Fully Non-Interactive Onion Routing with Forward-Secrecy

D. Catalano[1], M. Di Raimondo[1], D. Fiore[2], R.Gennaro[3] and O. Puglisi[1]

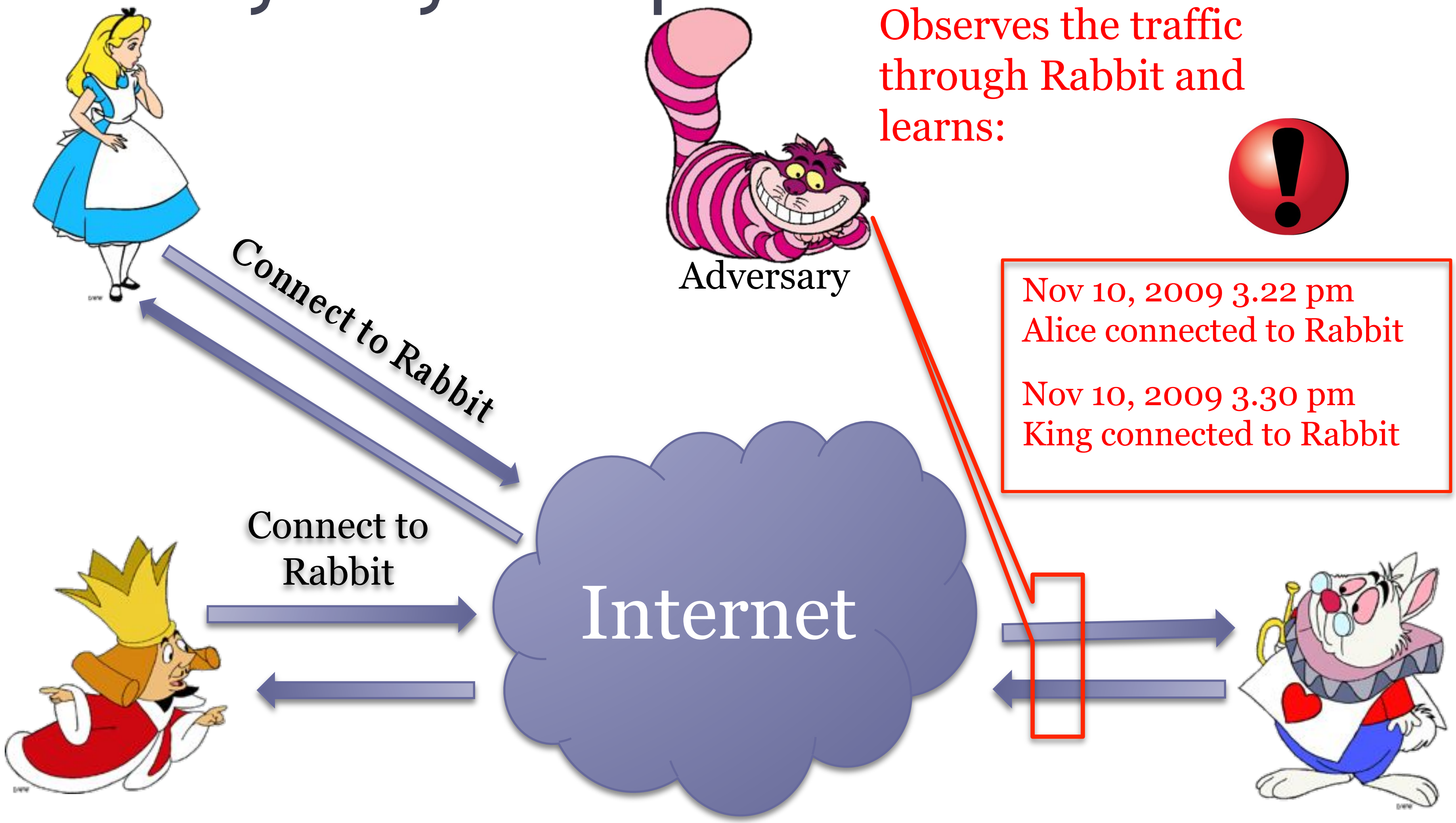
[1]Università di Catania, Italy
[2]ENS, France
[3]IBM Research, USA

ACNS 2011 – Nerja (Malaga), Spain

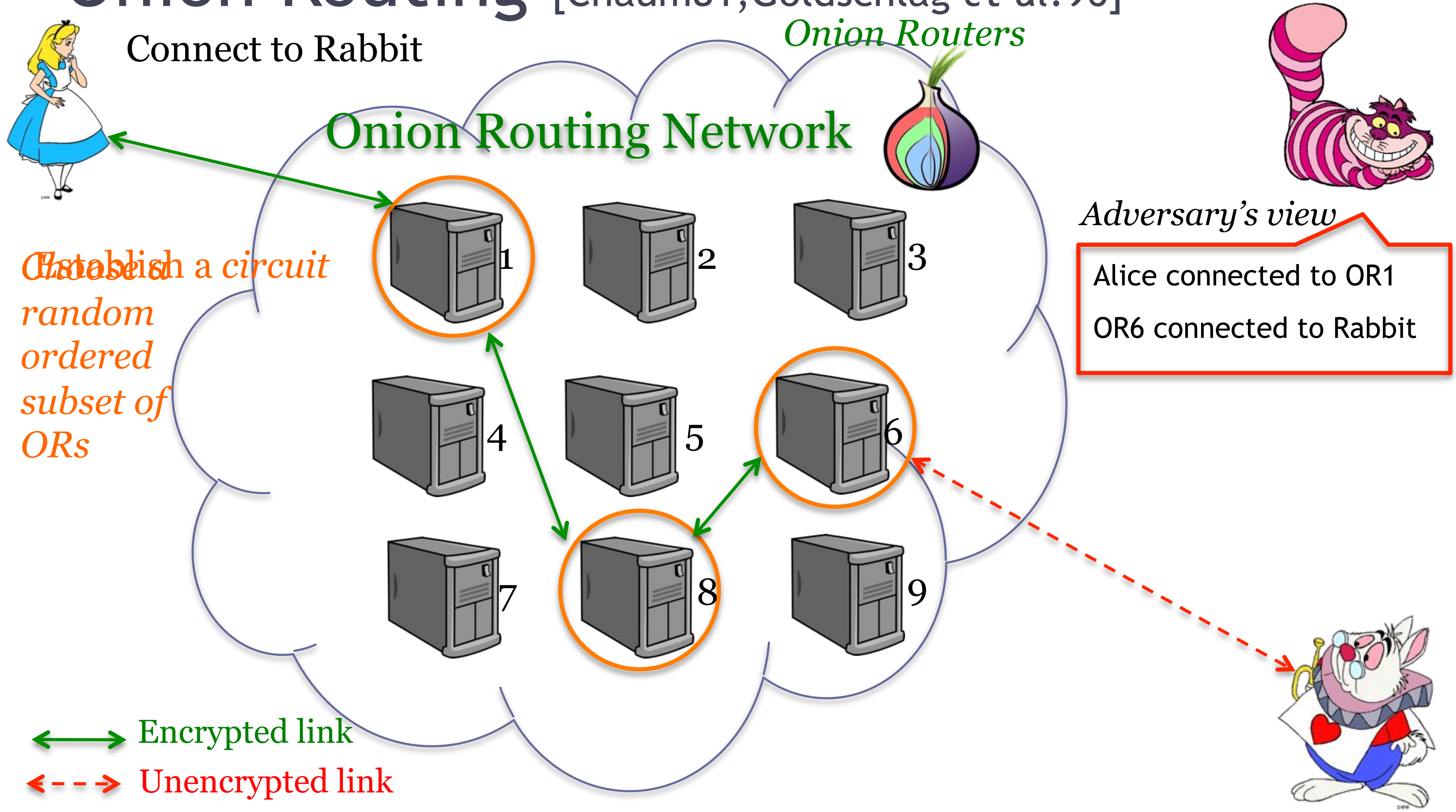
Outline

- Anonymity in a public network
- Onion Routing
  - Security properties
  - Previous work
- Forward-Secure Onion Routing
  - Our solution
- Comparisons

Anonymity in a public network

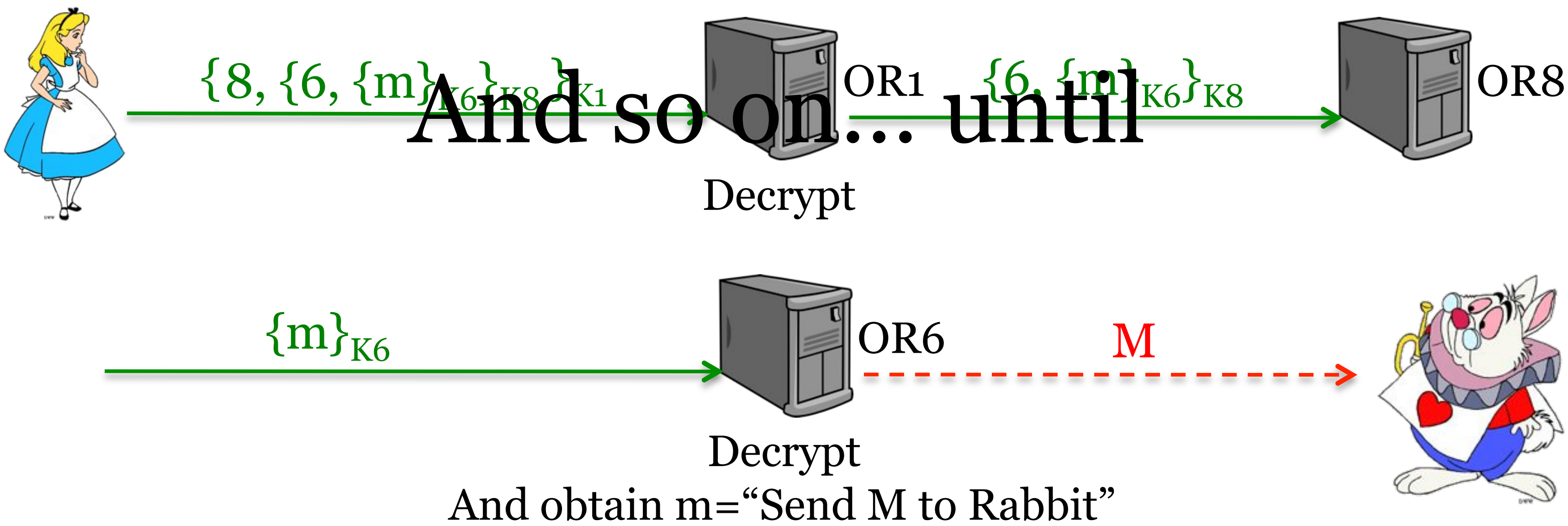


Onion Routing [Chaum81,Goldschlag *et al.*96]



Onion Routing

1. Alice establishes a session key with each Onion Router
   - K1 with OR1, K6 with OR6, K8 with OR8
2. Alice creates an "onion" ciphertext $\{8, \{6, \{m\}_{K6}\}_{K8}\}_{K1}$ and sends it to OR1



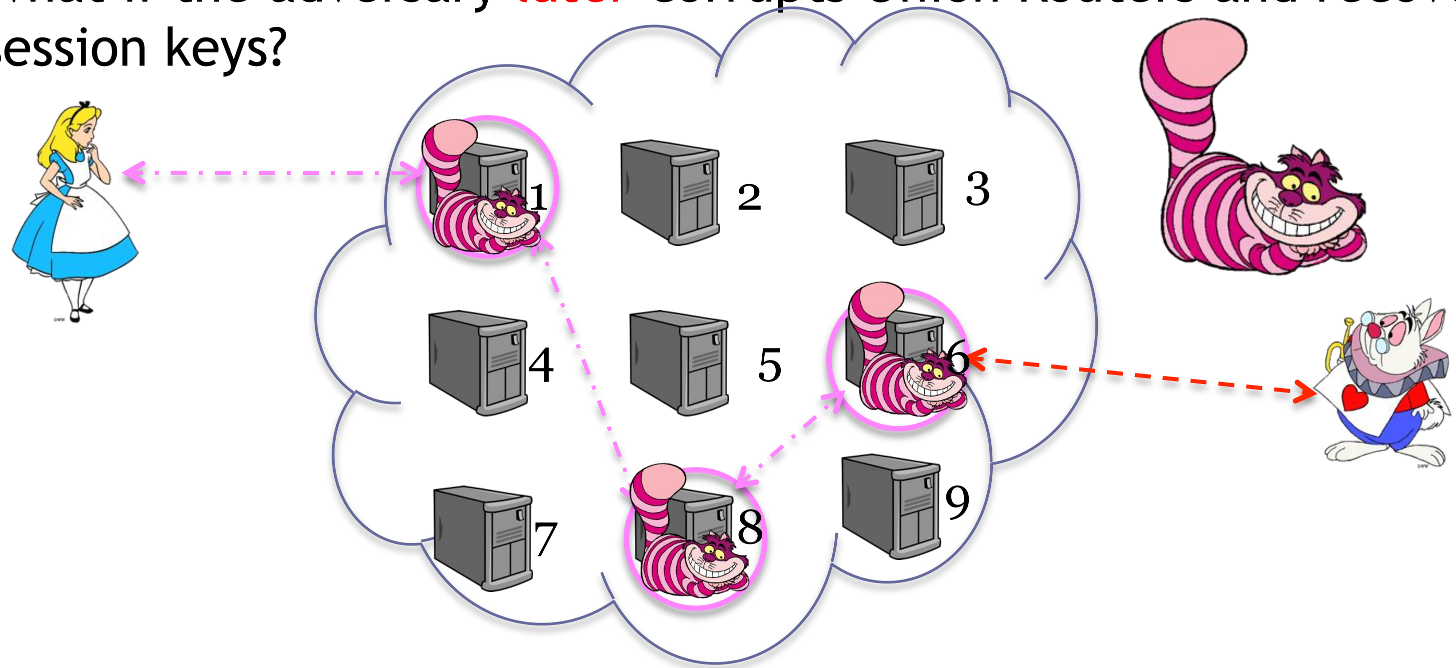
Why does OR achieve anonymity?

- Encrypted links hide the circuit
- The adversary cannot have a complete view of the entire network
- → it is infeasible to link Alice and the Rabbit!

- **How to establish session keys?**
  - This can be considered the main technical problem of each OR protocol
  - We focus on this part

Forward Secrecy

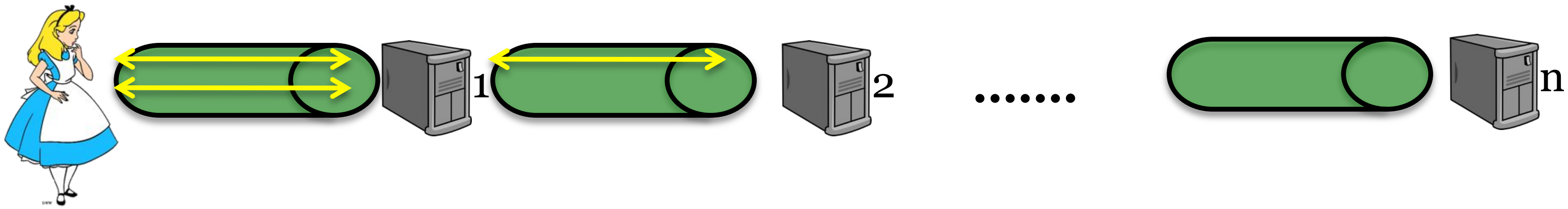
- First OR proposal [Goldschlag *et al*.96]:
  - pick a random session key *K*
  - send *K* encrypted with the recipient's public key
- What if the adversary *later* corrupts Onion Routers and recovers session keys?



- He would be able to learn the circuit and thus break anonymity of past communications!

Onion Routing Protocols

- Tor: The Second Generation Onion Routing Project
  - Active project that provides anonymity over Internet (currently with about 1000 onion routers and 100.000 users)
  - **First:** achieve forward secrecy by periodically changing public keys
    - Inefficient as it requires issuing new certificates and additional traffic
  - **Then:** Tor Authentication Protocol (TAP) using *telescoping* [Goldb.06]
- **Telescoping**



Choose OR1

Establish a secure channel with OR1 (via an RSA-encrypted Diffie-Hellman key-exchange)

Choose OR2

Use the channel with OR1 to establish another channel with OR2

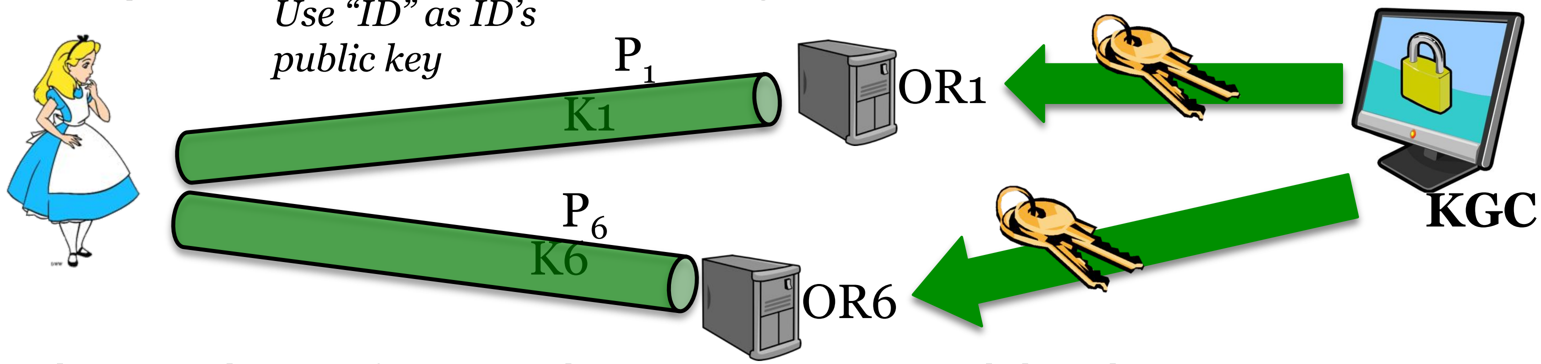
And so on until the last router in the circuit

TAP achieves forward secrecy using an interactive protocol.

Total cost = O(n²) exchanged messages

Pairing-Based Onion Routing [KGZ*07*]

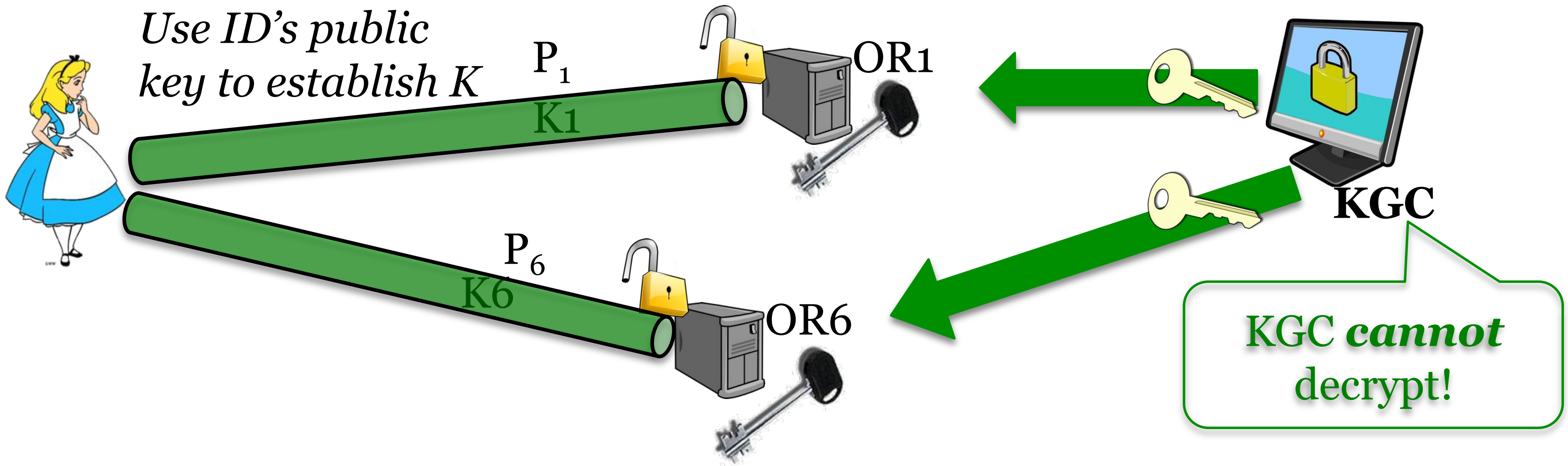
- Adopt the ID-based setting



- Alices doesn't need to get ORs public keys
  - The key-agreement is non-interactive
- In order to achieve forward secrecy:
  - KGC frequently changes master key *(e.g. every day)*
  - KGC frequently issues new private keys for onion routers *(e.g. every hour)*
- ☺ less traffic for users than in the PKI setting
- ☹ a lot of work for the KGC – interaction OR-KGC

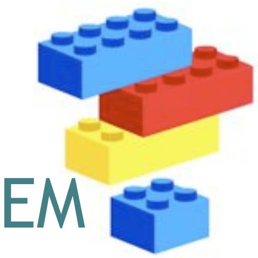
Certificateless Onion Routing [CFG09]

- Apply the idea of ***Certificateless Encryption*** to OR



- The key-agreement phase is non-interactive
- ☺ Routers update keys by themselves
- ☹ Alice has to get new PKs at every update

Our Result: a fully non-interactive solution

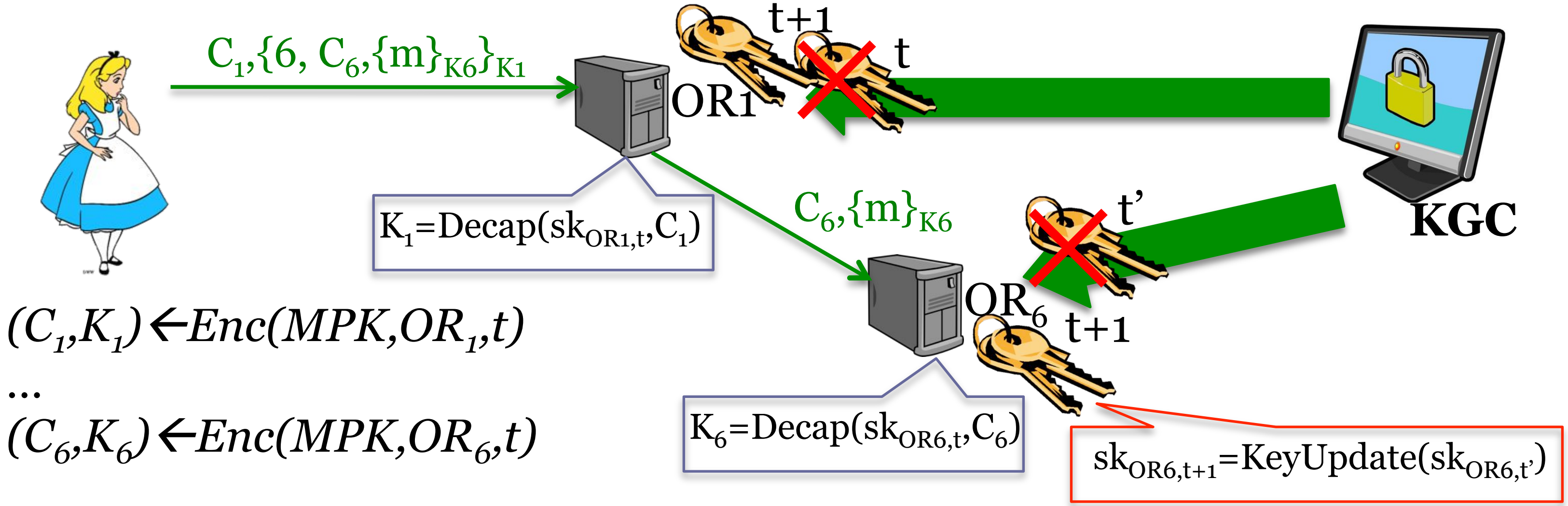


- **Our building blocks:**
  - CCA-secure Forward-Secure Identity-Based KEM
    - Extend FS-PKE [CHK03]
  - CCA-secure Symmetric Encryption

fs-IB-KEM:

- **Setup()**→ (MPK, MSK)
- **KeyGen(MSK,ID,t)**→$sk_{ID,t}$ //identity string ID, time t
- **KeyUpdate($sk_{ID,t}$)** → $sk_{ID,t+1}$
- **Encap(MPK, ID, t)** → (C, K)
- **Decap($sk_{ID,t}$,C)** → K

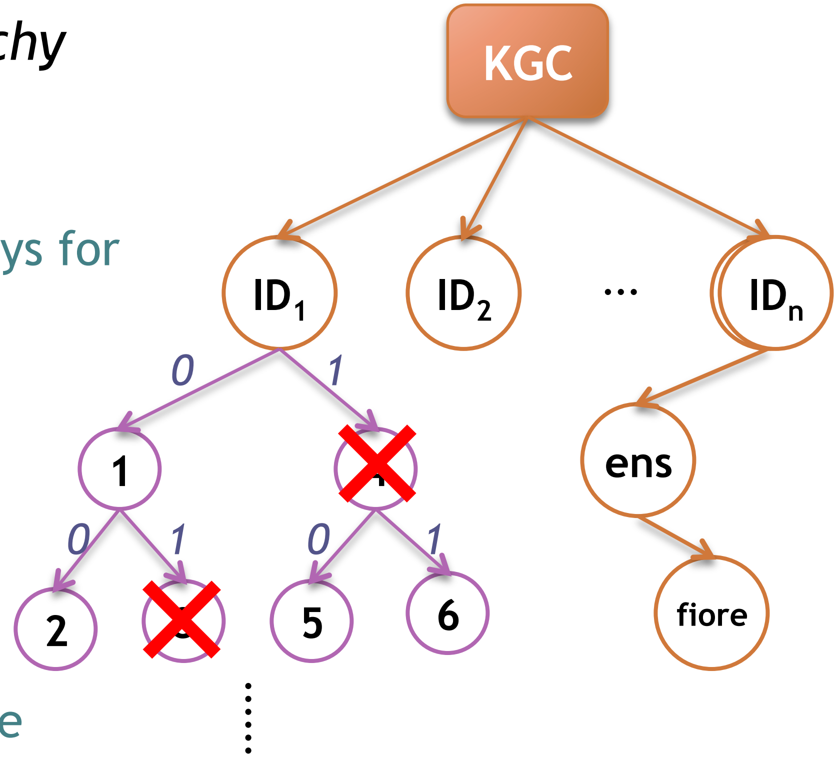
Forward-Secure Onion Routing



- Forward-Secrecy
  - Routers update keys by themselves
  - Alice uses always the same public key
- Formally prove security assuming CCA-secure fs-IB-KEM and CCA-secure SKE
  - *Fixed small flaw in [KGZ07] saying that a CPA SKE was sufficient*

A concrete construction of fs-IB-KEM

- *Extend [CHK03] to an hybrid hierarchy*
- **Basic Idea: use HIBE**
  - Users organized in a hierarchy
  - Each user can generate (delegate) keys for any of its descendants



- **fs-IB-KEM**
  - 1^{st} level: users
  - levels>=2: time periods
  - Encrypt(ID_1,3)=Encrypt(ID_1|01)
  - Keys associated with nodes in the tree
  - At time 3, ID1 has $sk_{ID,3}$,$sk_{ID,4}$. In case of corruption 1,2 are preserved
  - **KeyUpdate**: time 3→4. Erase $sk_{ID,3}$
  - time 4→5: Generate $sk_{ID,5}$, $sk_{ID,5}$, erase $sk_{ID,4}$

A concrete construction of fs-IB-KEM

- ***We start from the [BBG05] HIBE***
- Setup: $MPK=(g, g_1=g^a, g_2, u, v, h_1, \ldots, h_L, z=e(g_1,g_2), H)$, $MSK=g_2^a$
  L tree's depth (upper bound on time periods)
- KeyGen(MSK,ID,t): $w_1,\ldots,w_k$ nodes representing t

  $$d_0=g_2^a(uv^{H(ID)}\Pi h_i^{f(wi)})^r,\ d_1=g^r,\ \{b_i=h_i^r\}_{i=k+1,\ \ldots,\ L}$$

- KeyUpdate($SK_{ID,t}$,t+1): b=0/1 descendant of t
  $d_0=d_0'(uv^{H(ID)}\Pi h_i^{f(wi)}h_{k+1}^{f(b)})^t,\ d_1=d_1'g^t,\ \{b_i=b_i'h_i^t\}_{i=k+2,\ \ldots,\ L}$
- Encrypt(MPK,ID,t): $C_0=(uv^{H(ID)}\Pi h_i^{f(wi)})^s, C_1=g^s, K=z^s$
- Decrypt($SK_{ID,t}$,C): $K=e(C_0,d_1)/e(C_1,d_0)$

- **Theorem:** IND-CPA-secure under *l-wBDHI** assumption in the random oracle model
- Generic conversion to IND-CCA security

Comparison with previous works

Property / Protocol	Tor	PB-OR	CL-OR	Our
Interaction User-OR	☹ (telescoping)	☺	☹ (every update)	☺
Interaction OR-KGC	☺	☹ (every update)	☺	☺
Workload KGC	☺	☹ (every update)	☺	☺
Efficiency??				

Efficiency to build a circuit

- Considering basic operations costs with PBC lib.

Protocol		Total cost (in ms)	
		80-bits	128-bits
Tor	User	2.3n	16.5n
	OR	6.9	93.3
PB-OR	User	1.1n	9.3n
	OR	3.9	57.3
CL-OR	User	2.1n	5.1n
	OR	3.4	8.2
Our	User	7.8n	63.4n
	OR	15.6	178

- **Concrete example:** 80-bits, 3 nodes, network latency (50ms)
  - **Tor:** 627ms
  - **Our protocol:** 370ms

Some Caveats – Key Escrow

Property / Protocol	Tor	PB-OR	CL-OR	Our
Key-Escrow	☺	☹	☺	☹ !!

- **2 possible solutions:**
  1. Generic conversion to the CL-setting

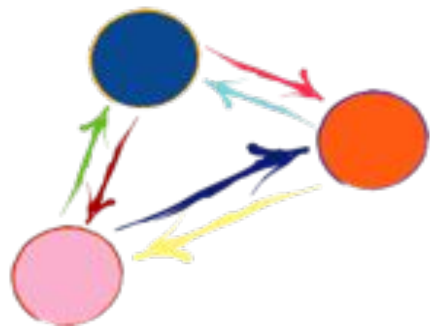
     ☺ No key-escrow

     - Slightly less efficient (it requires running 2 schemes in parallel)
  2. A PKI variation

     ☺ No key-escrow

     - No KGC. Each user acts as its own KGC. It can update keys while the MPK remains always the same.
     - *Same computational efficiency as the id-based one!*
     - *(!) Our scheme has a long public key*
     - ***Recent result (not in the paper):** can obtain constant-size public key using RO*

A look at interaction



- We removed interaction from the cryptographic part of onion routing protocols
- OR protocols still have an interactive component
  - The user has to get the list of active routers
- In our case, list updates do not have to include updated keys (they remain the same)

Conclusions

OUR RESULTS:

1. A general approach for non-interactive onion routing protocols with forward-secrecy
   - It works in either the ID-based, CL, PKI settings
   - Formally prove its security based on the basic ingredients (fs-IB-KEM, SKE)
   - Fixed small flaw in [KZG07]
2. A practical construction that implements our idea

OPEN PROBLEMS:

- More efficient constructions of fs-IB-KEM

Thanks!