Cryptographic Algorithms and Security Protocols for ICN

Christopher A. Wood
UC Irvine, PARC
woodc1@uci.edu, cwood@parc.com

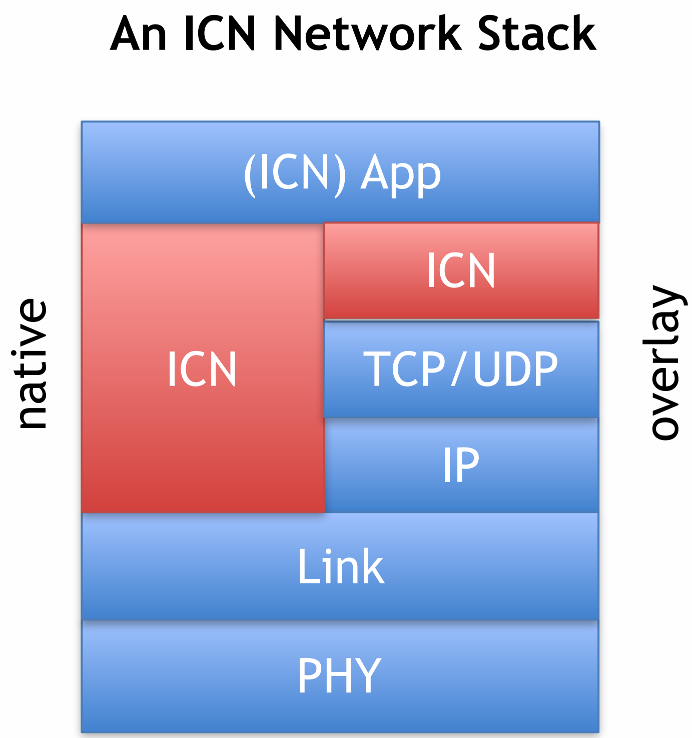
Background

- Security was not a goal of the IP protocol
  - Added after the fact via IPSec, SSL/TLS, etc.
- Existing IP security solutions are:
  - Rooted in old (long-standing) security solutions
  - Change and adapt very slowly
  - Only now starting to look ahead (PFS, PQ algorithms and protocols, etc.)

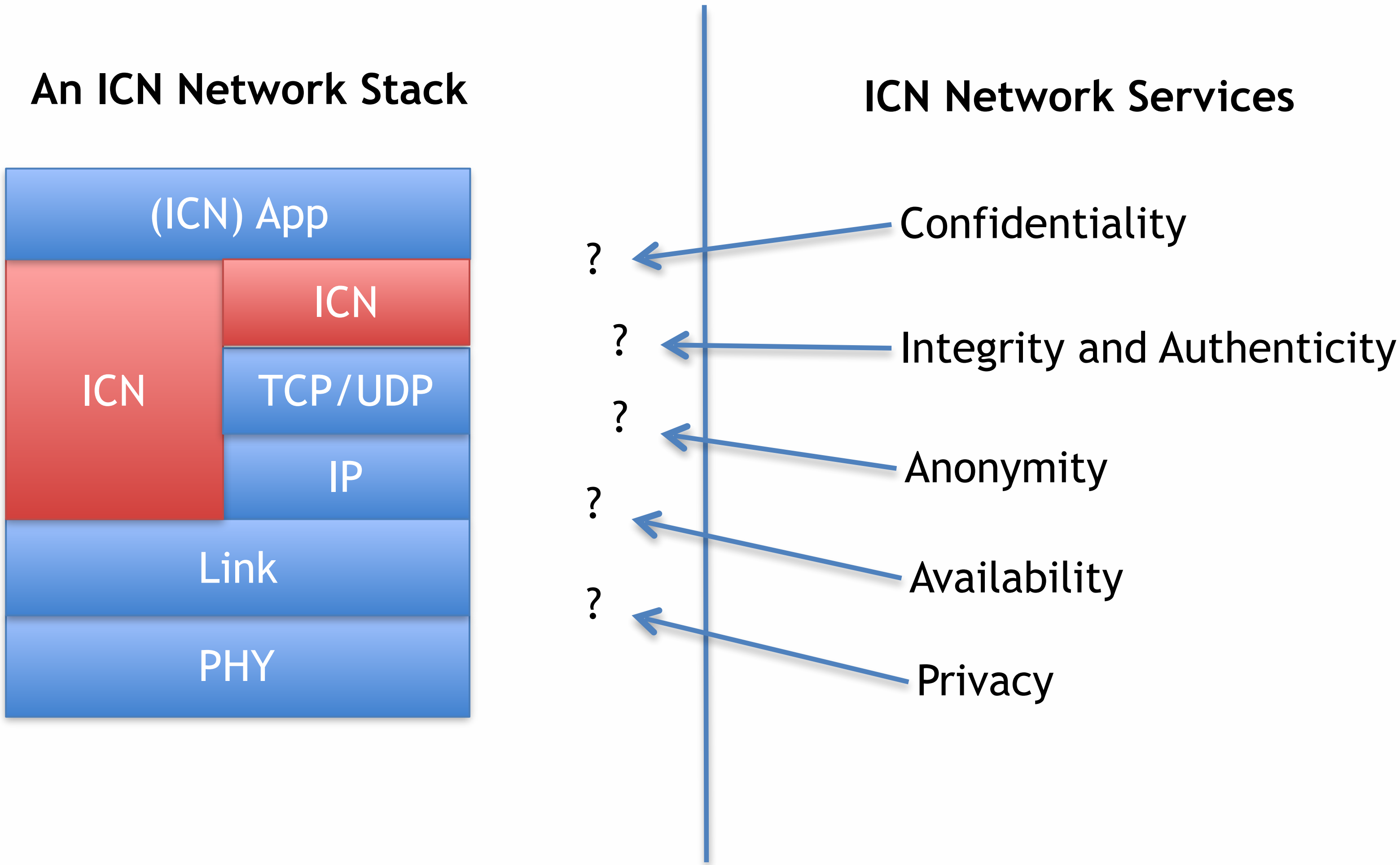
Looking Forward

- ICN architectures are:
  - Built on a **clean slate**
  - Can use **new and modern crypto**

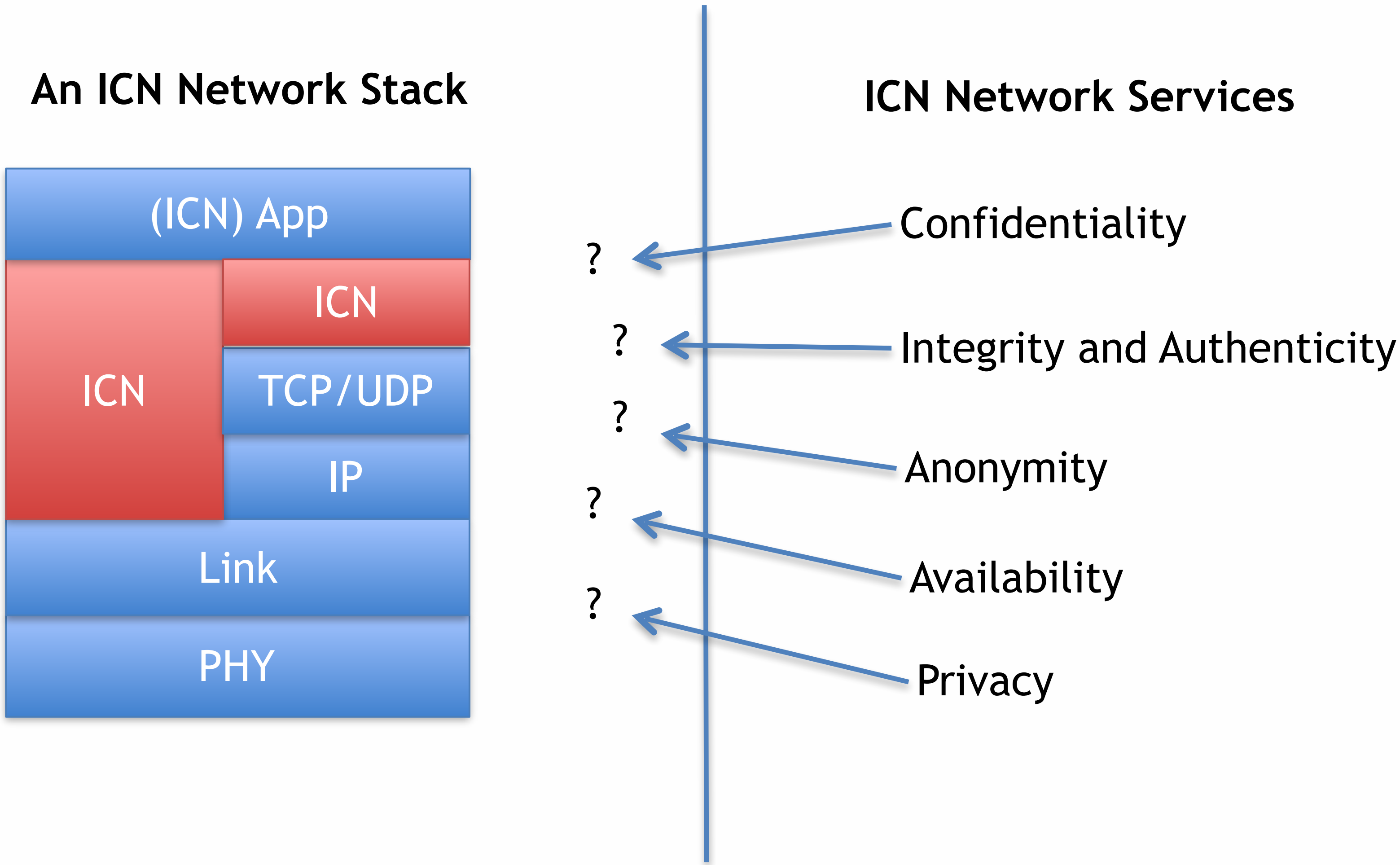
ICN Network Stack



ICN Network Services



ICN Network Services



Q: What crypto and security techniques can enable these core services?

Selection Criteria

- Goal: Crypto that **can enables essential network services**
- What's essential?
  - Integrity and authenticity
  - Anonymity
  - Privacy (e.g., as per RFC 6973)
  - Availability
- What's non essential?
  - Confidentiality (application-layer concern)

Topic Breakdown

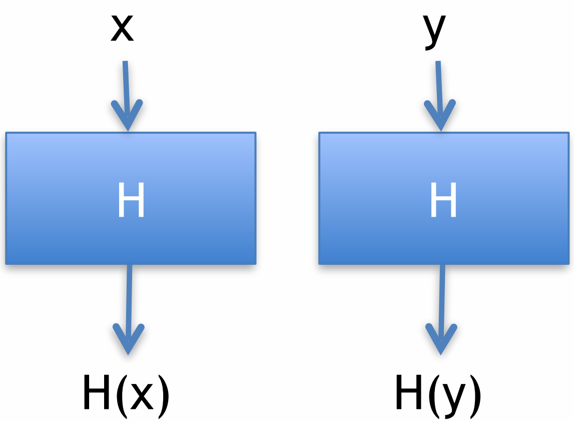
- Integrity and authenticity
  - Hash-based signatures
- Anonymity
  - TOR and Hornet
- Availability
  - Authenticated Denial of Existence (DoE)
- Privacy
  - Encrypted Deep Packet Inspection (DPI)
  - Password-Authenticated and Non-Interactive Key Exchange (PAKE and NIKE)
  - Private Information Retrieval (PIR)
  - Randomizable public-key encryption
  - Secure searchable encryption (SSE)
  - Oblivious data structures

Integrity and Authentication

Hash-Based Signatures

- Traditional signature schemes are based on trapdoor functions
  - RSA, DSA, ECDSA, etc.
  - PQ-secure?
- Hash-based signatures are quantum-secure, e.g., they don't fall to Shor's algorithm
- Based on one-time signatures (OTSs)
  - A key pair can be used only once

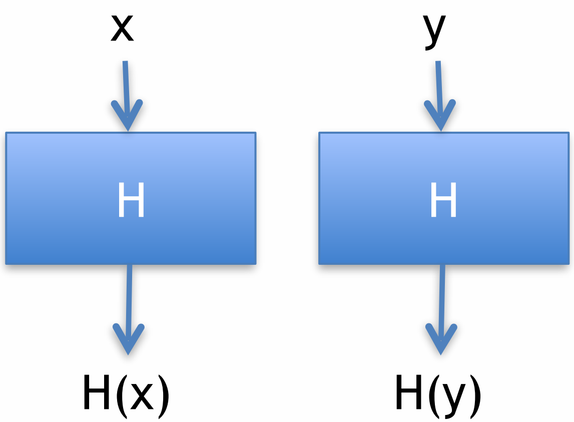
Lamport OTS Idea [x]



Private key: (x, y)

Public key: (H(x), H(y))

Lamport OTS Idea

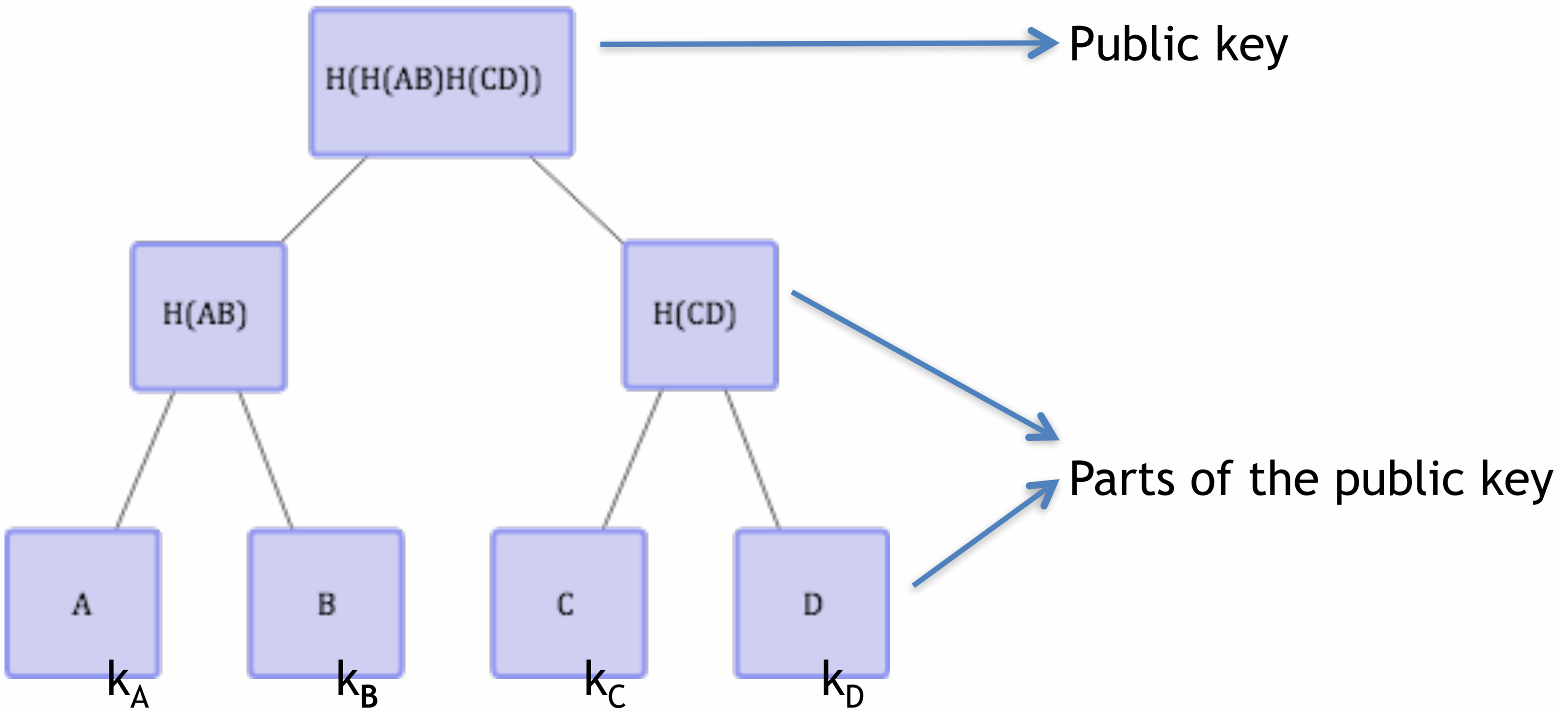


Private key: (x, y)

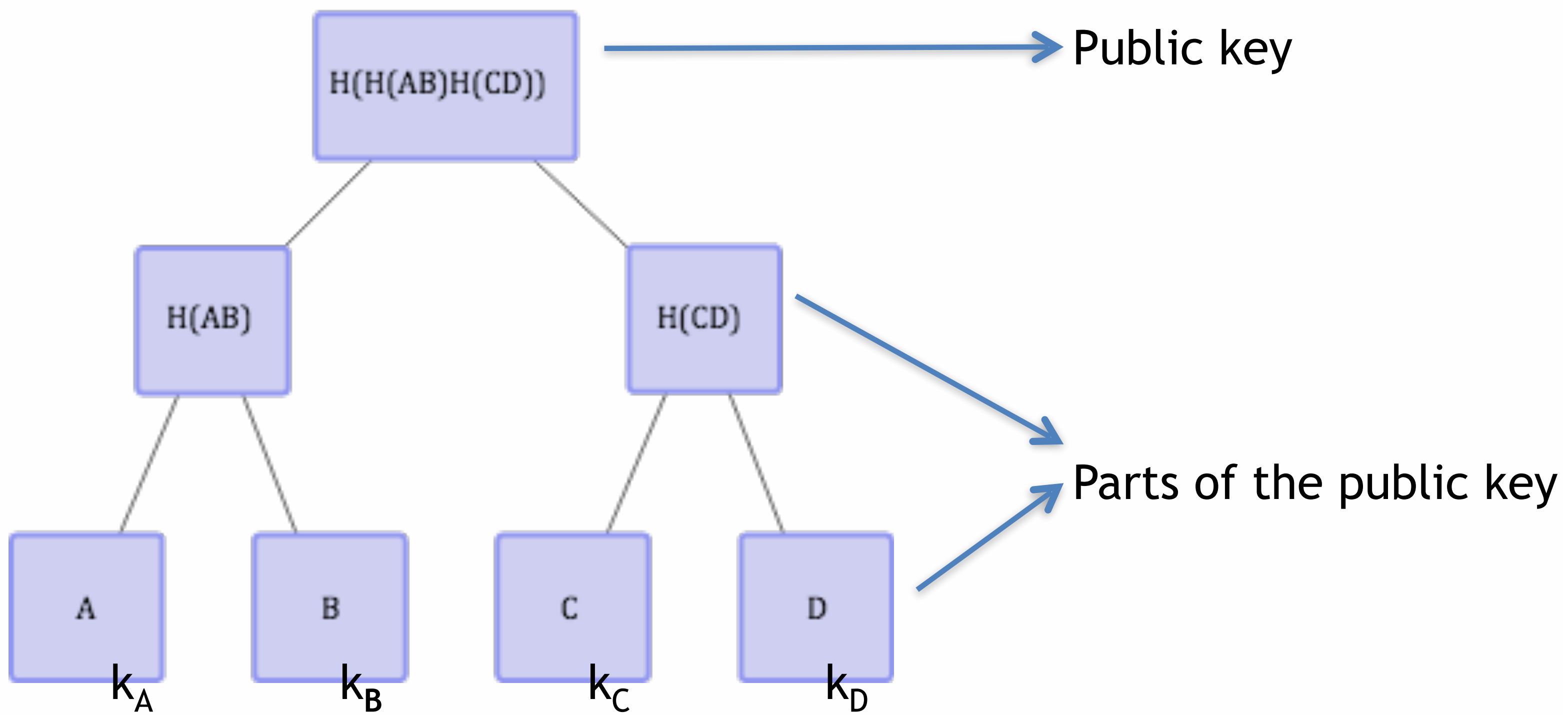
Public key: (H(x), H(y))

To sign a ‘0’: release x
To sign a ‘1’: release y

Merkle Tree Idea

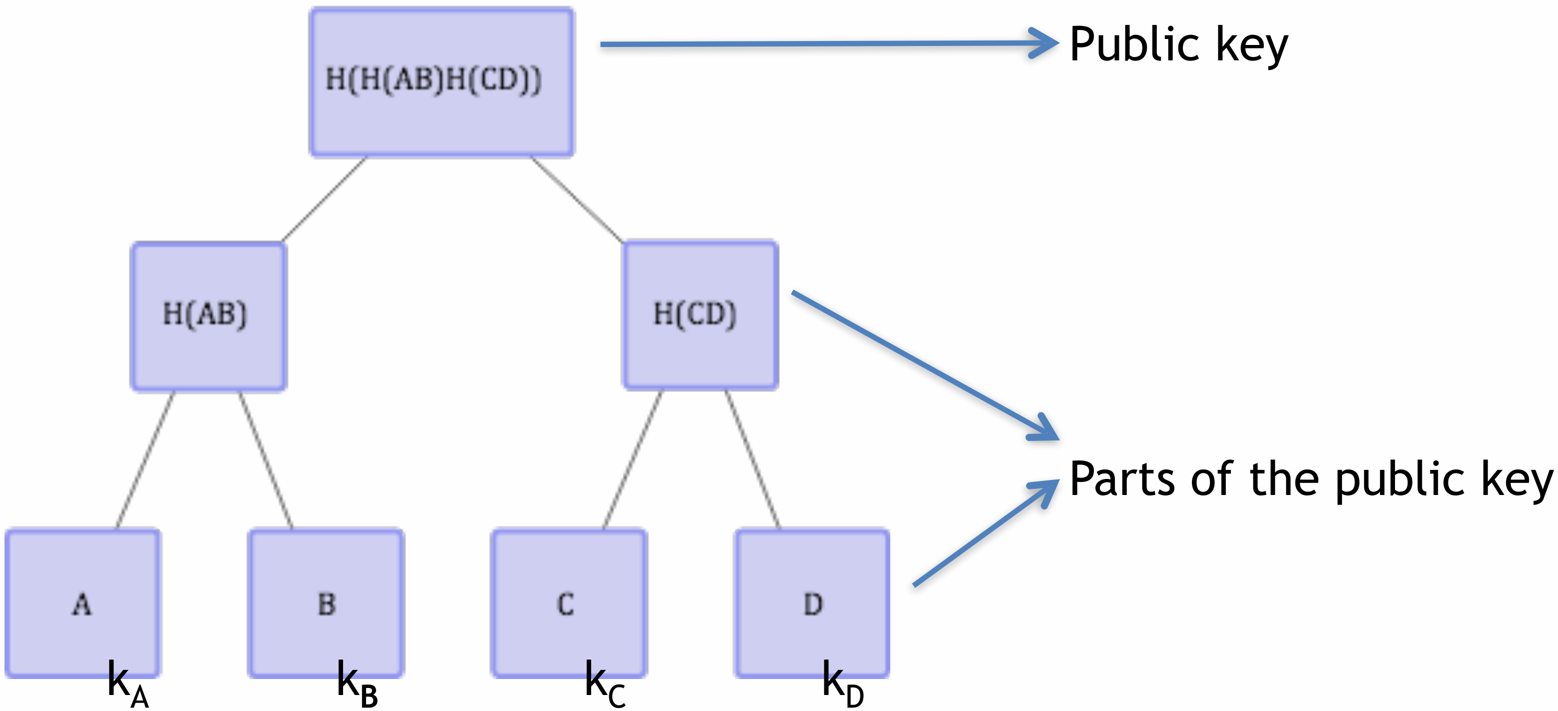


Merkle Tree Idea



To sign the first message m1: provide (A, B, H(CD), $sign_{kA}$(m1))

Merkle Tree Idea

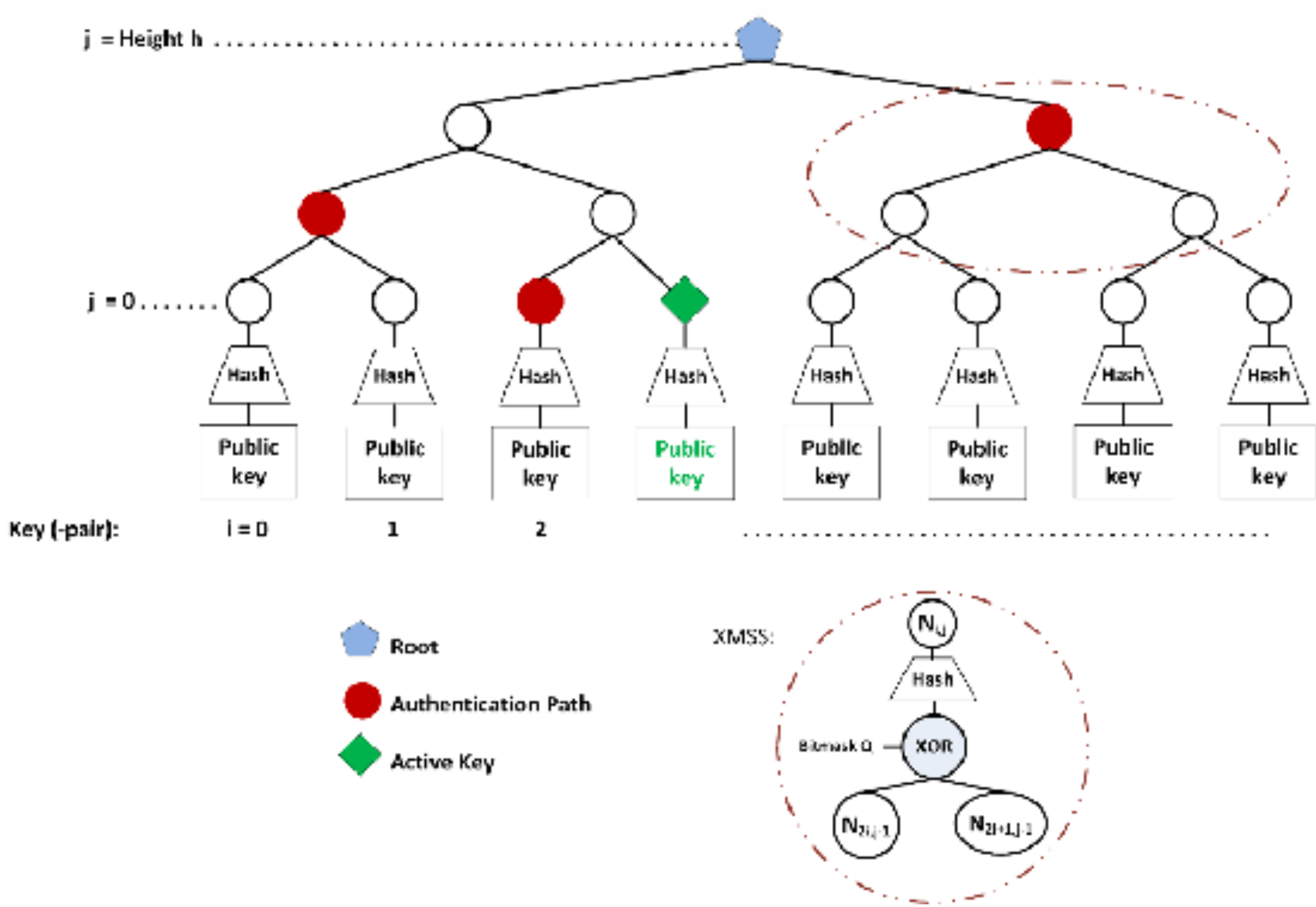


To sign the third message m3: provide (H(AB), C, D, $sign_{kA}$(m3))

Merkle Trees [x]

- The tree dimensions determine the number of messages that can be signed
- Other drawbacks:
  - It is secure in the Random Oracle model
  - The length is **proportional to the number of messages** that are signed
  - It is **stateful**
    - Store the index and intermediate tree results

XMSS Tree (Buchmann et al.)



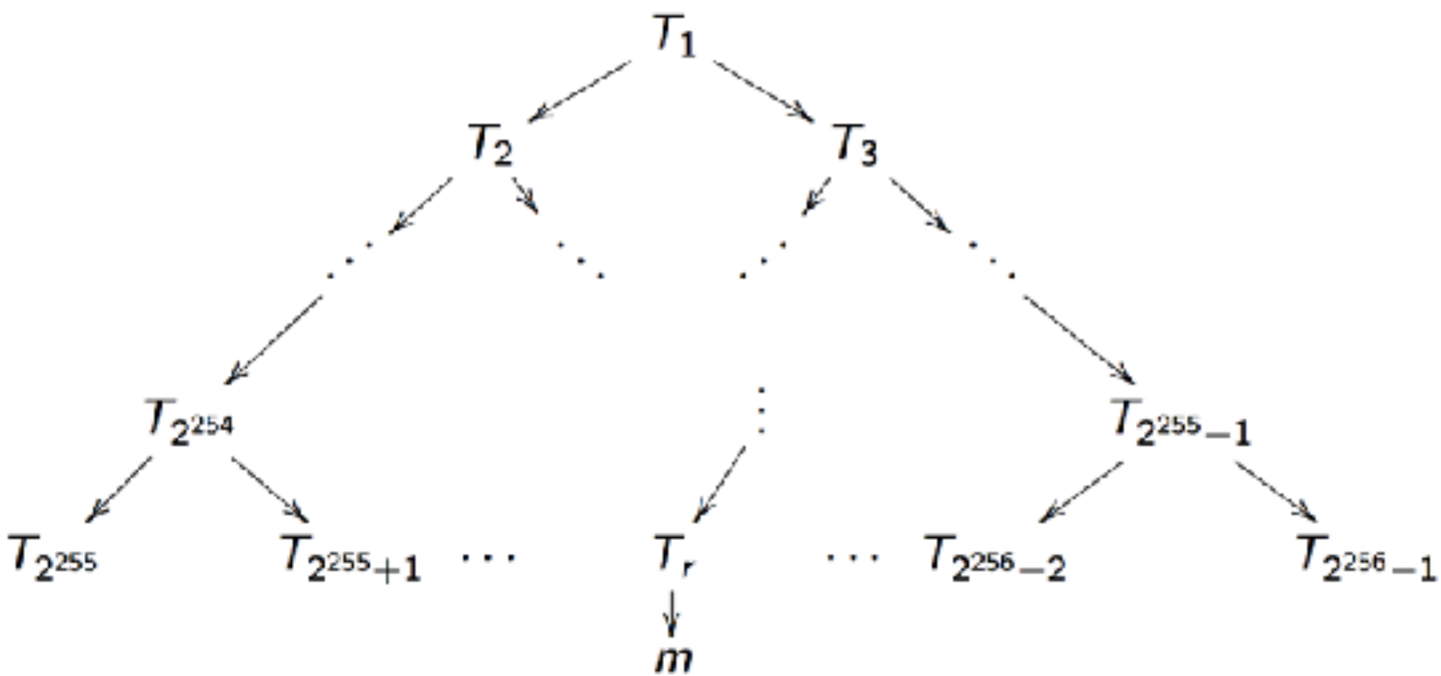
Borrowed from:
http://www.square-up.org/index/hbs.html

Stateless Hash-Based Signature

- Similar to Merkle trees but the "index" is chosen at random
- Requires huge trees to avoid collisions
  - For security parameter λ=128 we require $2^{2\lambda}$ leaves (by birthday paradox)
- OTS secret keys are generated pseudorandomly

Overview

Signer chooses random $r \in \{2^{255}, 2^{255}+1, \ldots, 2^{256}-1\}$,
uses one-time public key T_r to sign message;
uses one-time public key T_i to sign (T_{2i}, T_{2i+1}) for $i < 2^{255}$.
Generates ith secret key as $H_k(i)$ where k is master secret.



SPINCS (Bernstein et al.)

- Stateless hash-based signature
  - Inspired by Goldreich stateless hash-based scheme [X]
- Dramatically reduces tree size
- Replace one-time leaves with “few-time leaves”
- Use XMSS-like per-node masks

Hash-Based Signature Recap

- When to use?
  - Long-term security in a post-quantum world
- Will ICN data packets live forever?

Anonymity

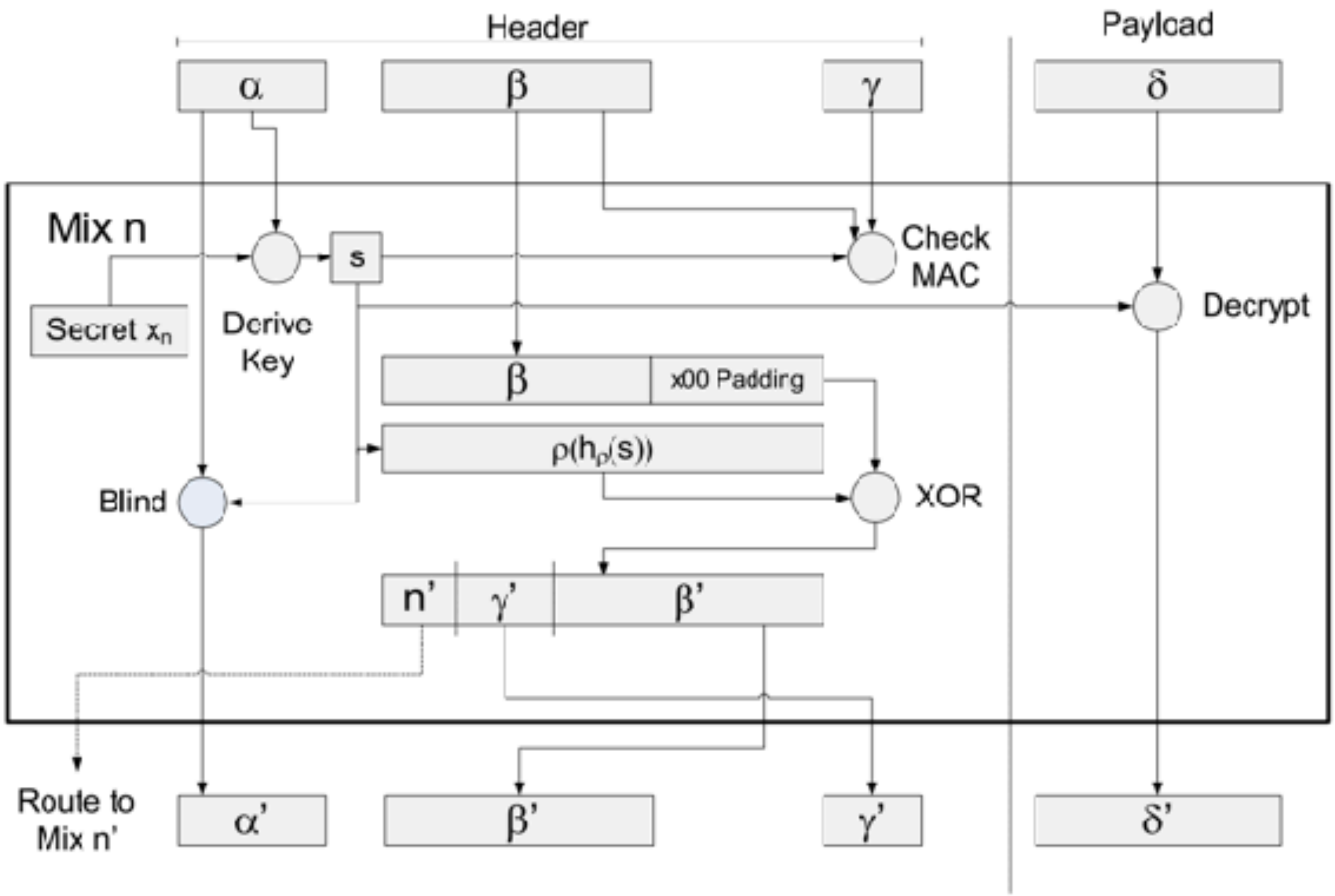
Onion Routing

- Based on Chaum's mixnets
- TOR (The Onion Router) is the most ubiquitous variant
  - Not provably secure
  - Performance could be improved

Sphinx Protocol [x]

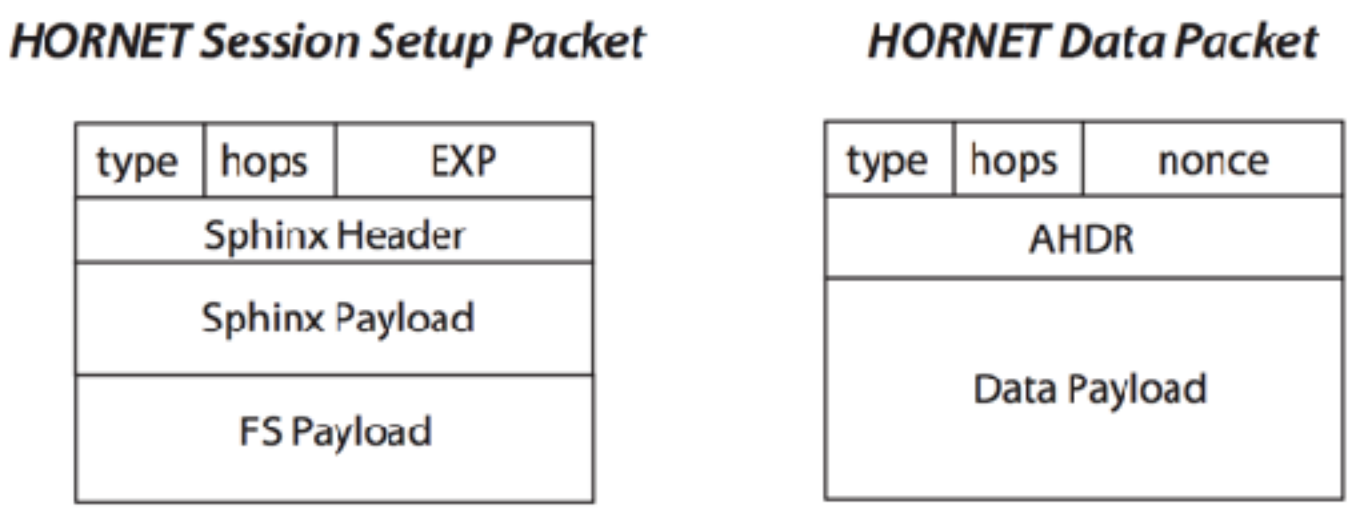
- Provably secure mix protocol
- Mixture of public- and symmetric-key cryptography for each packet
  - Derive per-packet MAC and encryption keys for each anonymizer
  - Onion-encrypt the payload with a large-block-size block cipher
  - Compute the MAC carefully
  - Wrap the derivation seed using key encapsulation

Sphinx Protocol

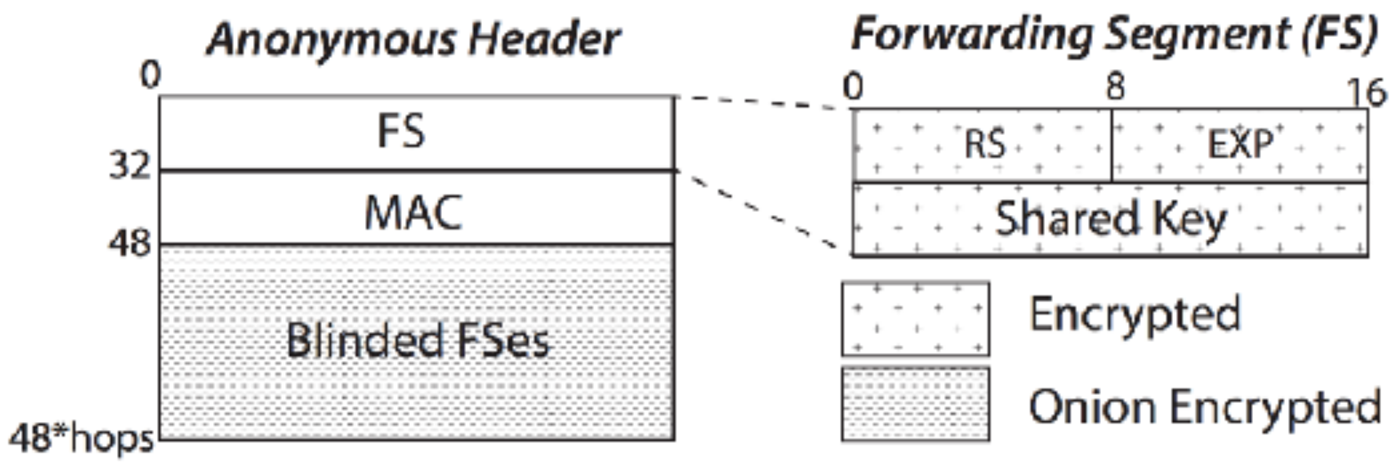


HORNET Protocol [x]

- Built on Sphinx to establish symmetric keys with each anonymizer
  - Extended to include per-hop “forwarding segments” used to process packets in a circuit
- Data packets are created with onion-encrypted payloads and anonymous headers (ADHR)



HORNET Data Packet Internals



Onion Routing Recap

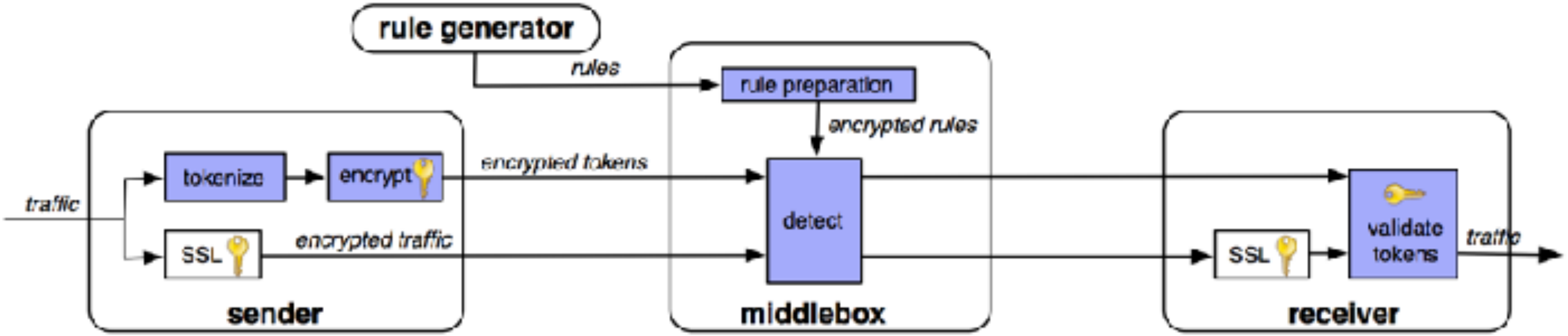
- ICN names reveal **too much**
- Use onion routing to onion-encrypt individual packets
  - G. Tsudik, E. Uzun, and C. A. Wood, AC3N: An API and Service for Anonymous Communication in Content-Centric Networking, in the Proceedings of CCNC 2016, Las Vegas, NV, USA, 2016.
  - S. DiBenedetto, P. Gasti, G. Tsudik and E. Uzun, ANDaNA: Anonymous Named Data Networking Application, the 19th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, 2012.
- … but do it right as a network service

Privacy

Encrypted DPI

- Perform deep-packet-inspection on encrypted packet payloads
  - Determine when a packet contains an encrypted version of a specific keyword
- Different measures of privacy
  - Exact-match privacy: only discover bytes that match target keywords
  - Probable cause privacy: decrypt a flow (entire packet) only if a keyword match is detected

BlindBox


rule generator
rules
rule preparation
encrypted rules
traffic
tokenize
encrypt
encrypted tokens
SSL
encrypted traffic
detect
SSL
validate tokens
traffic
sender
middlebox
receiver

BlindBox Details

- Packets are tokenized into tokens t1,…,tk
- For each token t, encrypt as follows

$$\text{salt}, \text{AES}_{\text{AES-k(t)}}(\text{salt}) \longrightarrow$$

ensures randomness of identical tokens

- To detect a token t, precompute salt and token combinations
- Speedup: use a single salt per token and then derive subsequent token encryptions based on a counter

Probable Cause Decryption

- Idea: embed message encryption key Dk in the encryption of each token

$$\text{salt}, (AES_{AES\text{-}k(t)}(\text{salt}) \text{ XOR } Dk)$$

BlindBox Rule Preparation

- The middlebox must learn AES-k(t) without revealing the encryption key k
- Solution: Garbled circuit to compute

AES-k(t)

for each token t

Privacy: Key Exchange

Password AKE

- Goal: create ephemeral keys from shared secrets – passwords (read: name) – in a way that is:
  - Not susceptible to offline dictionary attacks
  - Forward secure
- Most protocols are multi-round, e.g., J-PAKE
- Some protocols work in a **single round** without sacrificing forward secrecy

Non-Interactive Key Exchange (NIKE)

- Goal: Two parties with knowledge of each other's public keys agree on a shared secret without requiring any interaction [1]

  Alice: x, g^x

  Bob: y, g^y

  Shared key: H("Alice", "Bob", g^{xy})

- Use case: WSN shared key derivation
- Public-key encryption follows from NIKE

NIKE Protocols

- Setup: generate system parameters
- KeyGen: generate a private and public key pair for a given identity
- SharedKey: given (1) an identity, (2) its public key, (3) another identity, (4) and its secret key, compute and output a shared key

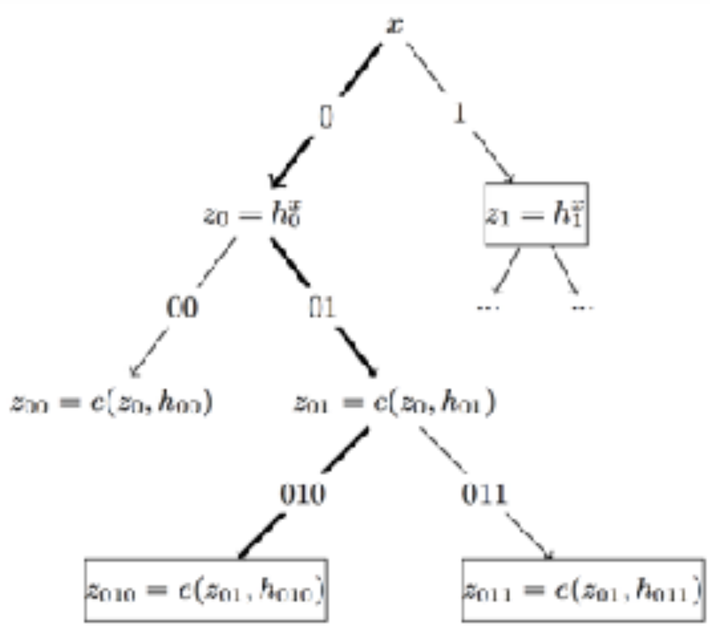
NIKE Protocols

- Setup: generate system parameters
- KeyGen: generate a private and public key pair for a given identity
- SharedKey: given (1) an identity, (2) its public key, (3) another identity, (4) and its secret key, compute and output a shared key

What's in an ICN certificate?...

Forward Secrecy?

- Yes – it's possible: evolve the public keys.
- Use <u>multilinear</u> maps and the tree-based key derivation technique
- Add an Update function to the scheme:
  - Move the secret key *forward* in time and space

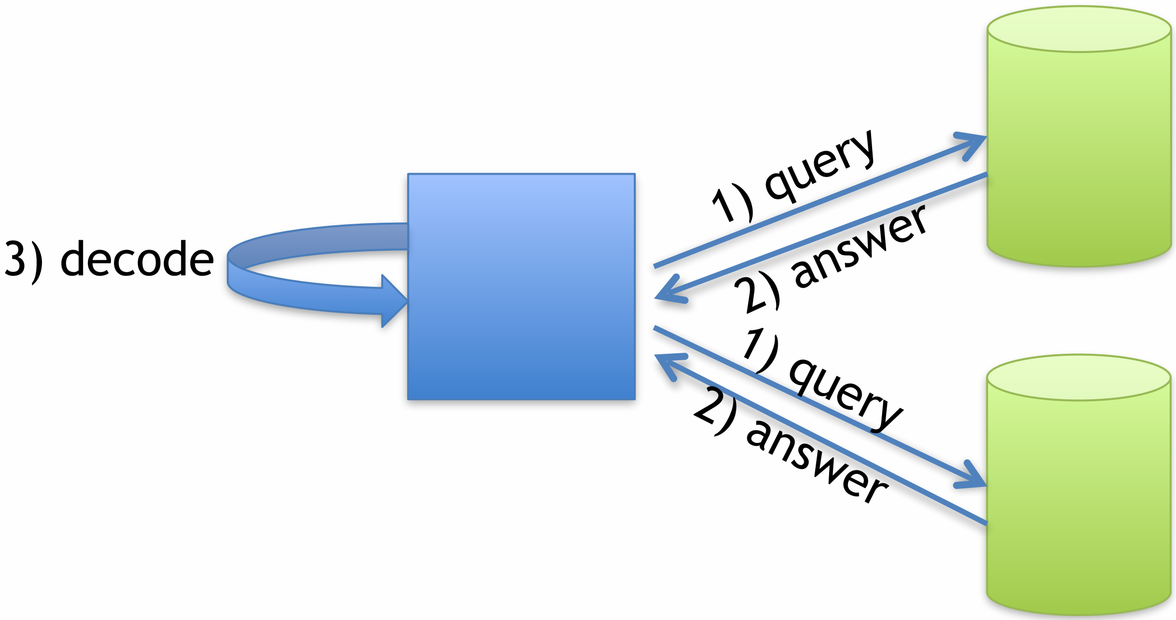


Privacy: PIR

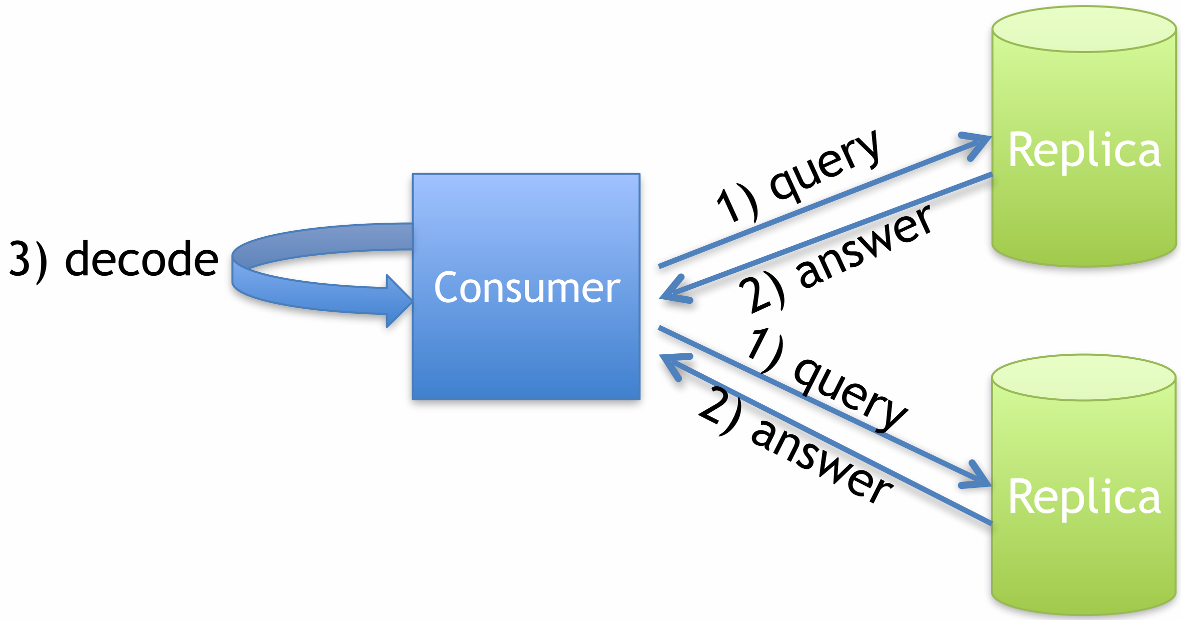
Private Information Retrieval

- A protocol to hide client requests to a server (e.g., a database)
- Two variants: computational (CPIR) and information-theoretic (IPIR)
- CPIR
  - Require only a single server
  - Much more computationally expensive
- IPIR
  - Require at least two non-colluding servers
  - More efficient but requires more communication

IPIR



IPIR



C. Tschudin, Private Information Retrieval over ICN, INFOCOM 2016 NOM Workshop, April, 2016.

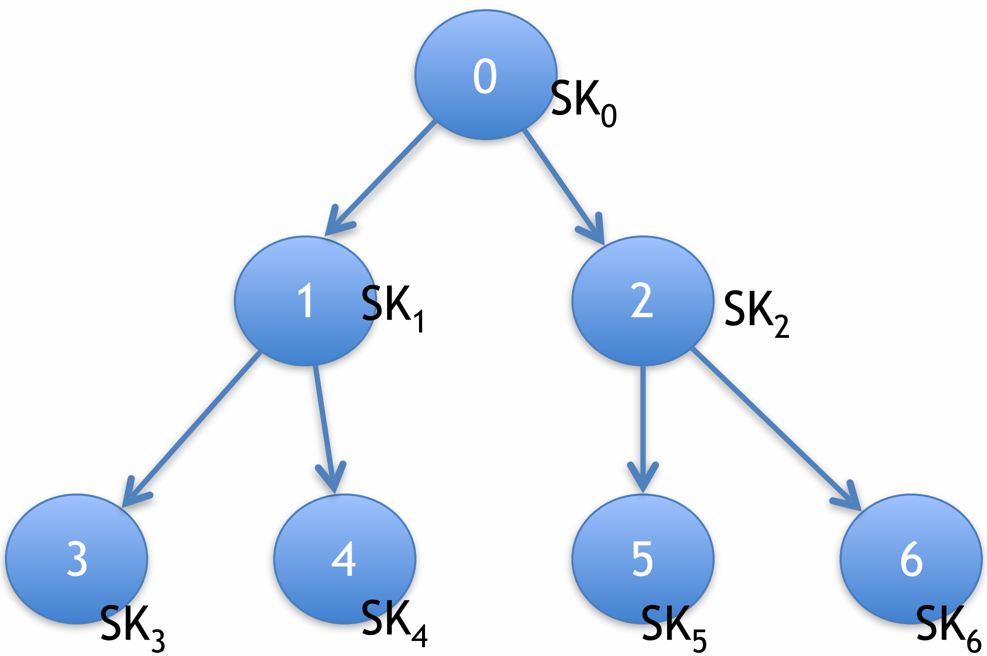
Privacy: Randomizable Encryption

Forward-Secure Public Key Encryption

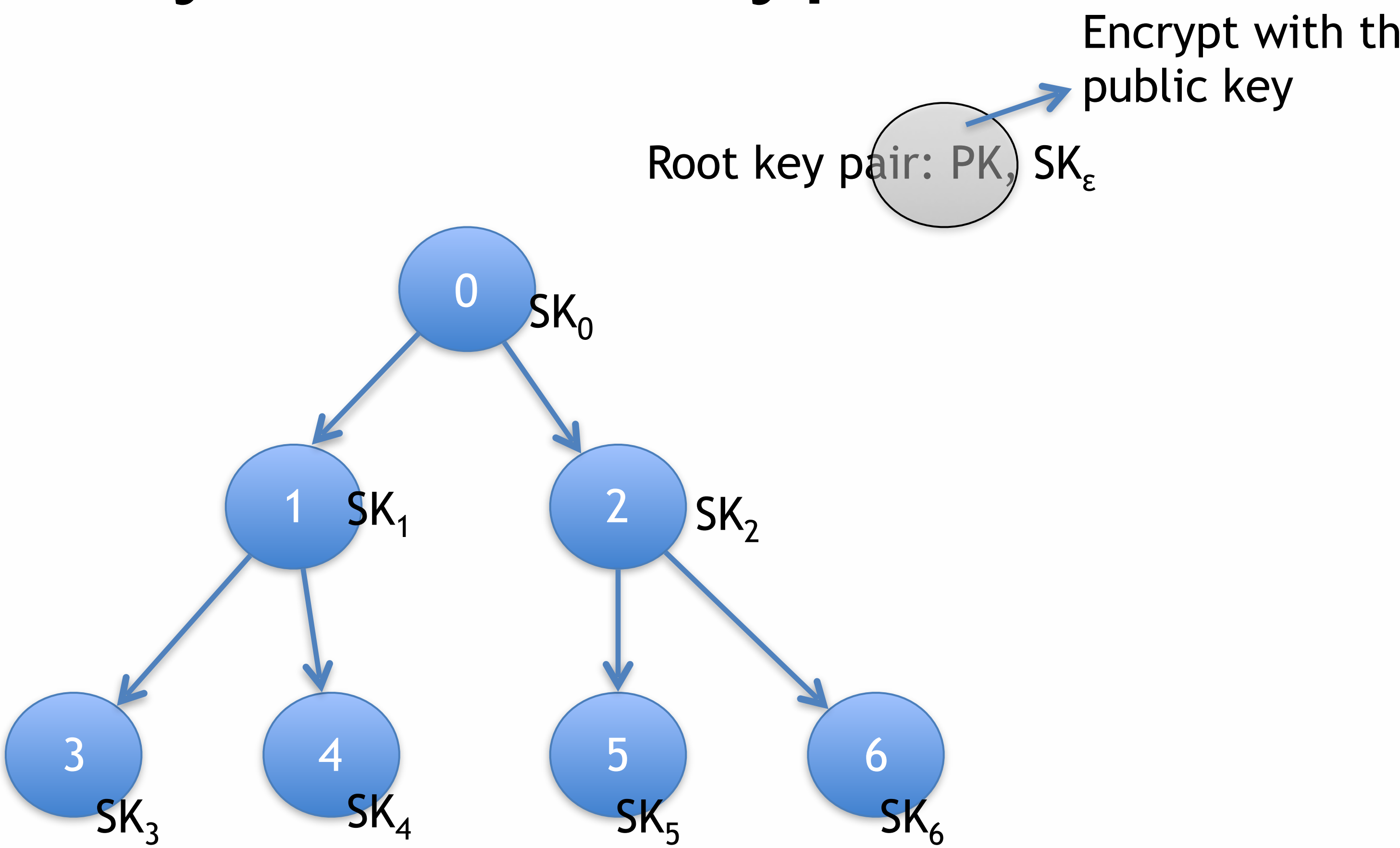
- In plain English, “key-evolving public key encryption”
- Consists of four algorithms:
  - Key generation
  - Key update
  - Encrypt
  - Decrypt

Binary Tree Encryption

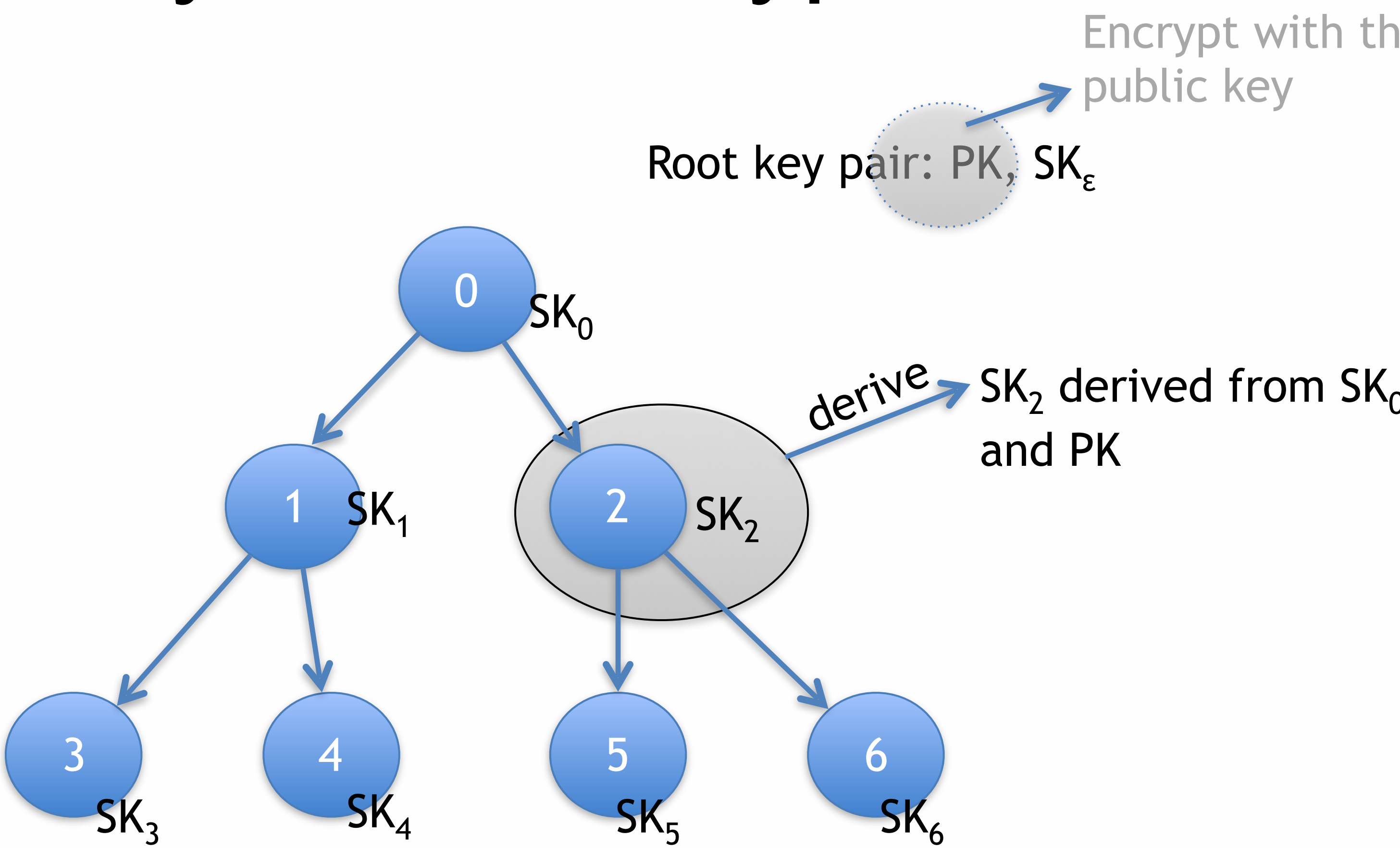
Root key pair: PK, SK_ε



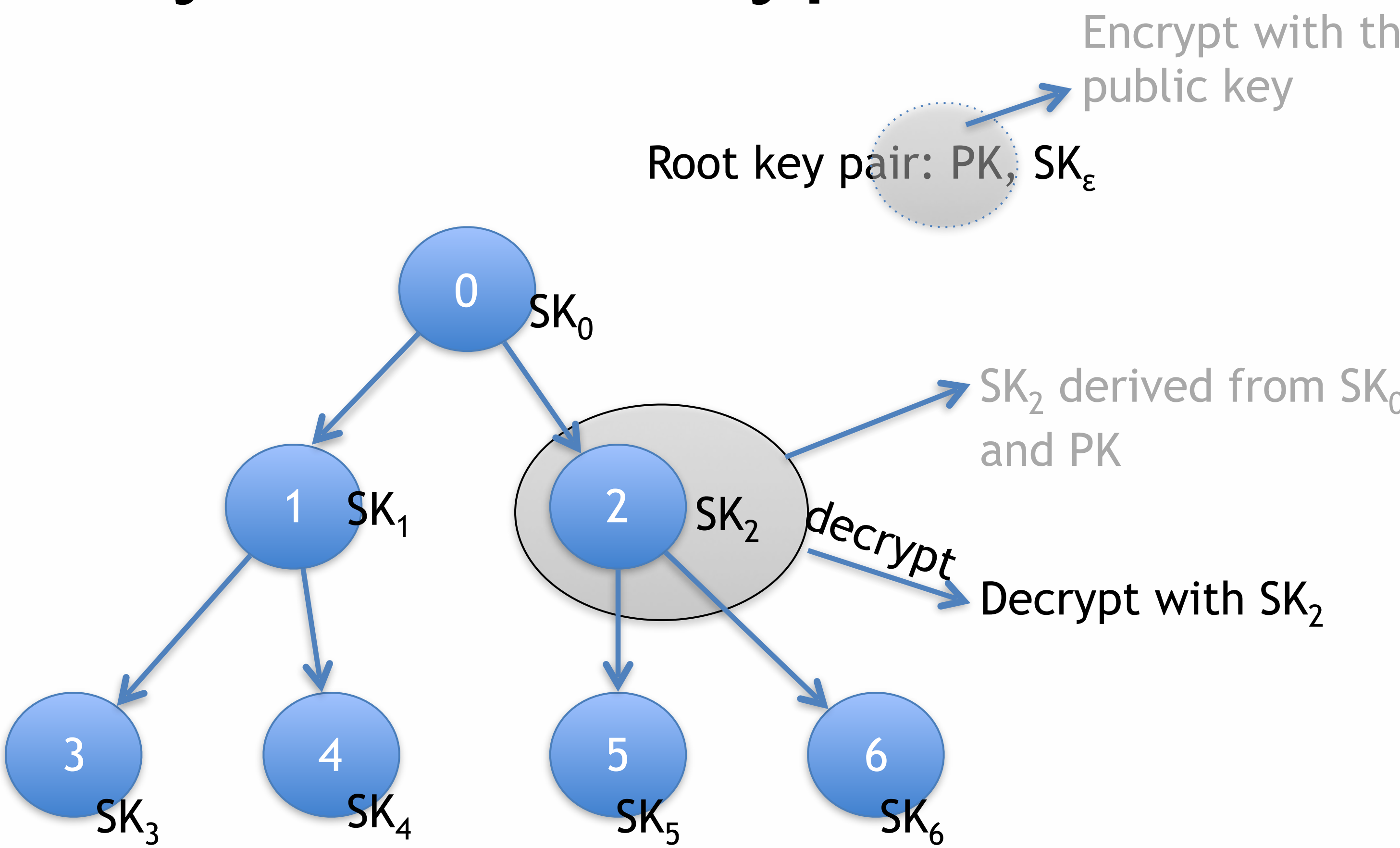
Binary Tree Encryption



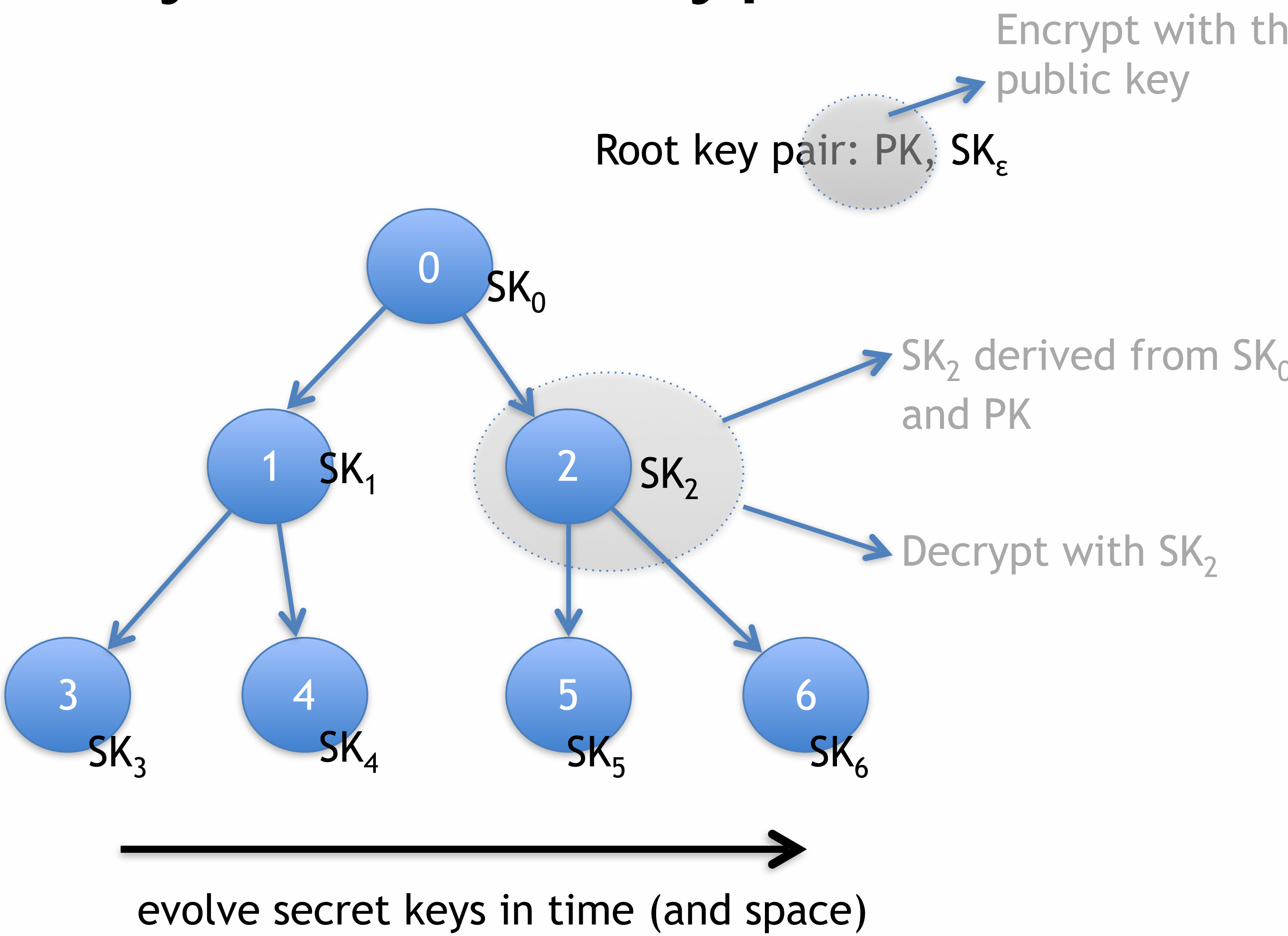
Binary Tree Encryption



Binary Tree Encryption



Binary Tree Encryption



Randomizable Public-Key Encryption

- Many encryption schemes can be re-randomized, e.g., ElGamal, BGN, etc.
- How can we maintain integrity after randomizing ciphertext?
  - *Randomizable signatures*

Randomizable Signatures

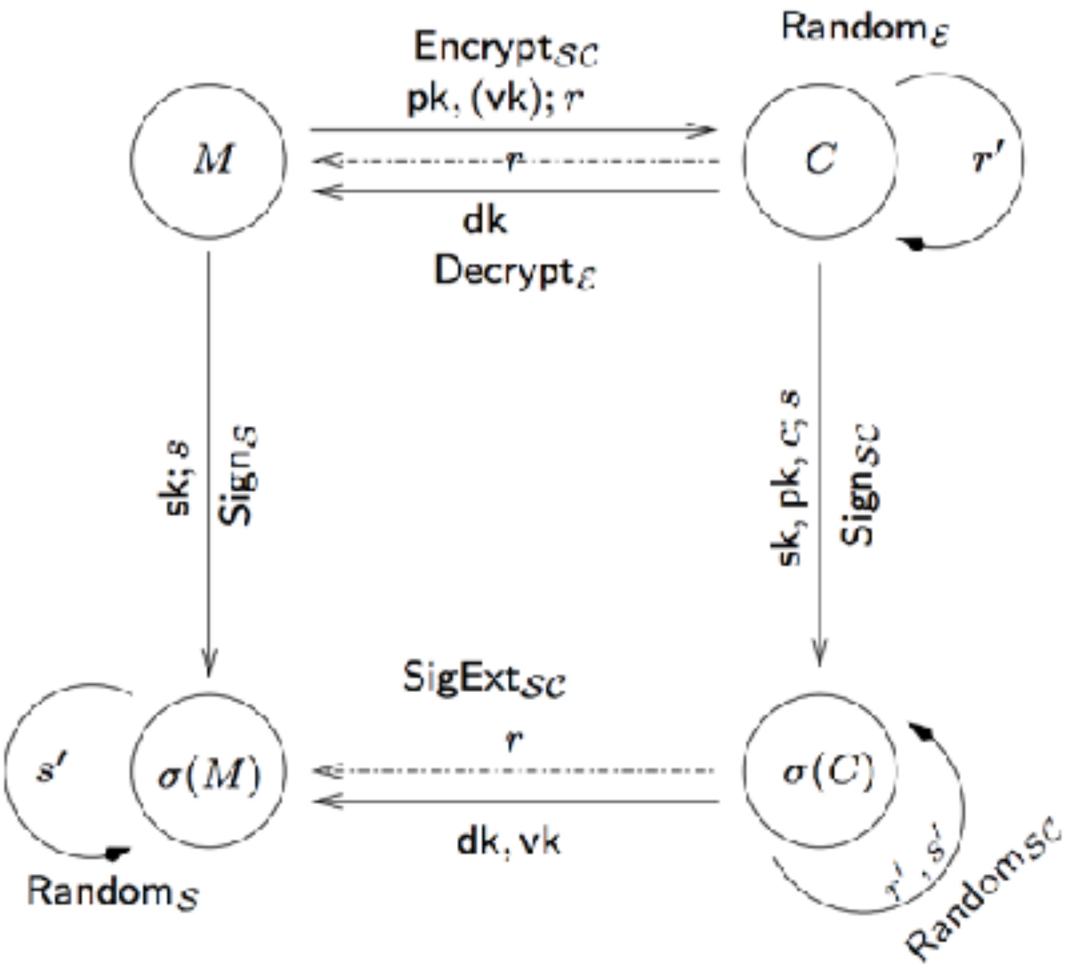
- Given a ciphertext, anyone can re-randomize the ciphertext and adapt the signature the new encryption, i.e.,

$$\mathcal{D}_0 = \{r' \xleftarrow{\$} \mathcal{R}_e; s' \xleftarrow{\$} \mathcal{R}_s : (c' = \mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, m; r'), \sigma' = \mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c'; s'))\}$$

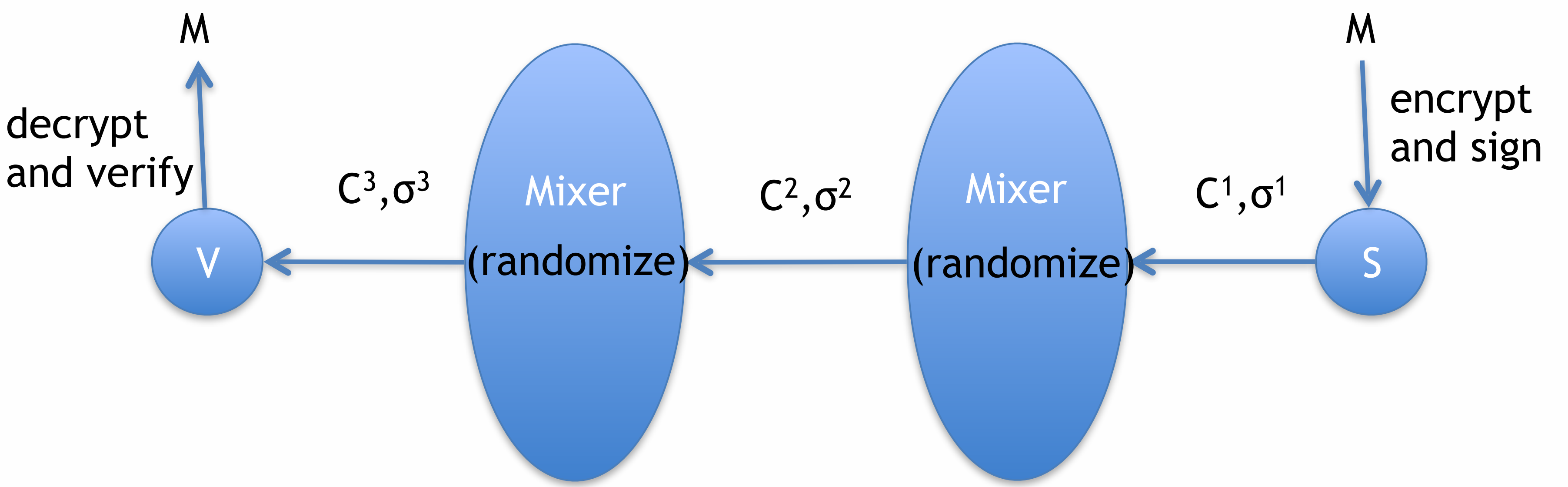
$$\mathcal{D}_1 = \{r' \xleftarrow{\$} \mathcal{R}_e; s' \xleftarrow{\$} \mathcal{R}_s : (c', \sigma') = \mathsf{Random}(\mathsf{vk}, \mathsf{pk}, c, \sigma; r', s')\}$$

are statistically indistinguishable

Extractable Signatures



Mixing in Motion



Privacy: SSE

Searchable Symmetric Encryption

- Goal: search over encrypted data using symmetric-key cryptography
- Many variants:
  - Interactive and non-interactive
  - Response-revealing and response-hiding

Searchable Symmetric Encryption

- Goal: search over encrypted data using symmetric-key cryptography
- Given a database of (encrypted) documents and list of keywords, identify the documents that contain keywords
- Many variants:
  - Interactive and non-interactive
  - Response-revealing and response-hiding

SSE Overview

- Client possess document list $D_1,\ldots,D_n$
- Client builds an index (DB) that maps keywords w to documents – DB[w]
- Protocol mechanics:
  - Setup: return key k and encrypted DB (eDB)
  - Token: on input w and k, return token t
  - Search: on input eDB and t, **the server** returns the identifiers in DB[w]
- Complexity: search is at best sub-linear in the number of documents and linear in those containing the keyword (**optimal**)

Other Solutions

Properties	[35, 25]	[35, 25]-light	[40]	[23]	[18]	SSE-1	SSE-2
hides access pattern	yes	yes	no	no	no	no	no
server computation	$O(\log^3 n)$	$O(\sqrt{n})$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
server storage	$O(n \cdot \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
number of rounds	$\log n$	2	1	1	1	1	1
communication	$O(\log^3 n)$	$O(\sqrt{n})$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
adaptive adversaries	yes	yes	no	no	no	no	yes

Other Solutions

Properties	[35, 25]	[35, 25]-light	[40]	[23]	[18]	SSE-1	SSE-2
hides access pattern	yes	yes	no	no	no	no	no
server computation	$O(\log^3 n)$	$O(\sqrt{n})$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
server storage	$O(n \cdot \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
number of rounds	$\log n$	2	1	1	1	1	1
communication	$O(\log^3 n)$	$O(\sqrt{n})$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
adaptive adversaries	yes	yes	no	no	no	no	yes

…but the search complexity is always linear in the number of matching documen

Non-Symmetric Encrypted Search

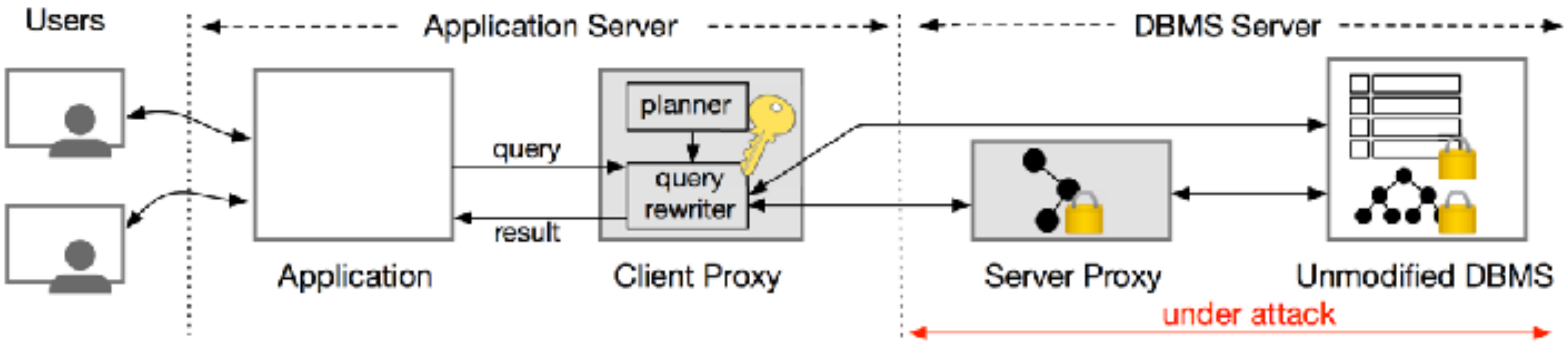
- Two variants: Practical and “not so much”
- Theoretical: Based on ORAM
  - Seek to hide everything except the result size
  - Often multi-round, *stateful*, and have heavy communication costs
- Practical: CryptDB and Arx (CryptDB v2)
  - These are actually deployed

ORAM

Technique	Example Applications	Client-side storage	Blowup
Pointer-based for rooted tree access pattern graph	`map/set`, `priority_queue`, `stack`, `queue`, oblivious memory allocator	$O(\log N)\cdot\omega(1)$	$O(\log N)$
Locality-based for access pattern graph with doubling dimension dim	maximum flow, random walk on sparse graphs; shortest-path distance on planar graphs; `doubly-linked list`, `deque`	$O(1)^{\mathsf{dim}}\cdot O(\log^2 N)+O(\log N)\cdot\omega(1)$	$O(12^{\mathsf{dim}}\log^{2-\frac{1}{\mathsf{dim}}} N)$
Path ORAM [45]	All of the above	$O(\log N)\cdot\omega(1)$	$O(\frac{\log^2 N+\chi\log N}{\chi})$ for block size $\chi\log N$
ORAM in [29]	All of the above	$O(1)$	$O(\frac{\log^2 N}{\log\log N})$

Borrowed from:
https://eprint.iacr.org/2014/185.pdf

Arx



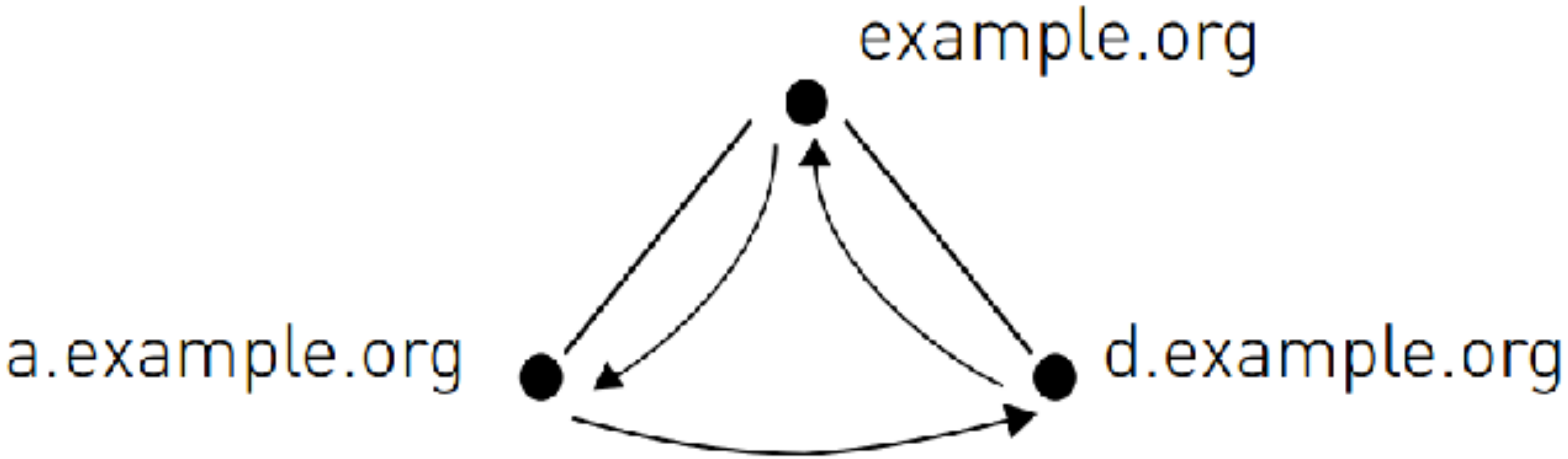
Users
Application Server
DBMS Server
planner
query
rewriter
query
result
Application
Client Proxy
Server Proxy
Unmodified DBMS
under attack

Availability

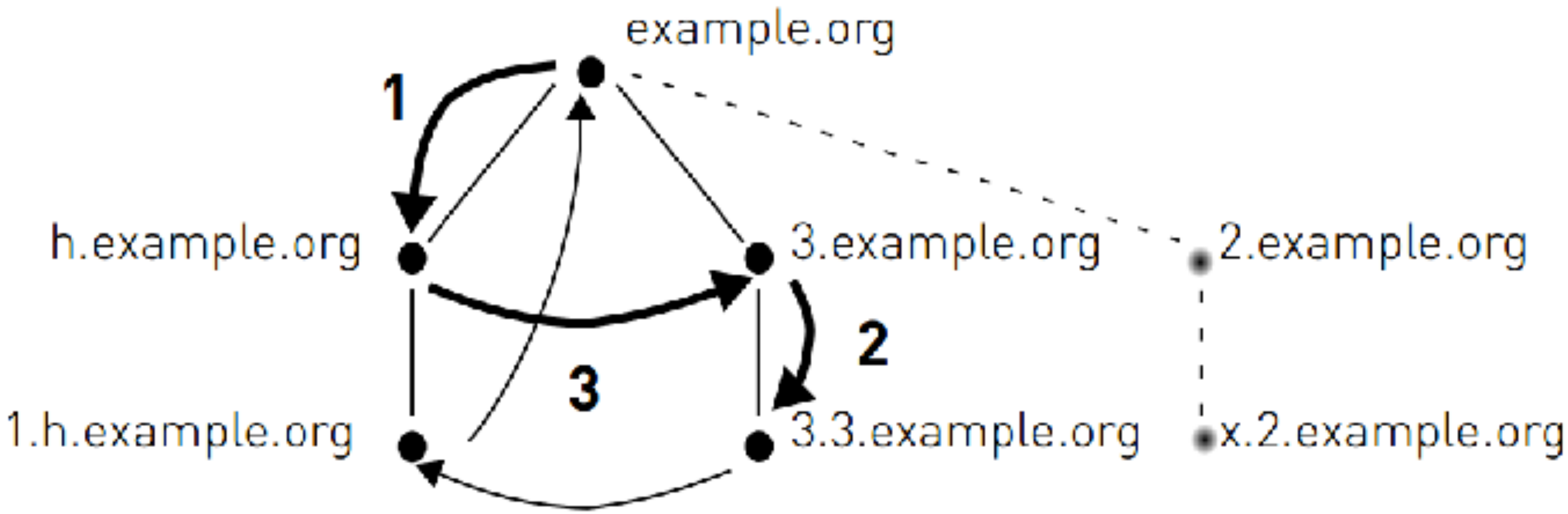
Authenticated DoE

- DNSSEC
  - Prevent forging or modifying DNS records
  - Allow the DNS to prove that a query answer **does not exist**
- Current version allows for zone enumeration

DNSSEC NSEC



DNSSEC NSEC3



NSEC5: Preventing Zone Enumeration

- Main result: public-key operations are necessary to prevent zone enumeration
- Idea: replace the hash in NSEC3 with a keyed hash

Signed Records

Secondary keys: ($PK_S = e$, $SK_S = d$)

For each record x,

$$S(x) = (h_1(x))^d \bmod N$$

$$F(x) = h_2(S(x))$$

For hash functions h_1 and h_2

Proving Non-Existence

On query q

- Compute S(q) and F(q)
- Respond with S(q) and all hashes after F(q)

To verify a response

1. Check that $(S(q))^e = h_1(q)$
2. Verify the response with PK_S
3. Check that $F(q) = h_2(S(q))$ is before all hashes

Comments on Confidentiality

- There's **a tremendous number** of papers on confidentiality in ICN
- All of them solve the problem at the application layer
- This problem is **important**, but not yet a core network service