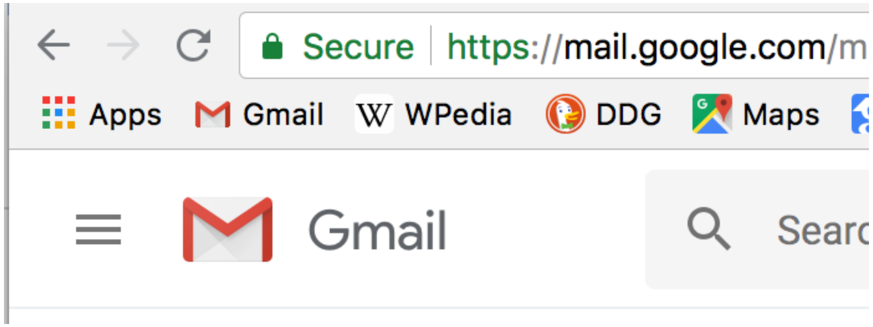
Secure Opportunistic Multipath Key Exchange (SMKEX)

Sergiu Costea, **Marios O. Choudary**, Doru Gucea,
Björn Tackmann and Costin Raiciu
University Politehnica of Bucharest, ETH Zürich, IBM Research Zürich

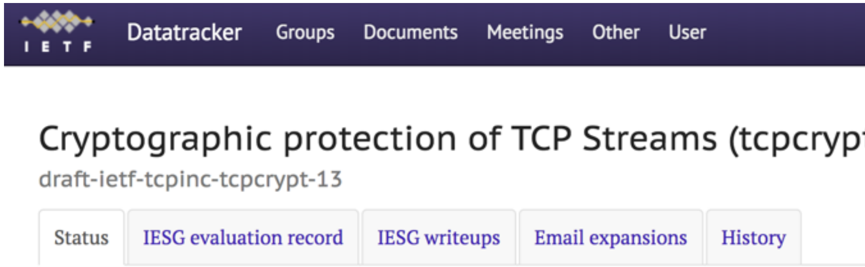
ACM CCS 2018, Toronto

Major shifts in the security of Internet communications

- HTTPS enabled by default on major websites



- Opportunistic encryption instead of no encryption at all



Cryptographic protection of TCP Streams (tcpcryp

draft-ietf-tcpinc-tcpcrypt-13

- TOFU-based secure messaging

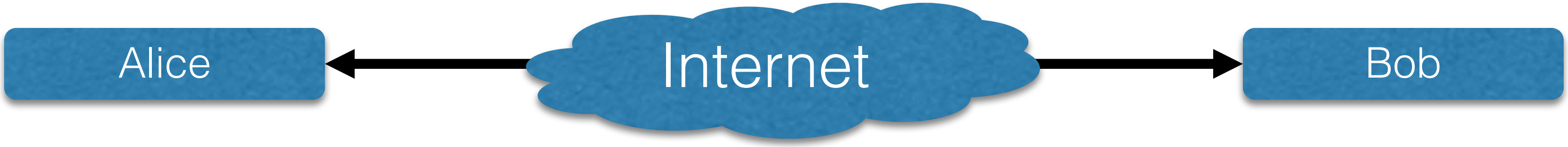
But there are still problems…

- HTTPS is broken if certificate authorities are compromised or fooled (remember previous talk)
- Opportunistic encryption only protects against passive attackers
- TOFU-based protocols are vulnerable against attackers on first connection

SMKEX can help now!

Main issue: difficult to authenticate a key exchange

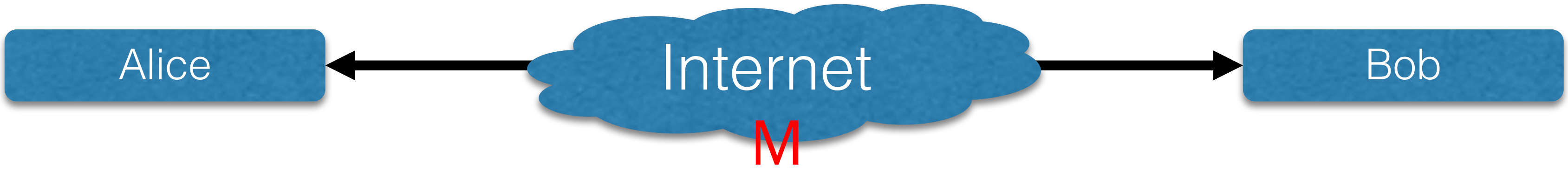
Ideal scenario:



K_{AB} DH key exchange K_{AB}

Main issue: difficult to authenticate a key exchange

Possible real scenario:



DH key exchange

DH key exchange

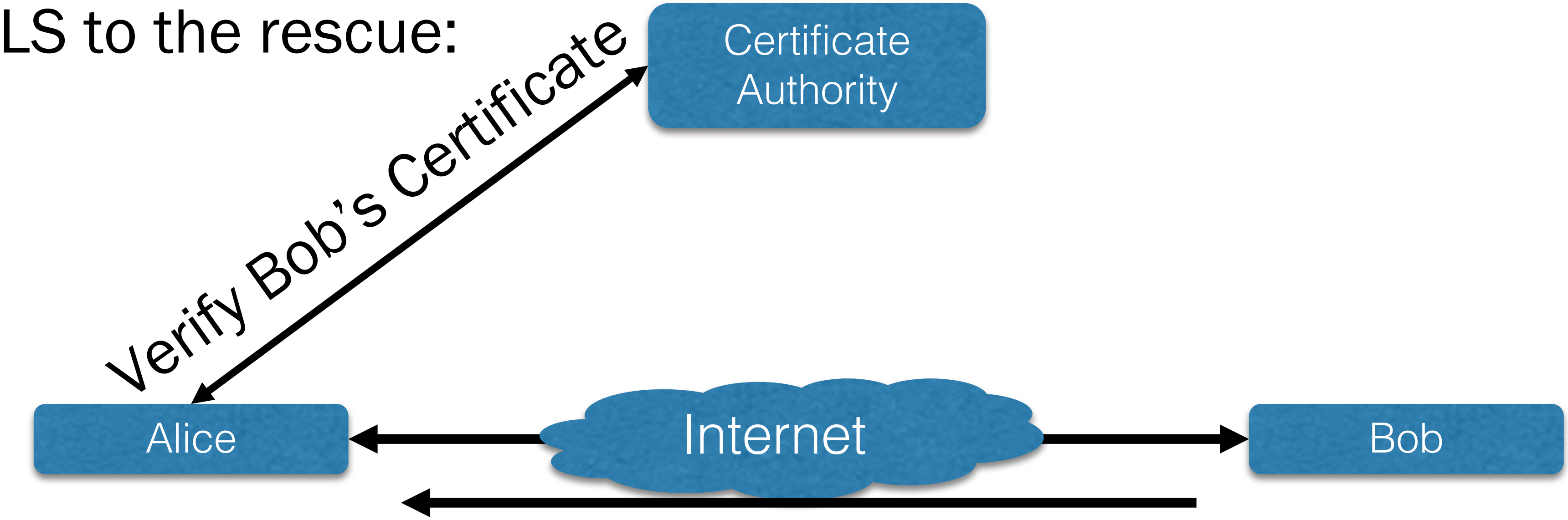
K_{AM}

K_{MB}

Problem for opp. encryption => need some aux channel for auth

Main issue: difficult to authenticate a key exchange

TLS to the rescue:

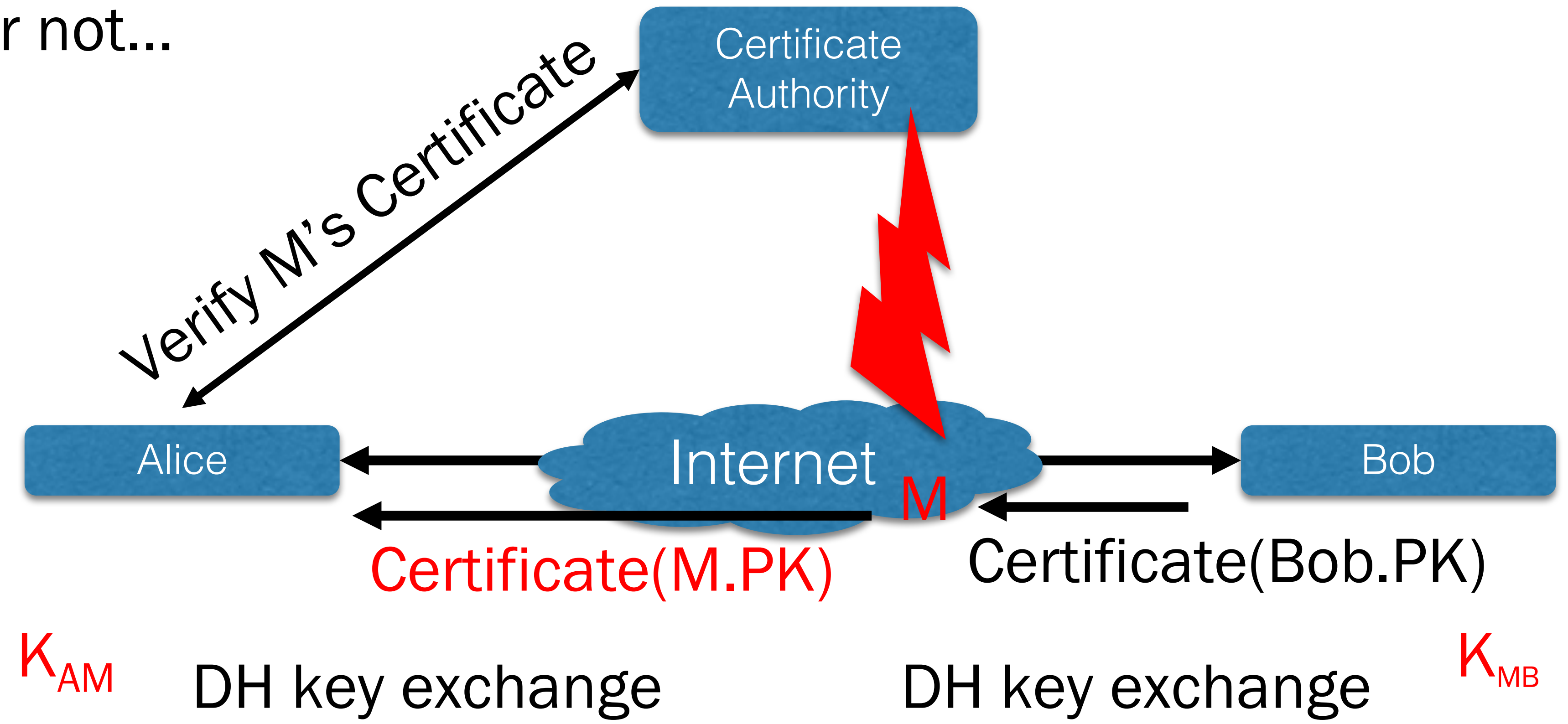


Certificate(Bob.PK)

K_{AB} DH key exchange authenticated by Bob.PK K_{AB}

Main issue: difficult to authenticate a key exchange

Or not…



Certificate Authority

Verify M's Certificate

Alice

Internet

M

Bob

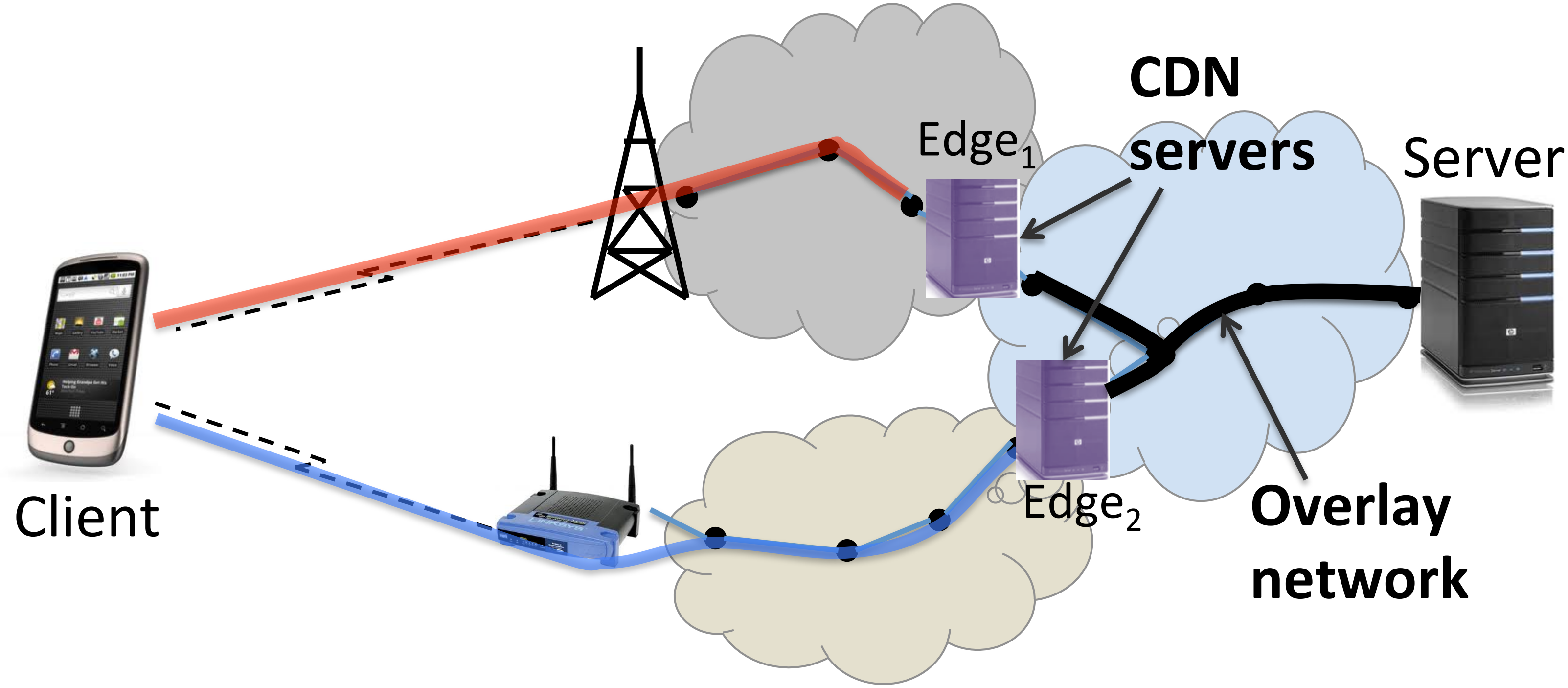
Certificate(M.PK)

Certificate(Bob.PK)

K_{AM} DH key exchange

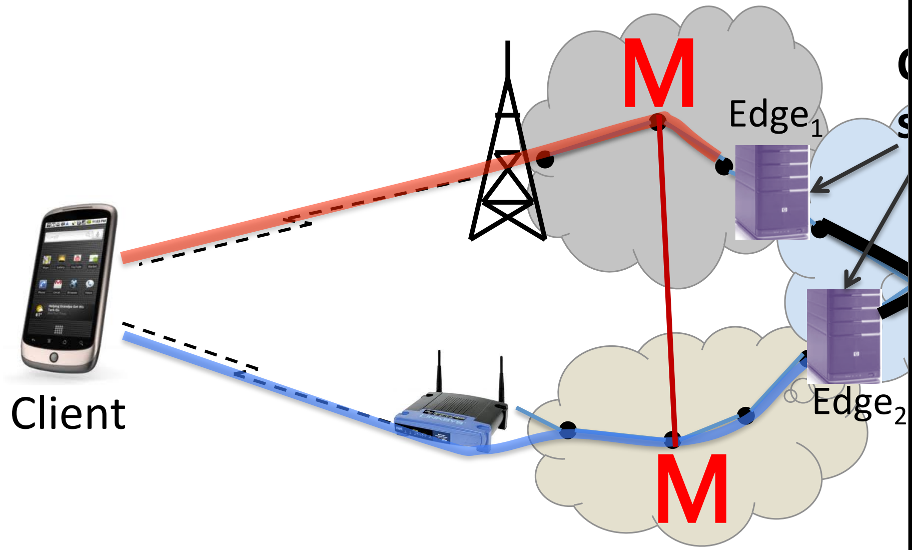
DH key exchange K_{MB}

Our **SMKEX** idea: use **multiple public channels** to **improve security**



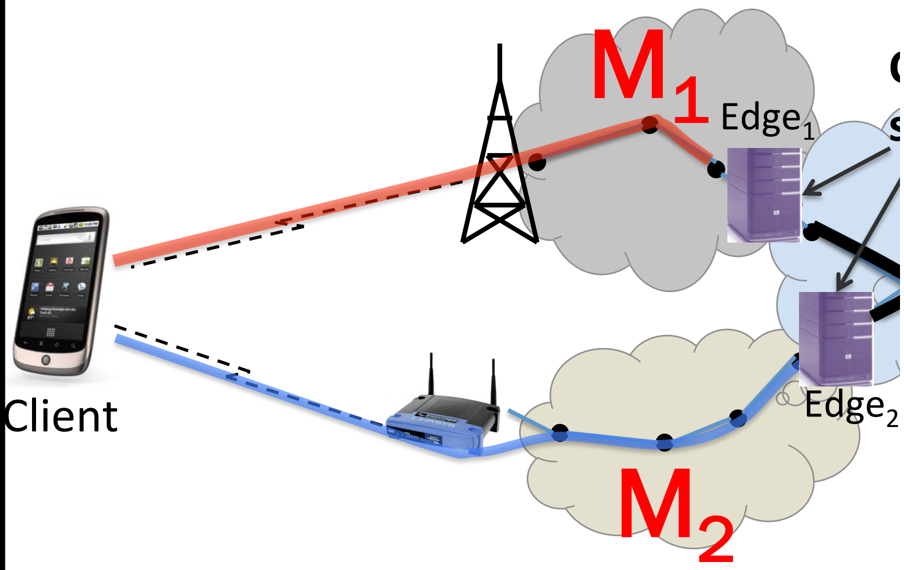
Attacker model in SMKEX

A-A



- Active attacker on both paths
- Synchronised across both paths

A/A

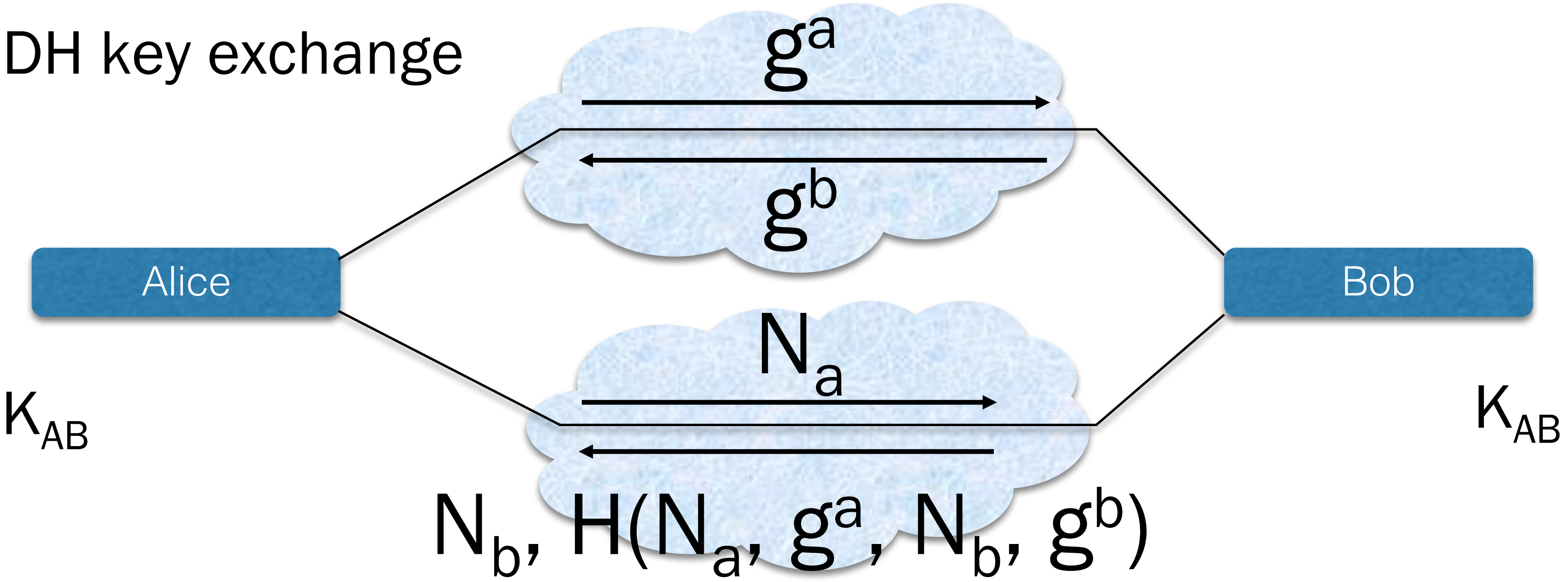


- Active attacker on both paths
- Unsynchronised across paths

SMKEX protocol

Path 1:

DH key exchange



Path 2:

Nonces + Hash of public keys and nonces

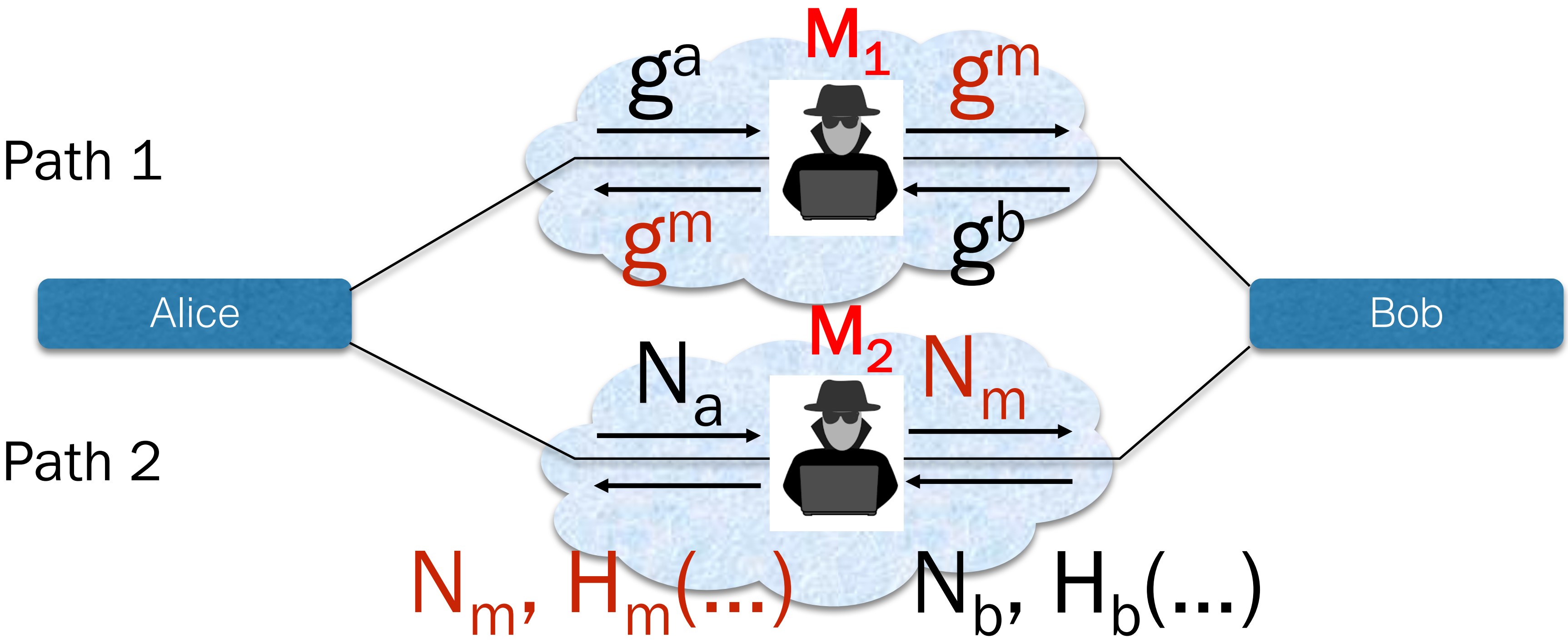
No CAs or pre-established keys used

SMKEX protocol

- Secure Key Exchange against A/A adversaries:

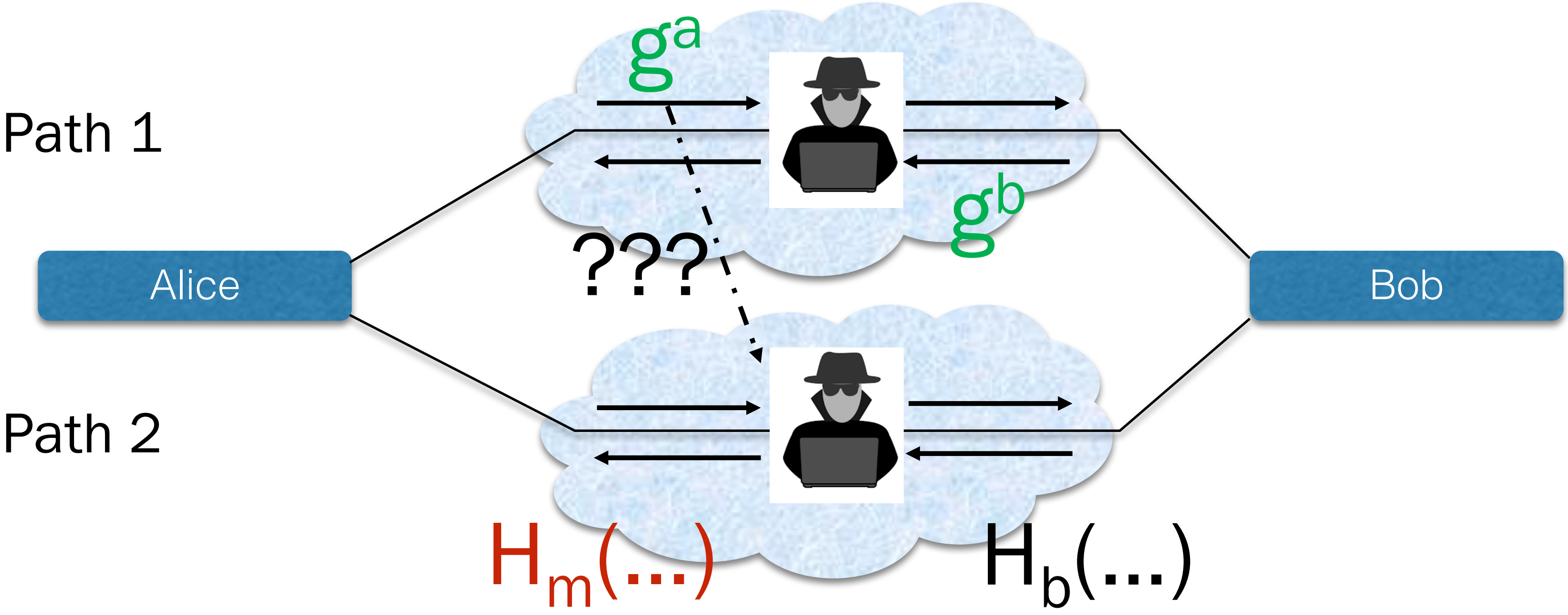
SMKEX protocol

- Secure Key Exchange against A/A adversaries:
  -> Intuition: adversary cannot change public values and send a correct hash at the same time



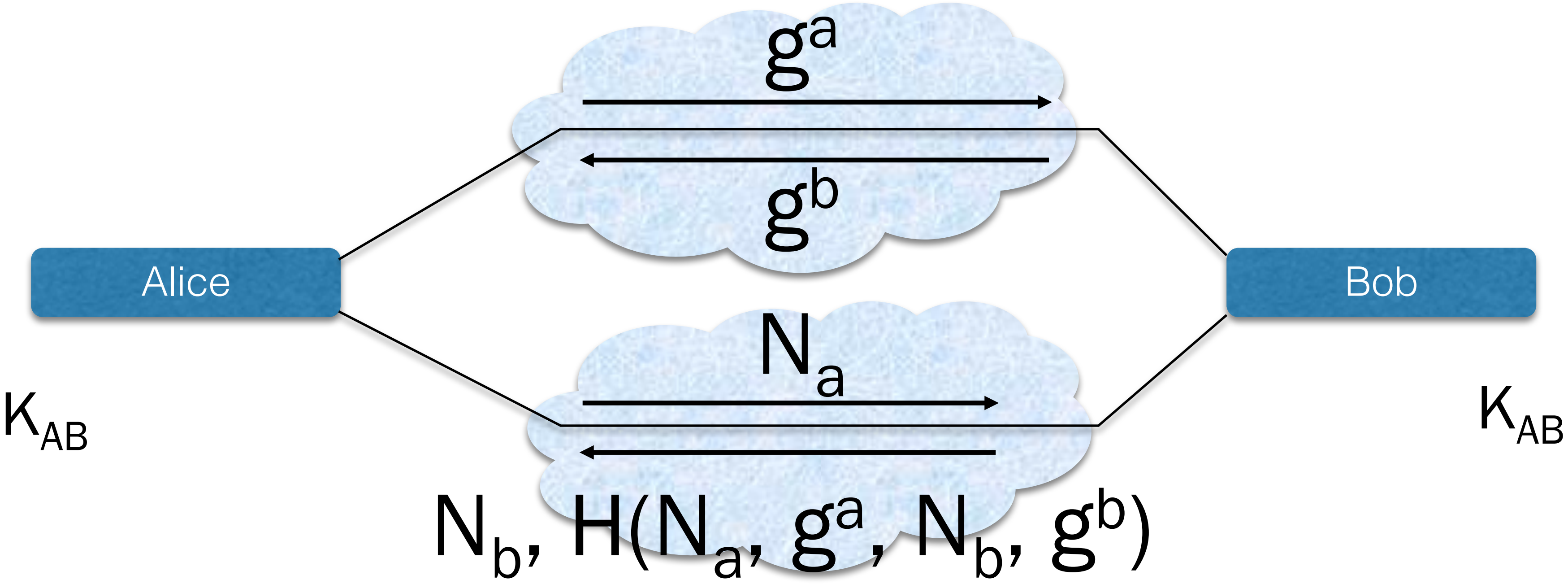
SMKEX protocol

- Secure Key Exchange against A/A adversaries:
  -> M_2 cannot generate correct H_m, as it lacks public key values



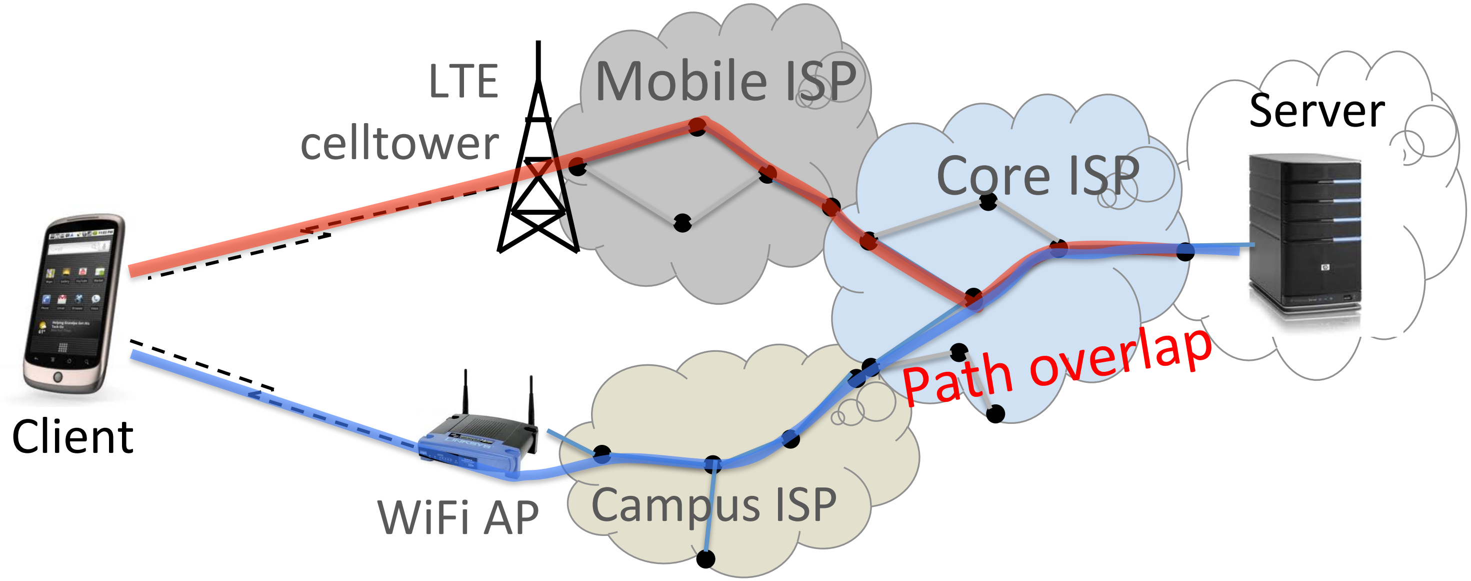
SMKEX protocol

- Forward and Backward secrecy across sessions:
  -> No long term secrets involved



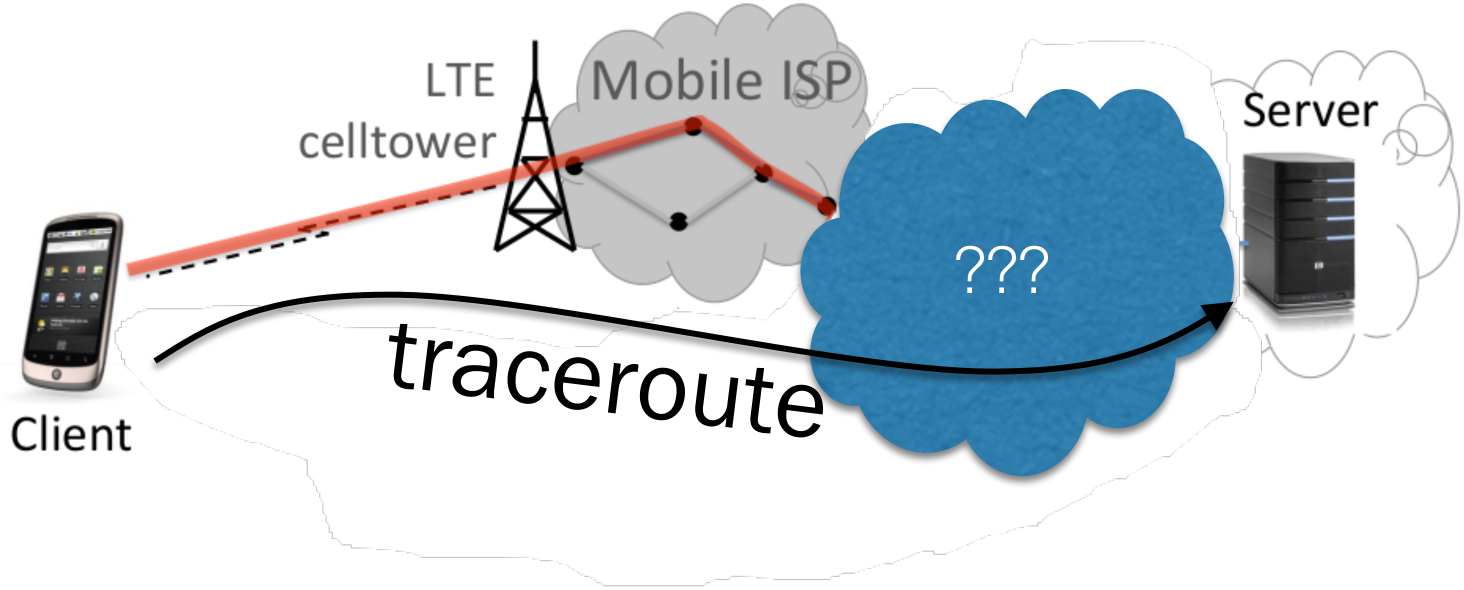
A/A scenario is realistic

- Measurement study:
  -> We measured path overlap between intermediary ASes from 5 countries to top 100 Alexa websites for combinations of fixed and mobile operators:



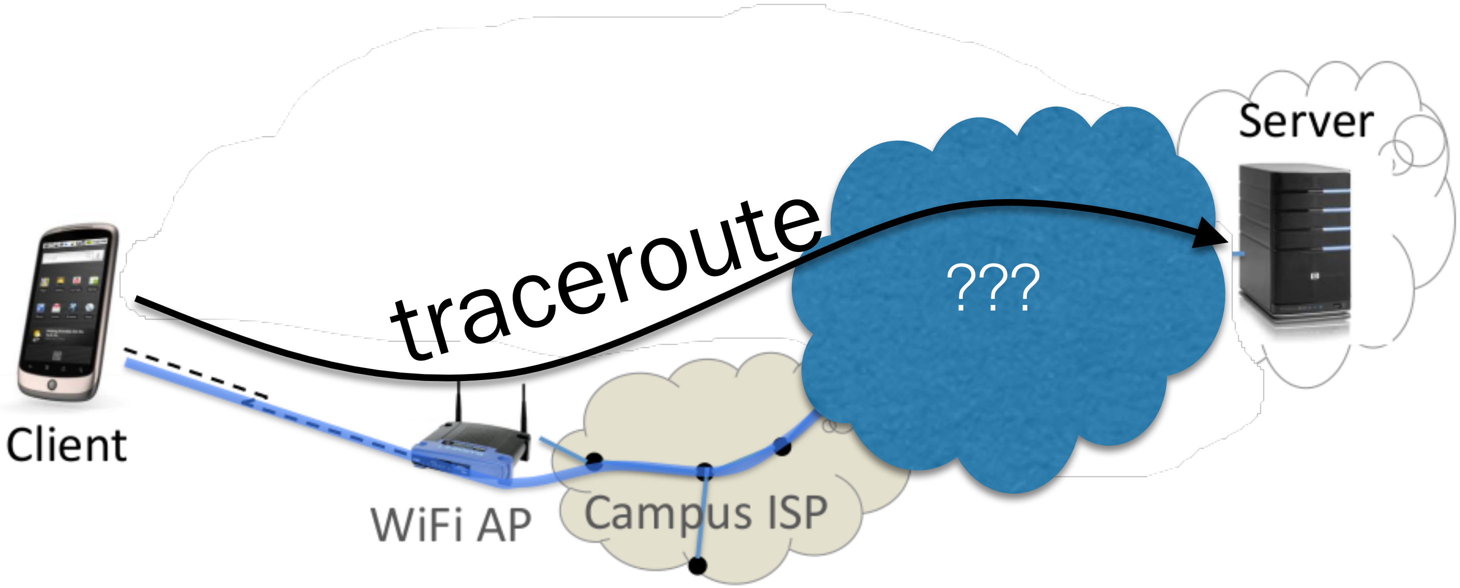
A/A scenario is realistic

- Measurement study:
  1) Run traceroute on mobile network to get list of intermediary AS-es from Client to Server
  => Set-mobile = {AS_M1, AS_M2, ...}



A/A scenario is realistic

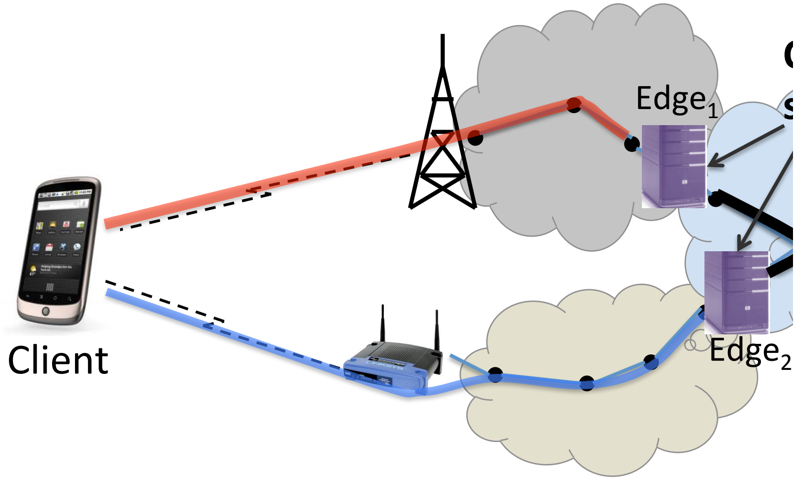
- Measurement study:
  2) Run traceroute on Wi-Fi/fixed network to get list of intermediary AS-es from Client to Server
  => Set-fixed = {AS_F1, AS_F2, …}



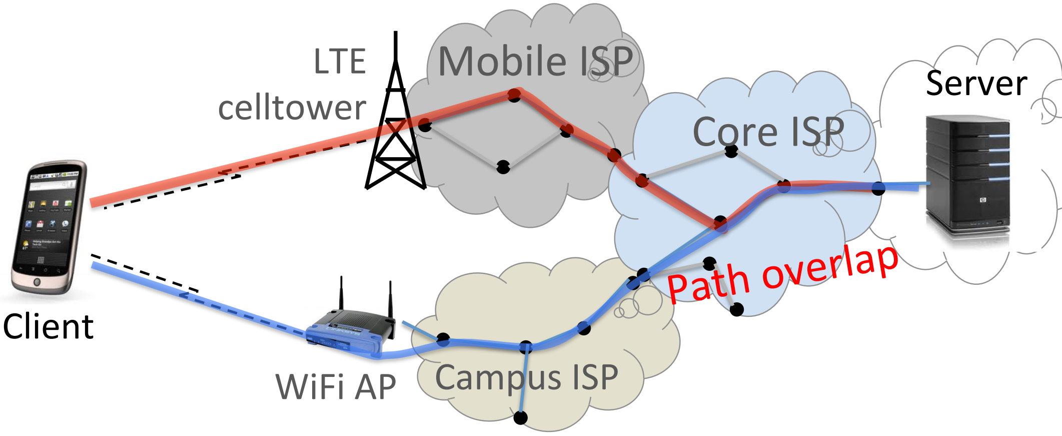
A/A scenario is realistic

- Measurement study:
  3) Get set of overlapping AS-es across both paths
  => Set-overlap = Set-mobile ∩ Set-fixed

No overlap



Path overlap



A/A scenario is realistic

- Measurement study:
  -> Result: NO path overlap for 50 to 70 top Alexa sites when choosing best mobile-fixed operator
  -> For worst combination, we still have 12 to 60 websites reachable with no path overlap

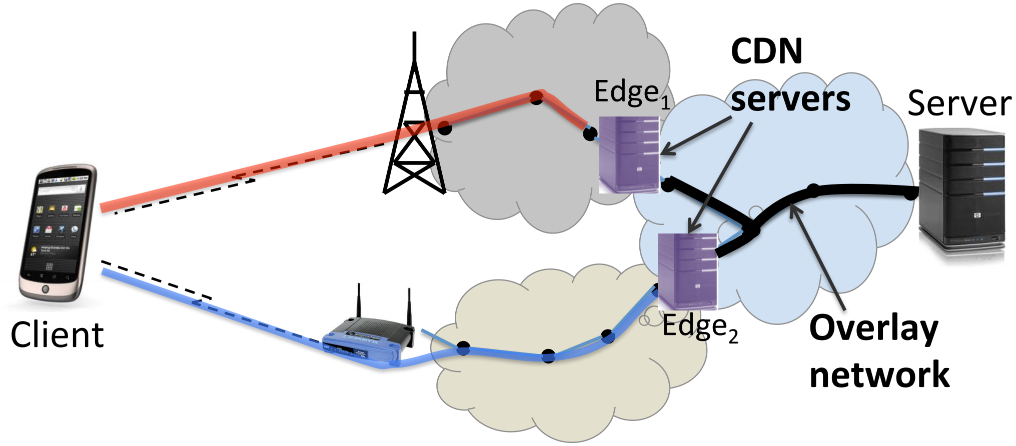
Country	Min #sites	Max #sites
USA	16	70
UK	48	54
Romania	12	50

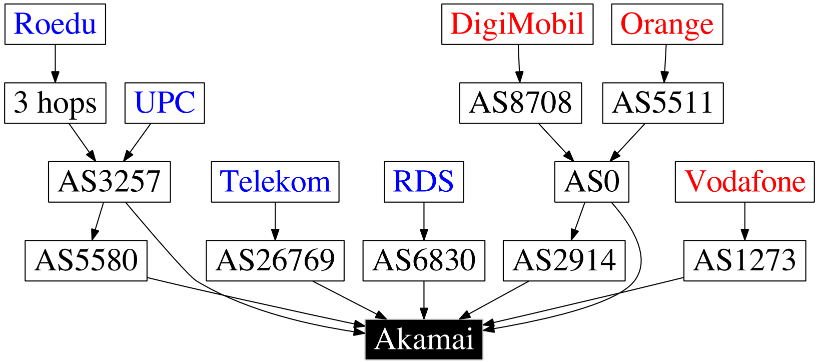
(number of sites reached with no path overlap)

=> SMKEX can already increase security against **local attackers** for many popular sites

A/A scenario is realistic

- Measurement study:
  -> No overlap also when using Akamai-hosted websites



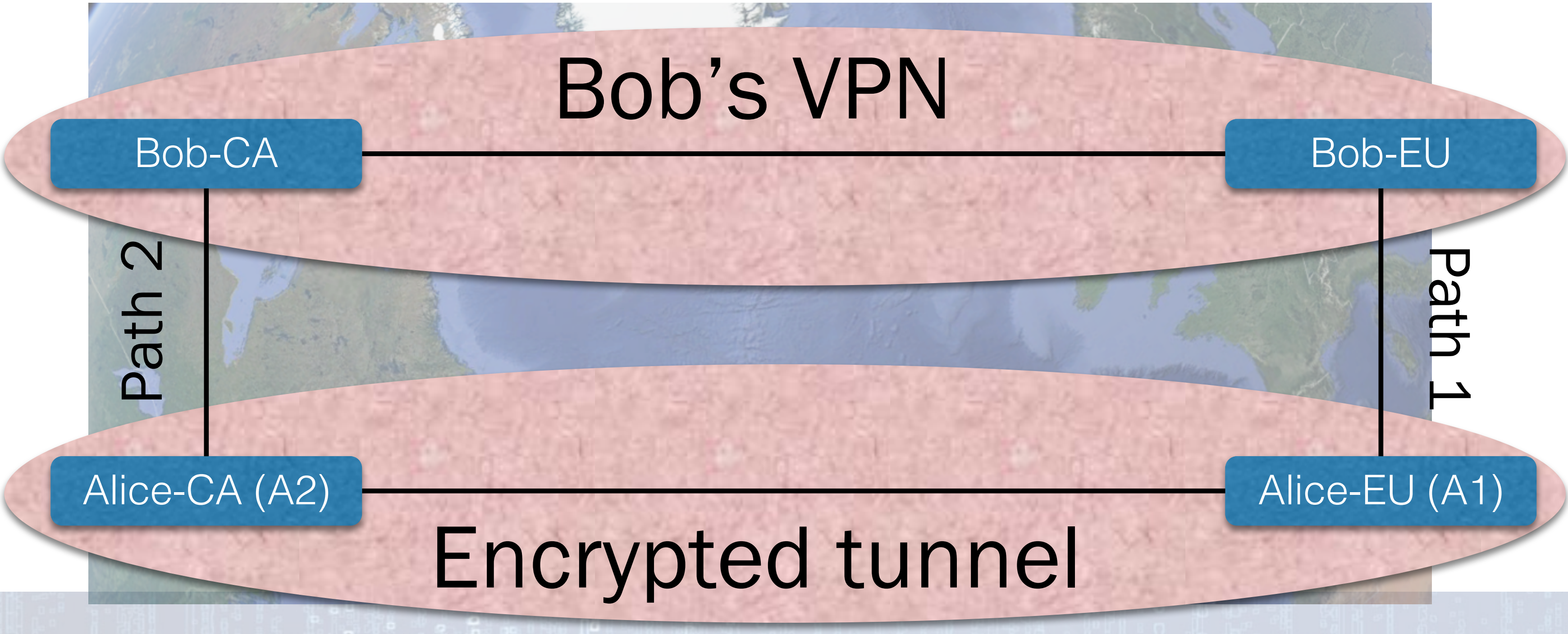


Example for Romania, which had worst path diversity in our study:
- Any combination of fixed-mobile has zero path overlap from client to edge server

=> SMKEX can also increase security for sites using a CDN such as Akamai

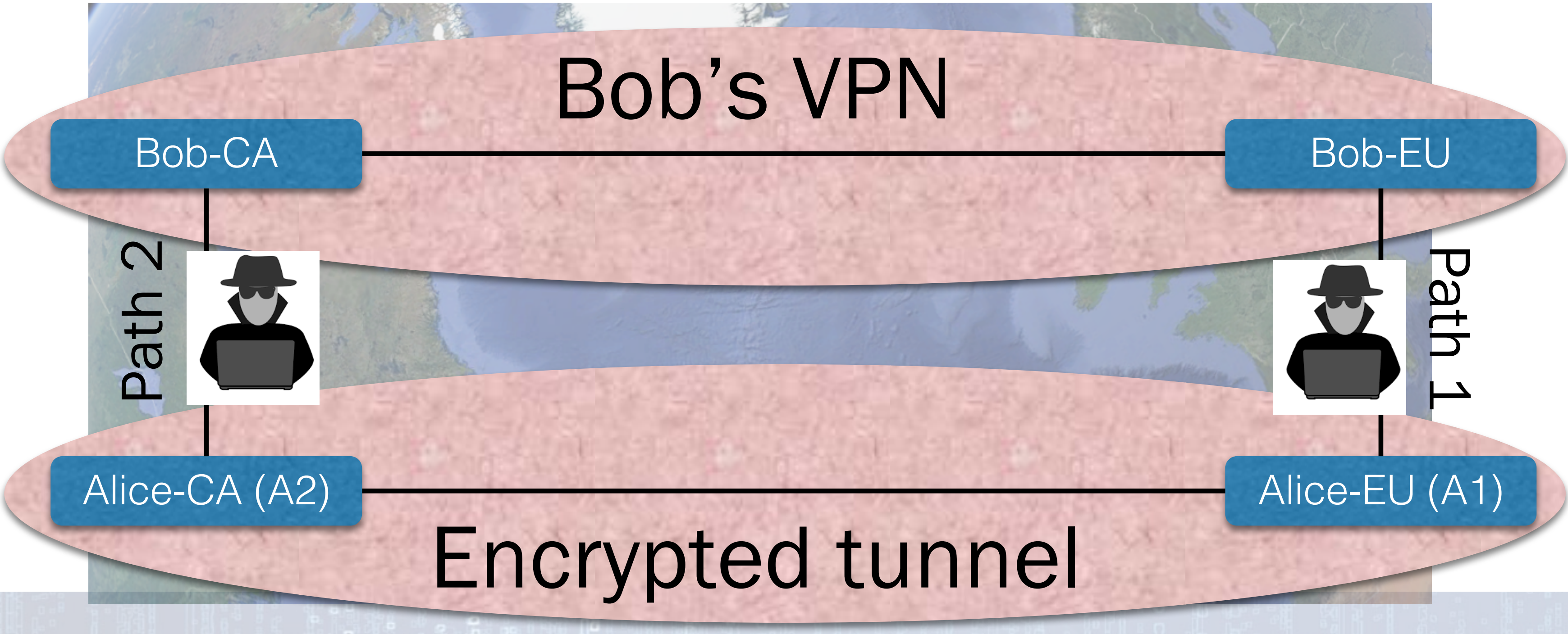
Enforcing A/A via long-term tunnels

- Main idea: use long-term encrypted tunnel
  -> set up only once to access any website/service



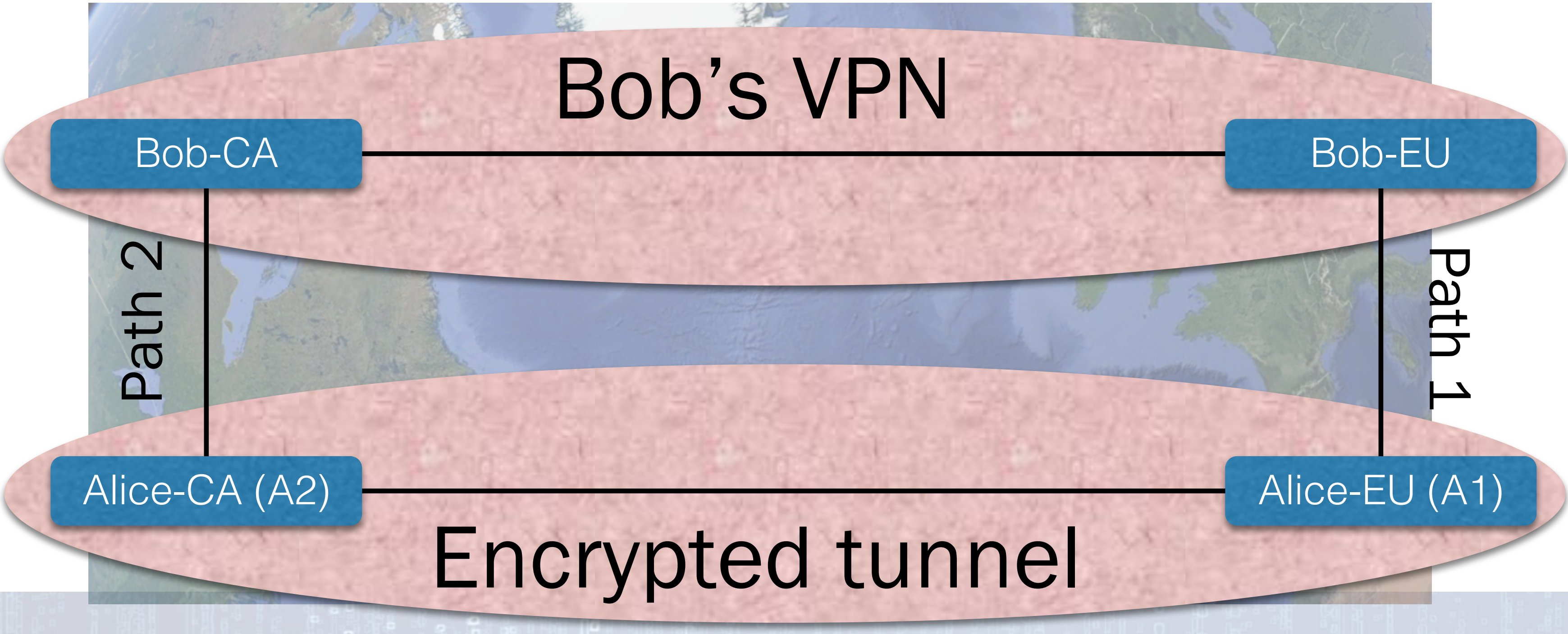
Enforcing A/A via long-term tunnels

- Main idea: use long-term encrypted tunnel
- Want to enforce A/A through non-overlapping paths



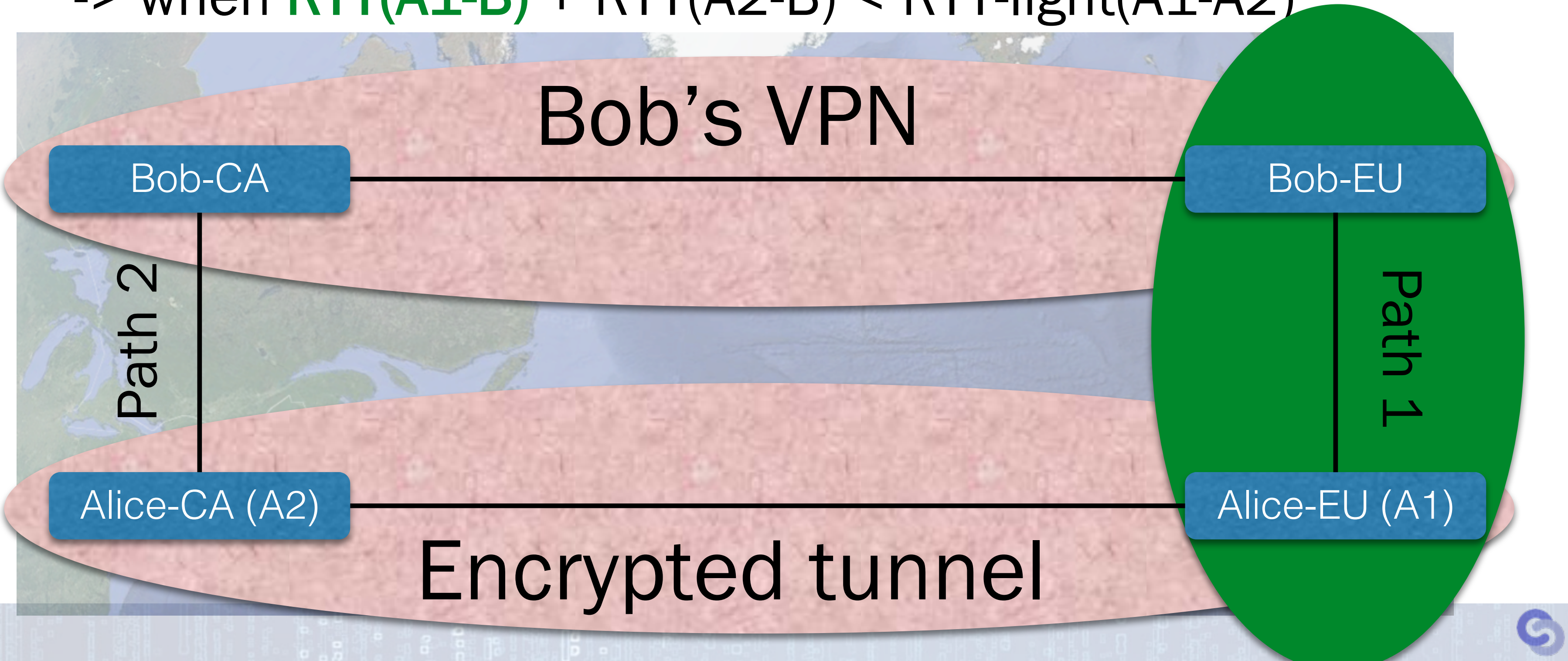
Enforcing A/A via long-term tunnels

- Main idea: use long-term encrypted tunnel
- Alibi routing can be used to guarantee disjoint paths:
  -> when RTT(A1-B) + RTT(A2-B) < RTT-light(A1-A2)



Enforcing A/A via long-term tunnels

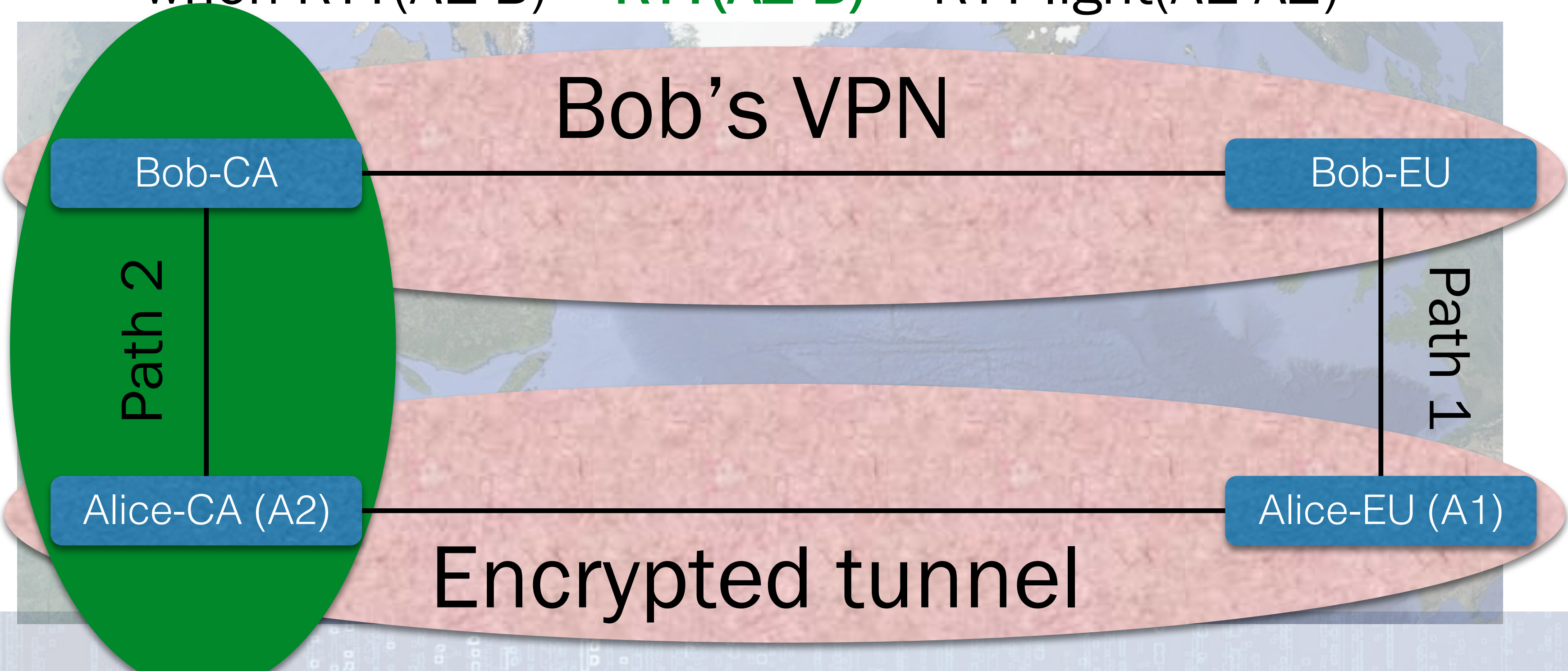
- Main idea: use long-term encrypted tunnel
- Alibi routing can be used to guarantee disjoint paths:
  -> when RTT(A1-B) + RTT(A2-B) < RTT-light(A1-A2)



Enforcing A/A via long-term tunnels

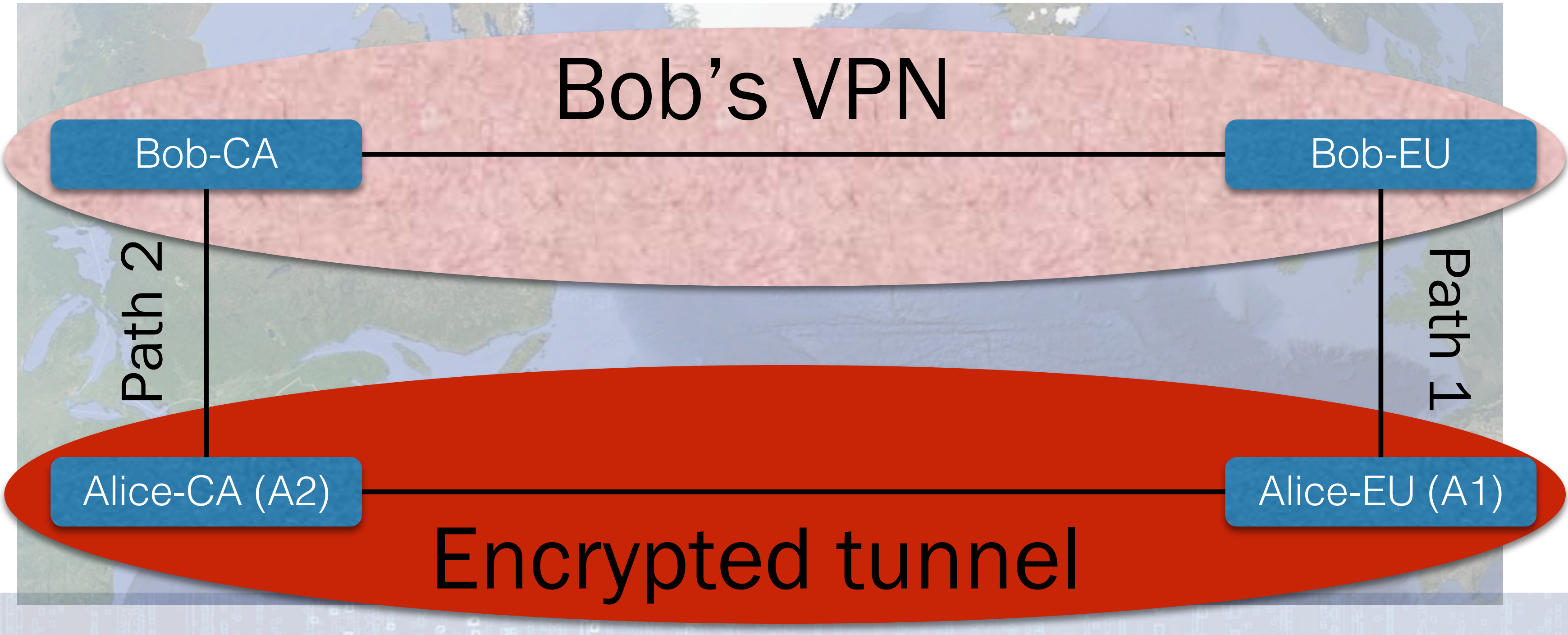
- Main idea: use long-term encrypted tunnel

- Alibi routing can be used to guarantee disjoint paths:
  -> when RTT(A1-B) + RTT(A2-B) < RTT-light(A1-A2)



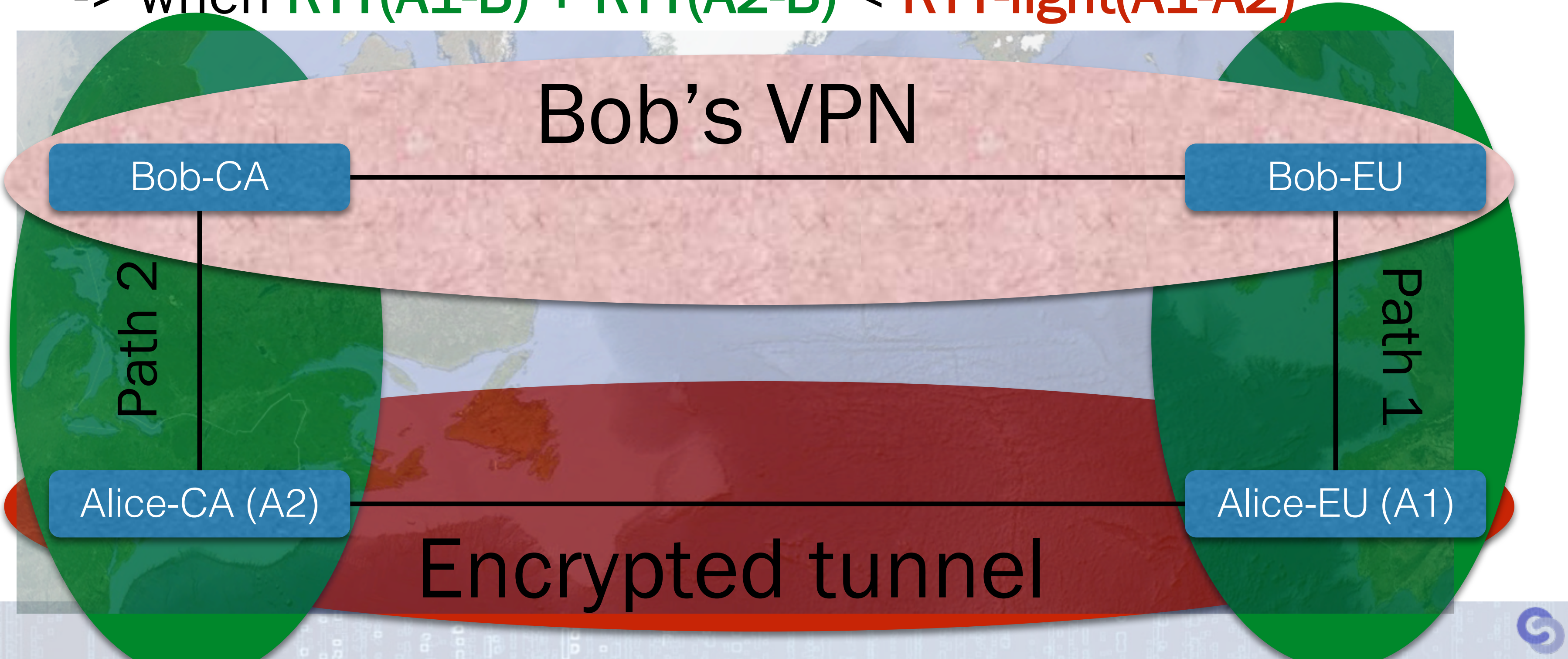
Enforcing A/A via long-term tunnels

- Main idea: use long-term encrypted tunnel
- Alibi routing can be used to guarantee disjoint paths: -> when RTT(A1-B) + RTT(A2-B) < RTT-light(A1-A2)



Enforcing A/A via long-term tunnels

- Main idea: use long-term encrypted tunnel
- Alibi routing can be used to guarantee disjoint paths:
  -> when RTT(A1-B) + RTT(A2-B) < RTT-light(A1-A2)



Enforcing A/A via long-term tunnels

- Main idea: use long-term encrypted tunnel

- Alibi routing can be used to guarantee disjoint paths:
  -> when RTT(A1-B) + RTT(A2-B) < RTT-light(A1-A2)

UK			**Switzerland**			**Romania**		
Thresh	Mobile	Fixed	Thresh	Mobile	Fixed	Thresh	Mobile	Fixed
60ms	3%	48%	68ms	0-50%	50-66%	82ms	34-38%	47-65%

Fraction of sites (in Alexa top 100) with guaranteed non-overlapping paths when using a tunnel from EU to a VM in US (Virginia)

=> SMKEX can also protect against **nation-wide attackers** when using a tunnel

Integration of SMKEX protocol

TOFU-based SMKEX

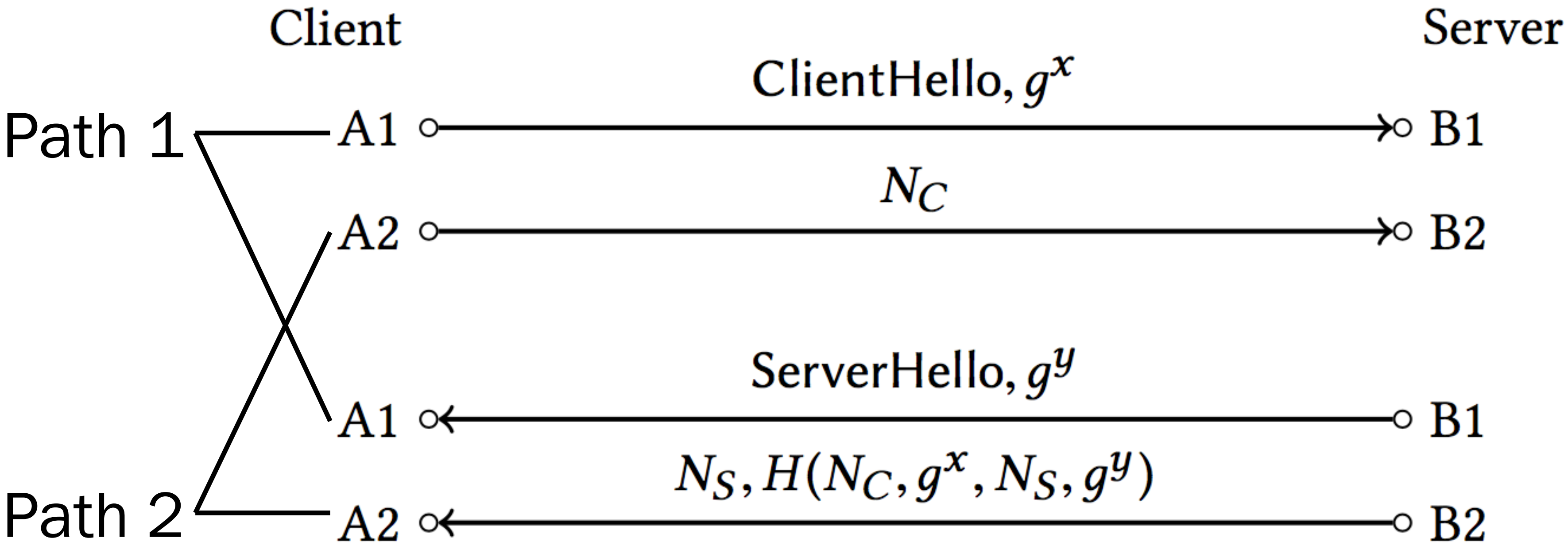
- Increase security of SMKEX while maintaining usability
  - E.g.: client stores long-term public key of server for verification in subsequent connections
  - Increases security of first connection and key update (weak point of TOFU protocols, e.g. SSH, WhatsApp)

SMKEX integration into TLS/QUIC

- Combine best of both protocols:
  - TLS/QUIC:
    -> enforced identification via certificates
    -> widely deployed
  - SMKEX: protection against A/A attackers:
    -> can detect local and national attackers using rogue certificates
  - On-going work for standardisation

SMKEX integration into TLS/QUIC

- Combine best of both protocols:

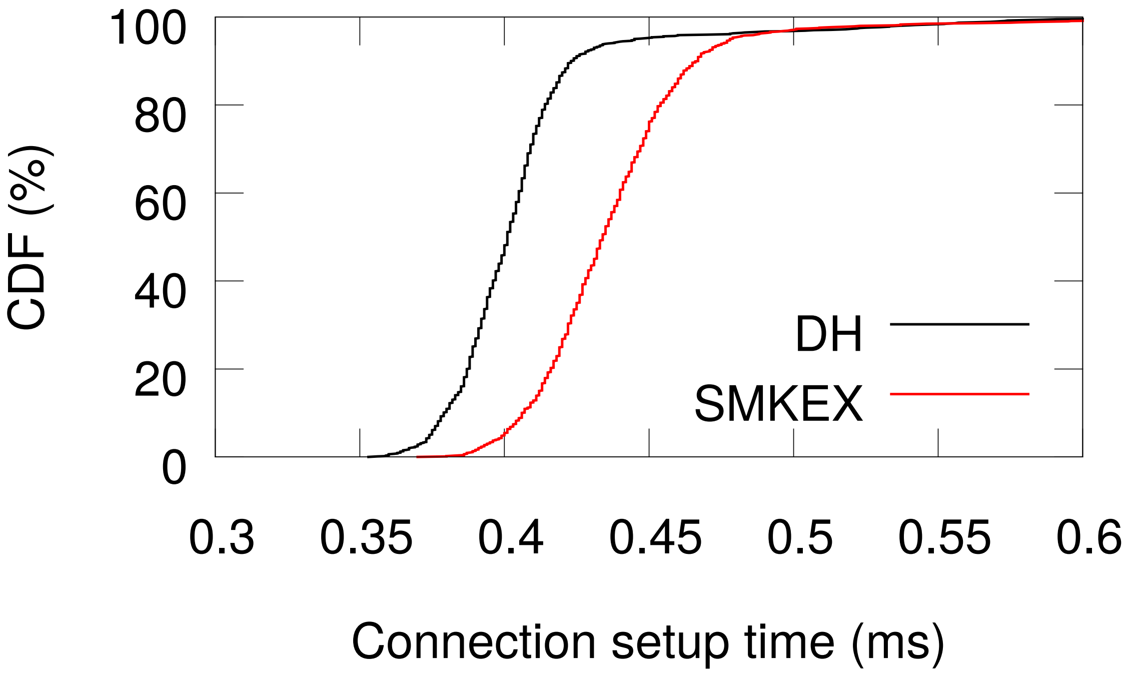


SMKEX implementation

- Two core components:
  - User-space library that overwrites SOCKET API
    -> can be (LD)preloaded before running client/server to allow unmodified server/client application to use SMKEX
  - MPTCP-enabled kernel (with some patches)
    -> to allow the seamless creation/management of multiple public channels without any modifications to the application

SMKEX implementation

- Minor overhead over standard DH key exchange (1 RTT):
  -> Experiments on two quad-core Xeon machines connected with two Gigabit links



Conclusions

- SMKEX increases the security of opportunistic encryption
  -> Protection against local and national active A/A attackers without trusted 3rd parties or pre-established keys

- SMKEX can also increase the security of TOFU and TLS
  -> TLS/QUIC-SMKEX provides best of both protocols

- Existing infrastructure already enables SMKEX **NOW!**
  -> We can even enforce path diversity via long-term tunnels

- Code is open source https://github.com/nets-cs-pub-ro/smkex https://github.com/nets-cs-pub-ro/mptcp-smkex

- See paper for many more details: implementation, implementation using CDNs, security proofs.

SMKEX Team

POLITEHNICA UNIVERSITATEA DIN BUCURESTI 1818

Computer Science
& Engineering
Department

Marios.choudary@cs.pub.ro
Costin.raiciu@cs.pub.ro
bta@zurich.ibm.com
Sergiu.costea@cs.pub.ro
gucea.doru@gmail.com

IBM Research | Zurich

ETH zürich