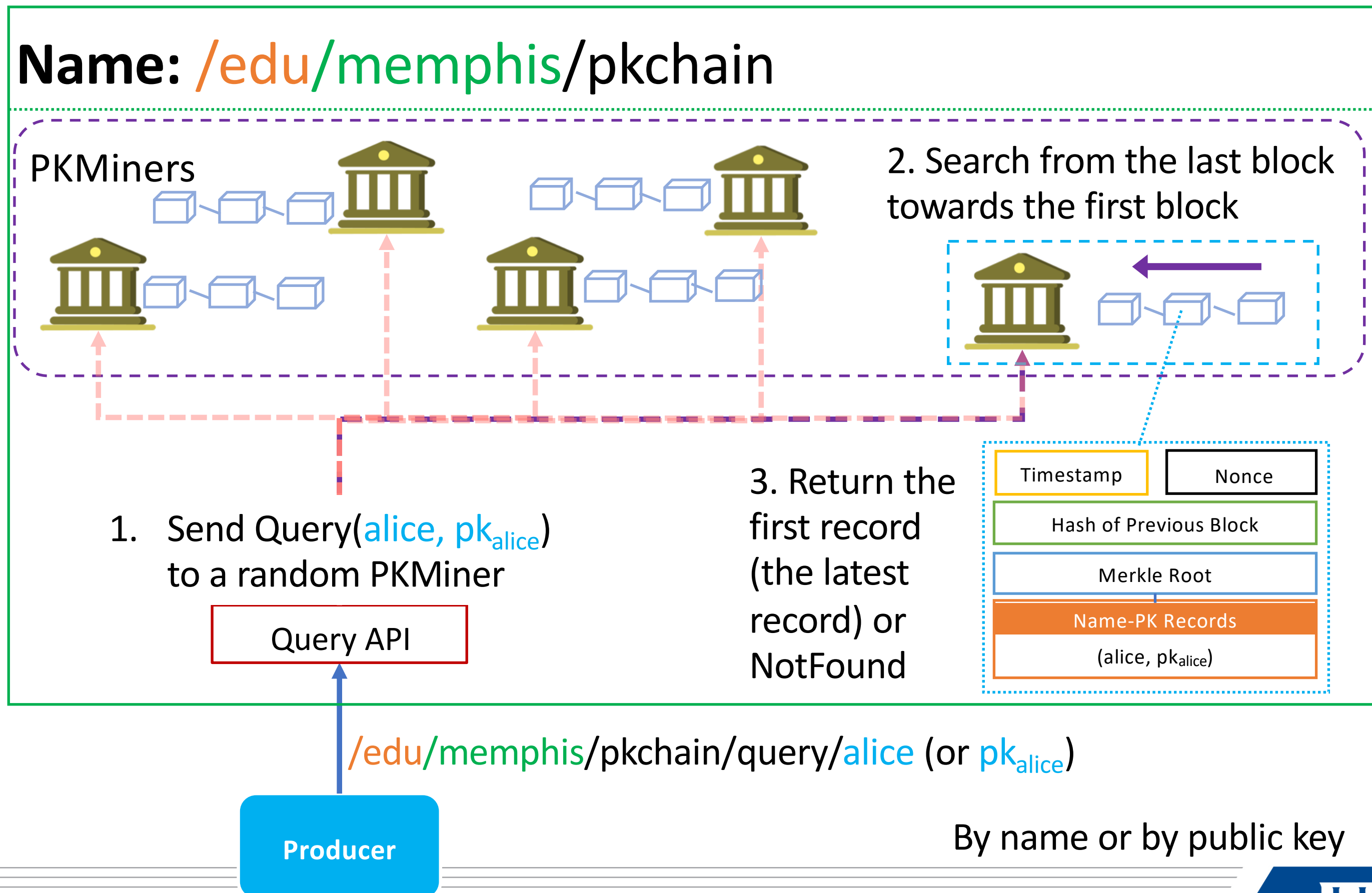
Framework of Blockchain-based PKM (BC-PKM)



PKChain: Register

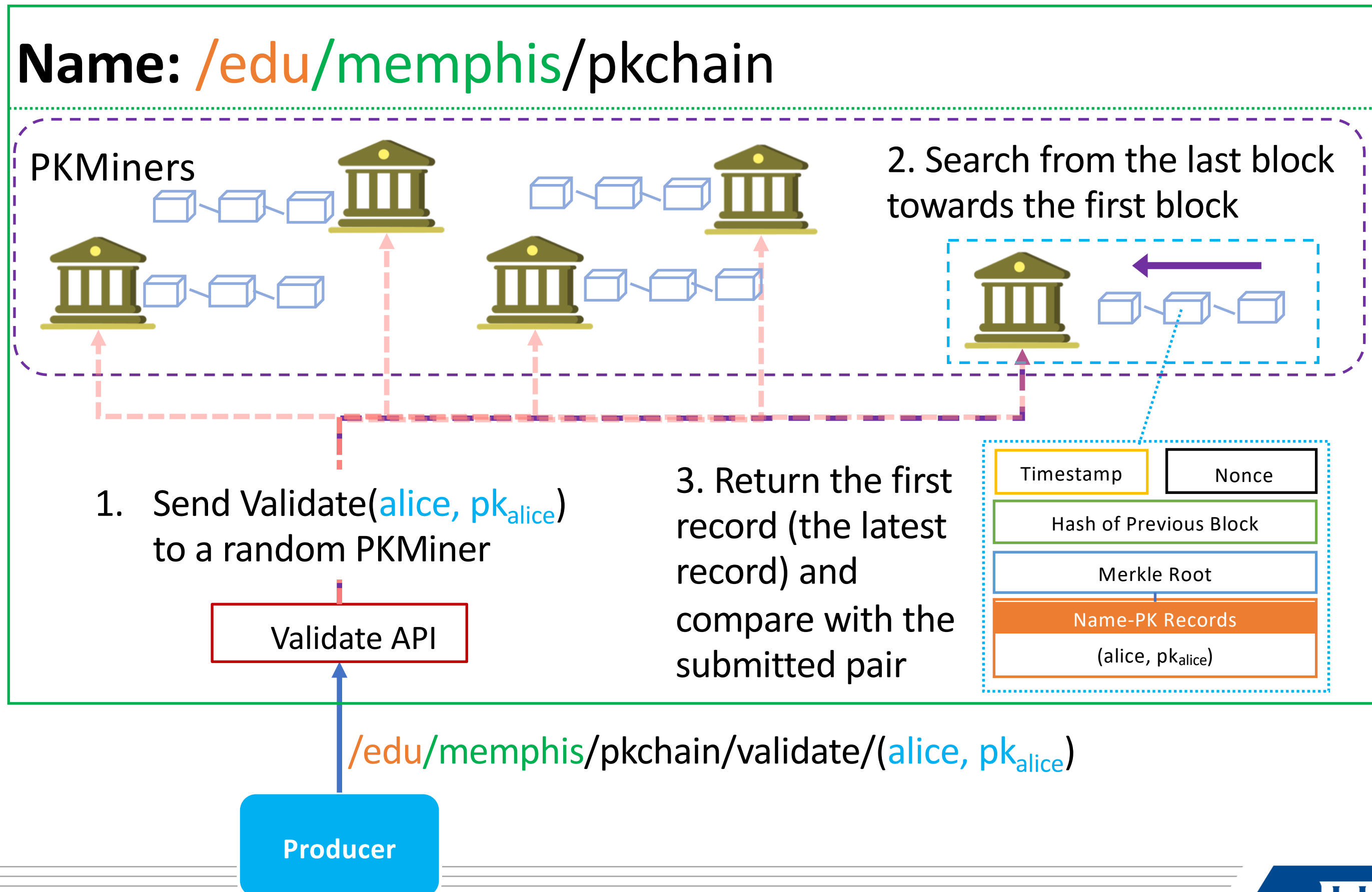


PKChain: Query

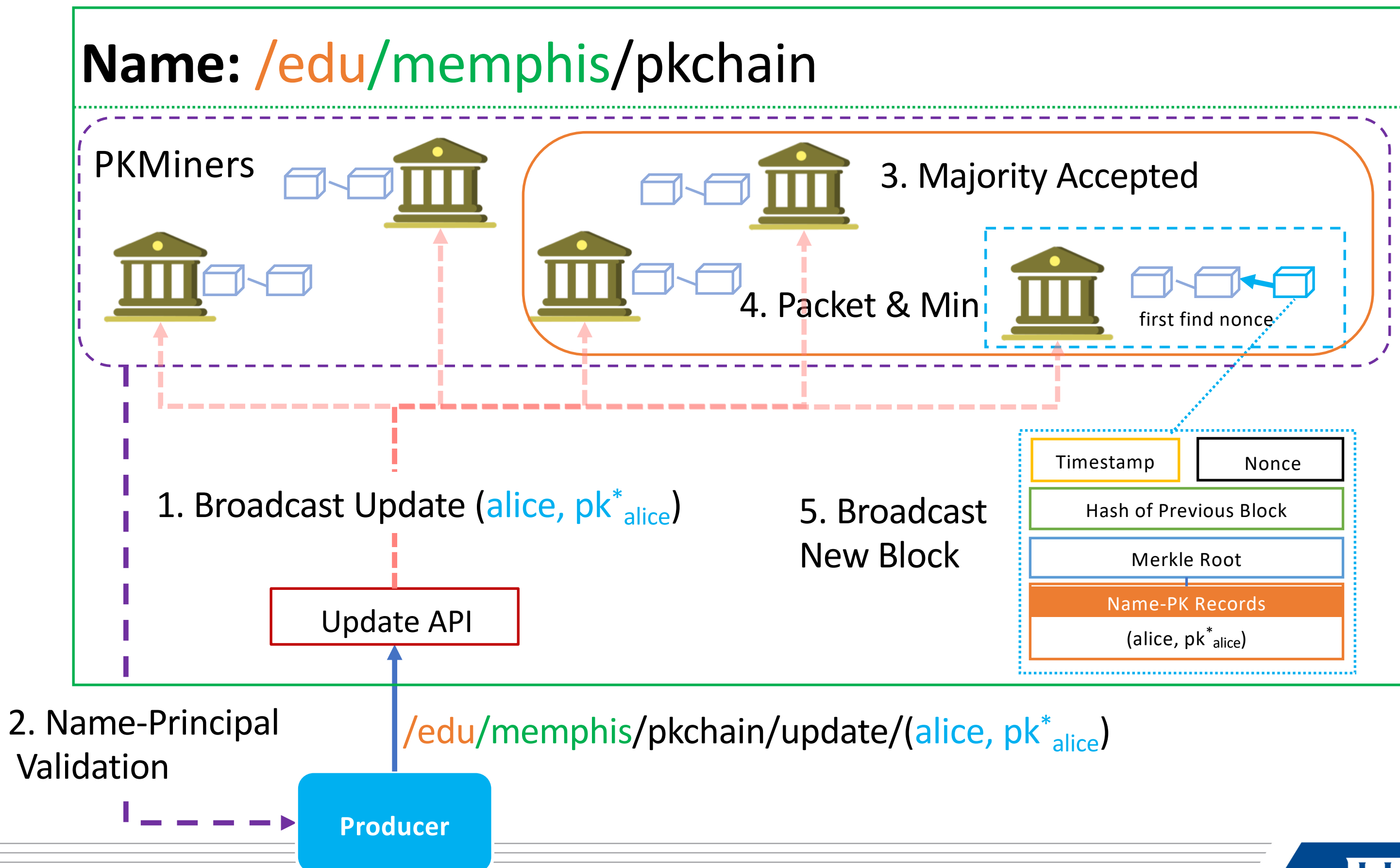


By name or by public key

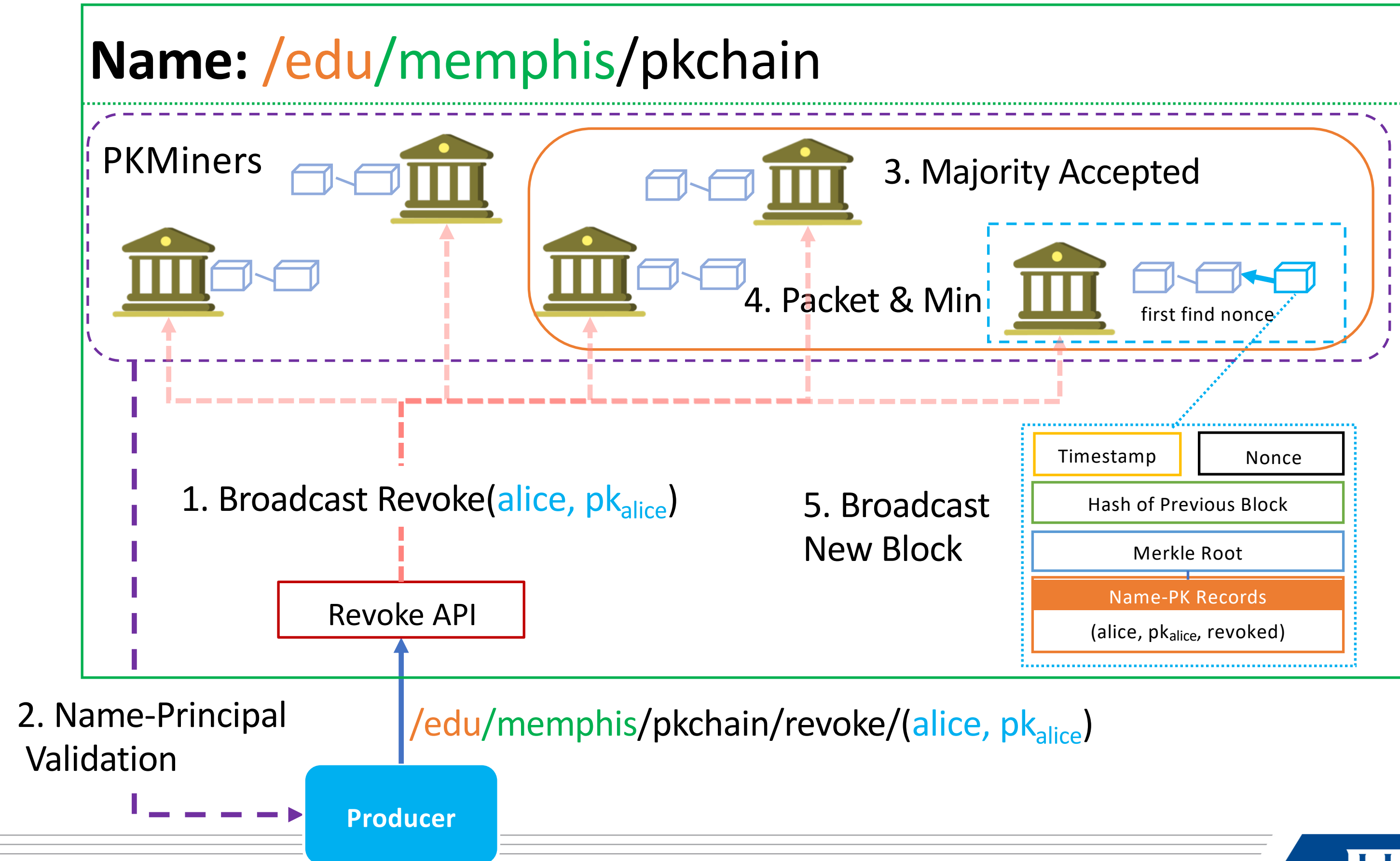
PKChain: Validate (query and compare)



PKChain: Update (add a new block)



PKChain: Revoke (add a revoking block)



Security Analysis

Theorem: BC-PKM can resist t *out of* n $(n > 2t - 1)$ *compromised PKMiners against*

- *registering public keys for fake principals*
- *illegally updating public keys for existing principals*
- *illegally revoking public keys for existing principals*

as long as there are more than half PKMiners are honest.

Guaranteed by the majority principle of the name-principal validation.

Refer to the paper for more details.

Prototype

- Implement by the Node.js framework
  - due to its asynchronous capabilities and ability to handle peer-to-peer communications well
  - The framework's event-driven, non-blocking I/O model makes it a good fit for our implementation.
- The command line interface was created with the help of a library called Vorpal (https://www.npmjs.com/package/vorpal)
- Note that, the main purpose of this prototype is to validate the functions of our BC-PKM system

```
jobin@MSI MINGW64 /d/BlockchainPKI (master)
$ node vorpal.js
BC-PKM$ help

  Commands:

    help [command...]                  Provides help for a given command.
    exit                               Exits application.
    start [port]                       Start or connect to the network
    register <name> <public_key>       Register a name with a public key
    update <name> <public_key>         Update an existing name with a new
                                       public key
    revoke <name> <public_key>         Revoke a name and public key
    validate <name> <public_key>       Check if a given name and public
                                       key pair is valid
    printchain                         Print the current blockchain
    updatechain                        Update the blockchain with the
                                       latest blockchain
    queryname <name>                   Query the blockchain for a name
    querypk <public_key>               Query the blockchain for a public
                                       key
    stop                               Exit the network

BC-PKM$ start
BC-PKM$ Starting the network....
A peer has connected to the network!
A peer has connected to the network!
BC-PKM$
BC-PKM$ register John 1xui45nal
Congratulations! A new block was mined.
BC-PKM$
```

Prototype: Register

PKMiner1

Timestamp	Mon, 06 Nov 2017 17:35:48 GMT
Name	Blockchain PKI
Public Key	0
Hash	0000849f7250903ea57f1614e1d16fc750a6c…
Nonce	113708
Miner ID	-1
Revoked	false

Block #1	
Previous Hash	0000849f7250903ea57f1614e1d16fc750a6c…
Timestamp	Tue, 17 Apr 2018 23:43:28 GMT
Name	John
Public Key	1xui45nal
Hash	000043792502902acf2839b11d39508548fa9…
Nonce	40951
Miner ID	0
Revoked	false

```
BC-PKM$
```

PKMiner2

Revoked	false

Block #1	
Previous Hash	0000849f7250903ea57f1614e1d16fc750a6c…
Timestamp	Tue, 17 Apr 2018 23:43:28 GMT
Name	John
Public Key	1xui45nal
Hash	000043792502902acf2839b11d39508548fa9…
Nonce	40951
Miner ID	0
Revoked	false

```
BC-PKM$ register Mary bqnas83p2a3iz
Congratulations! A new block was mined.
BC-PKM$
```

PKMiner3

Prototype: Query then Update

PKMiner1

Name	John
Public Key	1xui45nal
Hash	00007effc506666c3303f8e37ecbf0fa4ca12…
Nonce	23204
Miner ID	0
Revoked	false

Block #2	
Previous Hash	00007effc506666c3303f8e37ecbf0fa4ca12…
Timestamp	Tue, 17 Apr 2018 23:59:57 GMT
Name	Mary
Public Key	bqnas83p2a3iz
Hash	0000ec4f9e9f19dc5c79719643d72b6598db2…
Nonce	137170
Miner ID	1
Revoked	false

```
BC-PKM$
```

PKMiner2

Timestamp	Tue, 17 Apr 2018 23:59:57 GMT
Name	Mary
Public Key	bqnas83p2a3iz
Hash	0000ec4f9e9f19dc5c79719643d72b6598db2…
Nonce	137170
Miner ID	1
Revoked	false

```
BC-PKM$
```

PKMiner3

Public Key	bqnas83p2a3iz
Hash	0000ec4f9e9f19dc5c79719643d72b6598db2…
Nonce	137170
Miner ID	1
Revoked	false

```
BC-PKM$ queryname John
Name: John, Public Key: 1xui45nal
BC-PKM$ queryname Mary
Name: Mary, Public Key: bqnas83p2a3iz
BC-PKM$ querypk bqnas83p2a3iz
Name: Mary, Public Key: bqnas83p2a3iz
BC-PKM$ querypk 1xui45nal
Name: John, Public Key: 1xui45nal
BC-PKM$
```

Prototype: Updated View then Revoke

PKMiner1

Name	Mary
Public Key	bqnas83p2a3iz
Hash	0000ec4f9e9f19dc5c79719643d72b6598db2…
Nonce	137170
Miner ID	1
Revoked	false

Block #3	
Previous Hash	0000ec4f9e9f19dc5c79719643d72b6598db2…
Timestamp	Wed, 18 Apr 2018 00:01:40 GMT
Name	John
Public Key	7fhqnl3sg8gb
Hash	00004c2840e1a810372ce4cf553106ba790f8…
Nonce	134724
Miner ID	0
Revoked	false

```
BC-PKM$ |
```

PKMiner2

Timestamp	Tue, 17 Apr 2018 23:59:57 GMT
Name	Mary
Public Key	bqnas83p2a3iz
Hash	0000ec4f9e9f19dc5c79719643d72b6598db2…
Nonce	137170
Miner ID	1
Revoked	false

```
BC-PKM$
```

PKMiner3

Miner ID	1
Revoked	false

```
BC-PKM$ queryname John
Name: John, Public Key: 1xui45nal
BC-PKM$ queryname Mary
Name: Mary, Public Key: bqnas83p2a3iz
BC-PKM$ querypk bqnas83p2a3iz
Name: Mary, Public Key: bqnas83p2a3iz
BC-PKM$ querypk 1xui45nal
Name: John, Public Key: 1xui45nal
BC-PKM$ querypk 1xui45nal
Name: John, Public Key: 1xui45nal
The queried public key is no longer valid
BC-PKM$ querypk 7fhqnl3sg8gb
Name: John, Public Key: 7fhqnl3sg8gb
BC-PKM$
```

Prototype: Revoked View and Validate

PKMiner1

Timestamp	Wed, 18 Apr 2018 01:47:47 GMT
Name	John
Public Key	7fhqnl3sg8gb
Hash	0000c95be54c8450f267a90784c7861e92765…
Nonce	12158
Miner ID	0
Revoked	false

Block #4	
Previous Hash	0000c95be54c8450f267a90784c7861e92765…
Timestamp	Wed, 18 Apr 2018 01:47:59 GMT
Name	John
Public Key	7fhqnl3sg8gb
Hash	0000c1e867849c79b9be71d242971ba128f1c…
Nonce	137632
Miner ID	0
Revoked	true

```
BC-PKM$
```

PKMiner2

Hash	0000c1e867849c79b9be71d242971ba128f1c…
Nonce	137632
Miner ID	0
Revoked	true

```
BC-PKM$ validate John 7fhqnl3sg8gb
This pair is not valid.
BC-PKM$ validate Mary bqnas83p2a3iz
The pair is valid!
BC-PKM$ validate Mary 123456
This pair is not valid.
BC-PKM$
```

PKMiner3

Timestamp	Wed, 18 Apr 2018 01:47:12 GMT
Name	Mary
Public Key	bqnas83p2a3iz
Hash	0000452056c484b03e6027d94e5c0bd2a75be…
Nonce	21830
Miner ID	1
Revoked	false

```
BC-PKM$ queryname John
Name John is revoked!
BC-PKM$ querypk 7fhqnl3sg8gb
Public Key is revoked!
BC-PKM$
```

Conclusion and Future Work

- We proposed BC-PKM, a blockchain-based decentralized public key management system, for Named Data Networking.
- The BC-PKM can solve the compromised CA problem existing in traditional PKM systems and can tolerate less than half PKMiners are compromised by the adversary while keeping the system stable and secure.

In our future work, we will solve the following design questions:

- Who can be the miners/validators?
- How to validate a public key? How to do the name-principal validation?
- Which consensus mechanism should we use?

Thank You!

kan.yang@memphis.edu