Improving Wireless Privacy with an Identifier-Free Link Layer Protocol

Ben Greenstein, Damon McCoy, Jeffrey Pang, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall

Presented by Victoria Kirst

Problem

- Ubiquity of wifi devices increases privacy risks
  - Transmissions are broadcasted, so wireless more exposed than wired
- Easy to eavesdrop w/ free software
  - Use standards like **WPA 802.11** to encrypt…

WPA 802.11 Not Sufficient

- Low level identifiers (network names, addresses) used to find high-level identifiers (identities)
  - Probe requests show networks they're trying to read, authentication information, MAC addr, etc. in the clear

  - **Can link together**
    - Tracking and Inventorying (sender, receiver identities)
    - Profiling (sender, receiver relationships)

802.11 Probe	Is Bob’s network here?

802.11 Beacon	Bob’s network is here

Solution: Remove all identifiers!

- **SlyFi**: 802.11-like protocol that encrypts entire packets to remove explicitly identifiers

- How to communicate?
  - How do I know if I’m the destination?
  - How can I announce that I’m here?

- **All this can be supported without exposing identity**
  - Hide entire message contents from third parties
  - Prevent third parties from “linking” any two packets

Objective

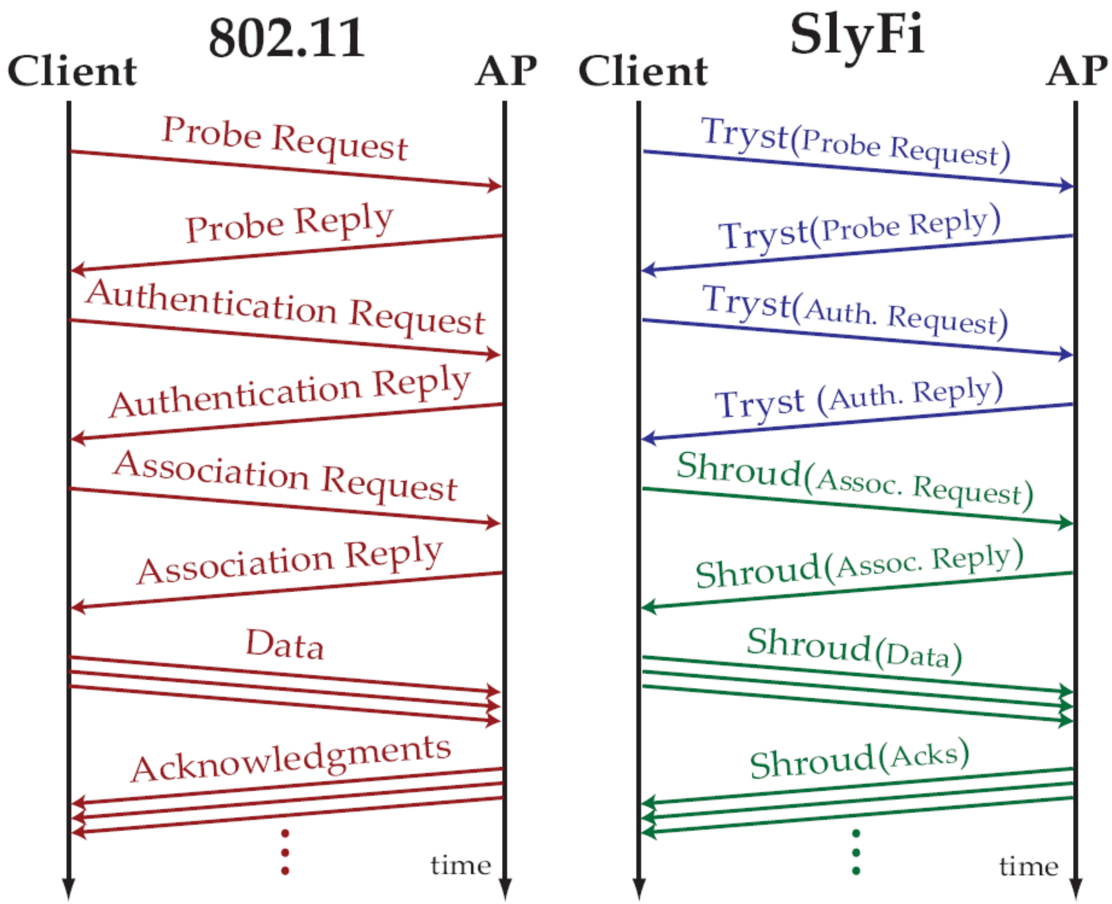
- When *A* generates *Message* to *B*, he sends:

  PrivateMessage = **F**(*A*, *B*, *Message*)

  where **F** has these security properties:

  - **Unlinkability**: Only *A* and *B* can link *PrivateMessages* to same sender or receiver.
  - **Authenticity**: *B* can verify *A* created *PrivateMessage*.
  - **Confidentiality**: Only *A* and *B* can determine *Message*.
  - **Integrity**: *B* can verify *Message* not modified.
  - **Efficiency:** *B* can process *PrivateMessage* fast as he can receive them.

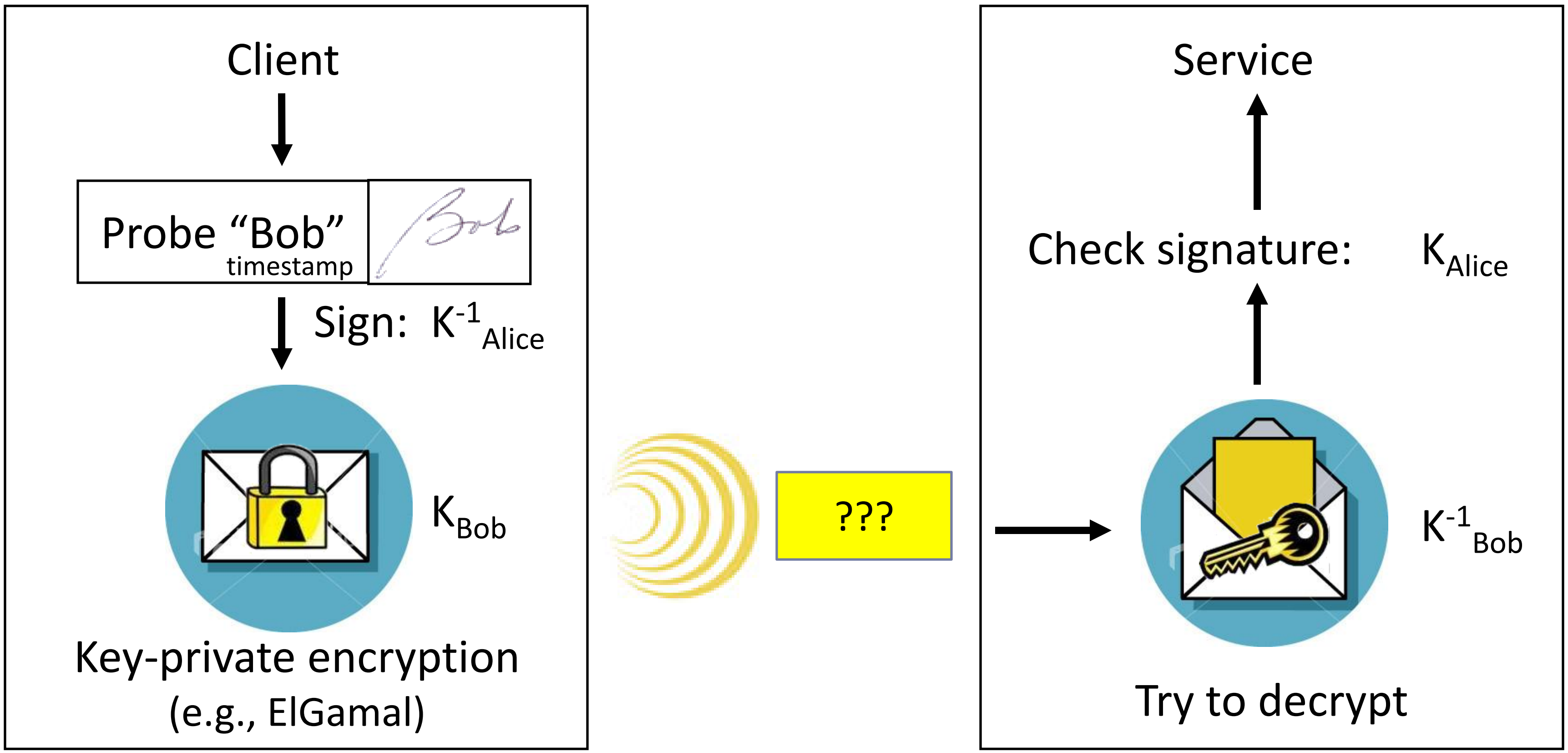
Solution Overview



Straw Man: Public Key Mechanism

- Alice signs statement, encrypts w/ Bob's public key
  - Uses encryption that does not reveal which key is used, so sender/recipient anonymous
- Bob then tries to decrypt all messages he receives
  - When successful, check signature and time
- **SLOW:** Bob can be backlogged trying to decrypt all the messages

Straw Man: Public Key Mechanism



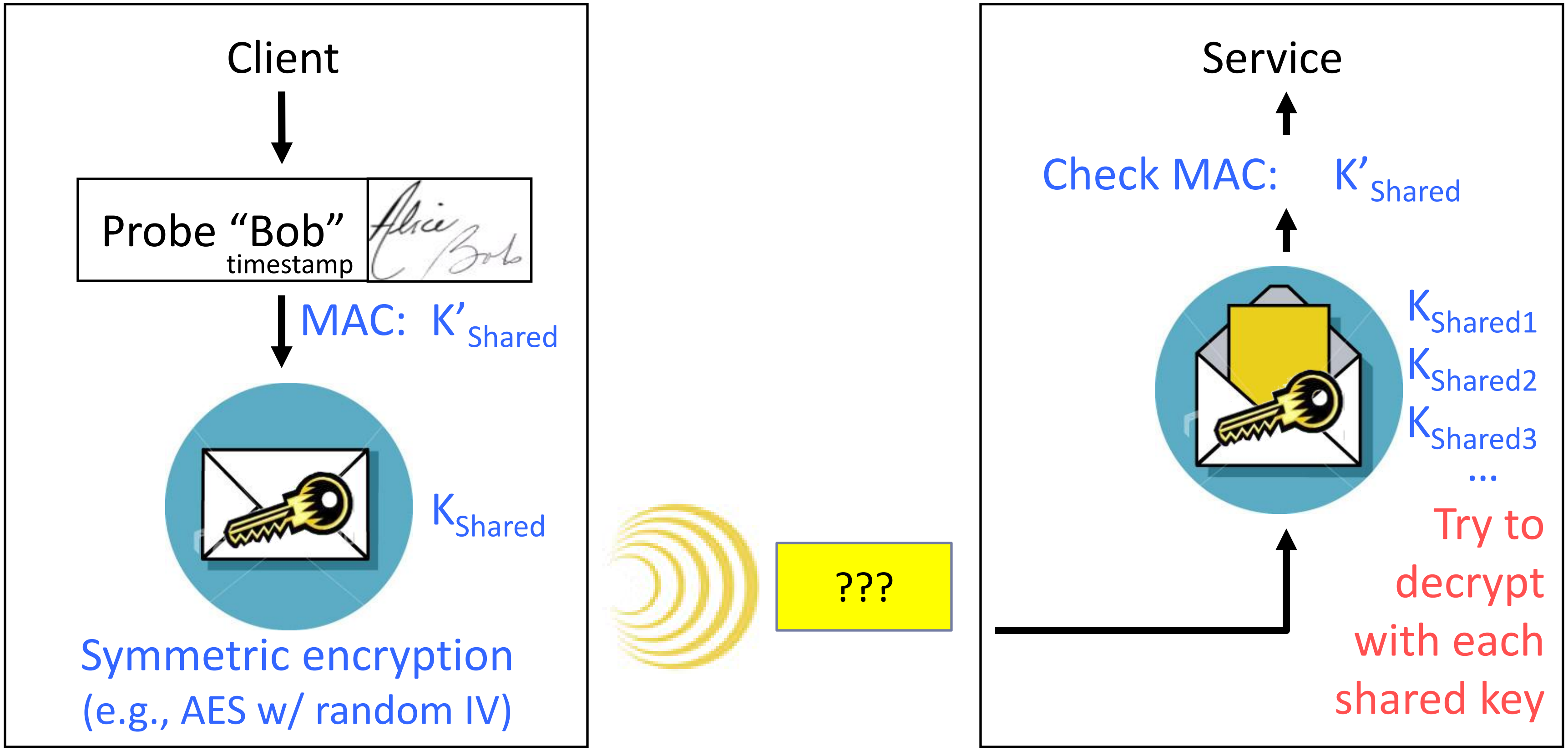
Straw man: Symmetric Key Protocol

- Alice encrypts statement using symmetric encryption (AES), generates MAC

- Bob verifies MAC in header with his key

- **SLOW:** Must try all symmetric keys he has
  - Can use locality by sorting keys by most-recently-used
  - Still slow for messages not intended for Bob
    - Especially if Bob has many keys

Straw man: Symmetric Key Protocol



Approach

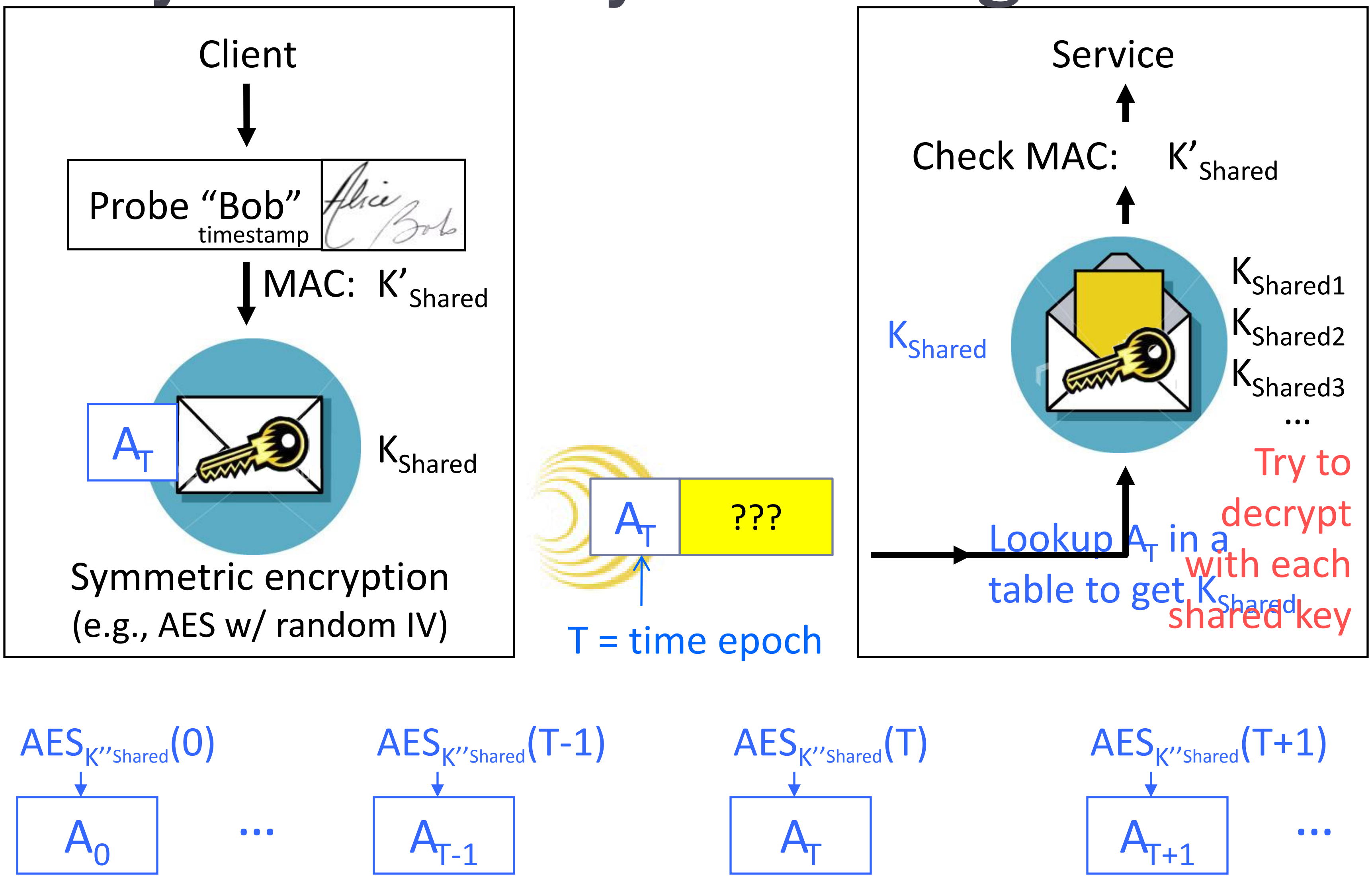
Tryst and Shroud

- Make a few key simplifying assumptions to speed up efficiency
- **Tryst: Discovering and binding**
  - Infrequent: only sent once per association attempt
  - Narrow interface: single application, few side-channels
  - Linkability at short timescales is usually OK
  - Can use temporary unlinkable addresses
- Shroud: Data transport

Tryst: Discovery & Binding

- Based off of Symmetric Key Straw Man
- Alice and Bob generate sequence of unlinkable addresses based on T_0 (time of initial key exchange)
  - T_i for every time interval I
- Bob maintains hash table of Addresses(T_i) –> Key; table updated every time interval
  - Use fast table lookup for key instead of trying all keys

Tryst: Discovery & Binding

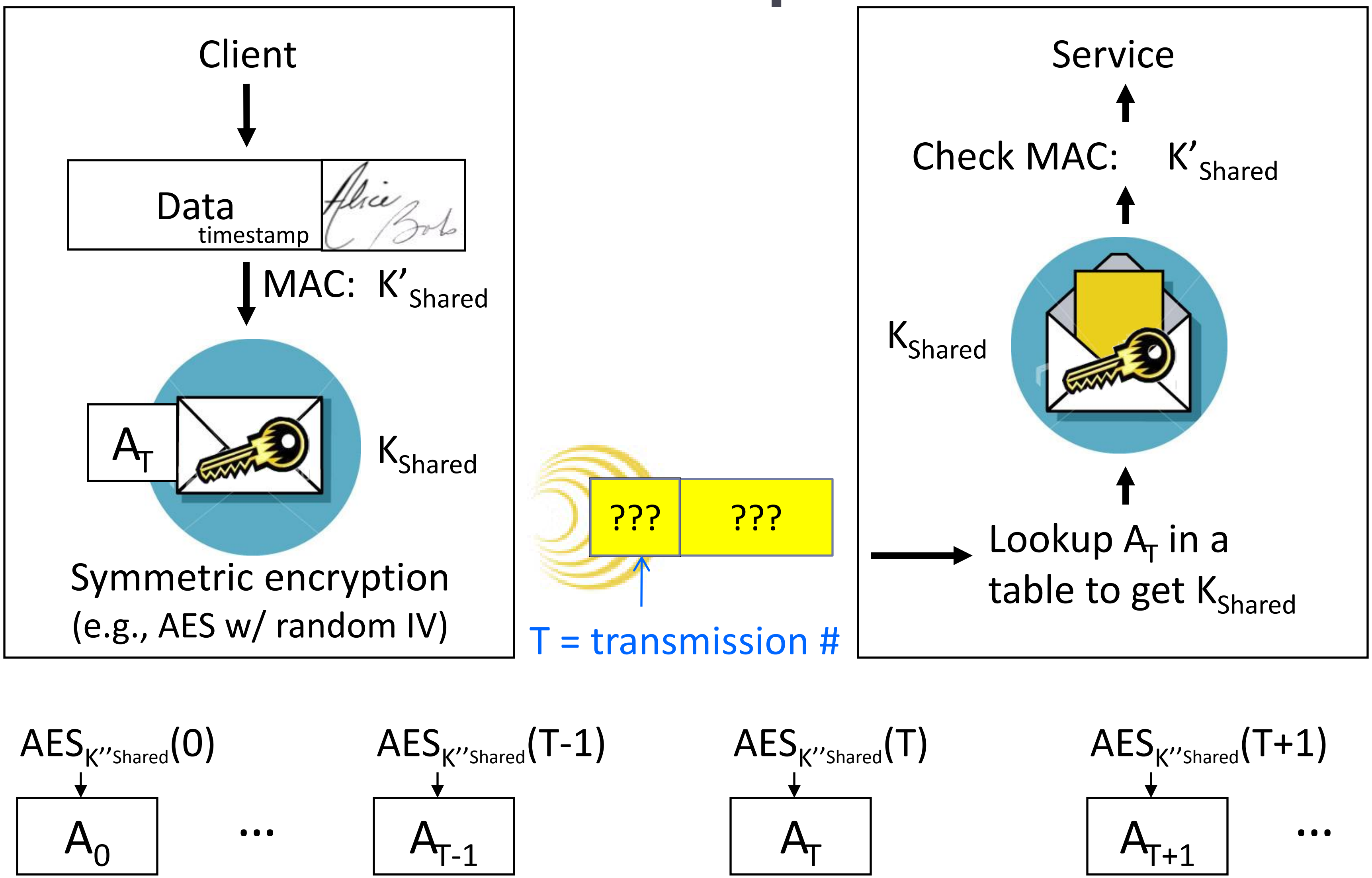


Shroud: Data transport

- Tryst assumptions not sufficient for data transport
- New assumptions for data:
  - Only sent over established connections
  - Expect messages to be delivered, barring message loss

- **Similar to Tryst:** generate addresses at Ti, but Ti is transmission number i instead of time interval
- In authentication messages, exchange random session keys for A and B (protected by Tryst)
- Bob maintains table of Addresses(T_i) –> Key; table updated every new packet

Shroud: Data transport



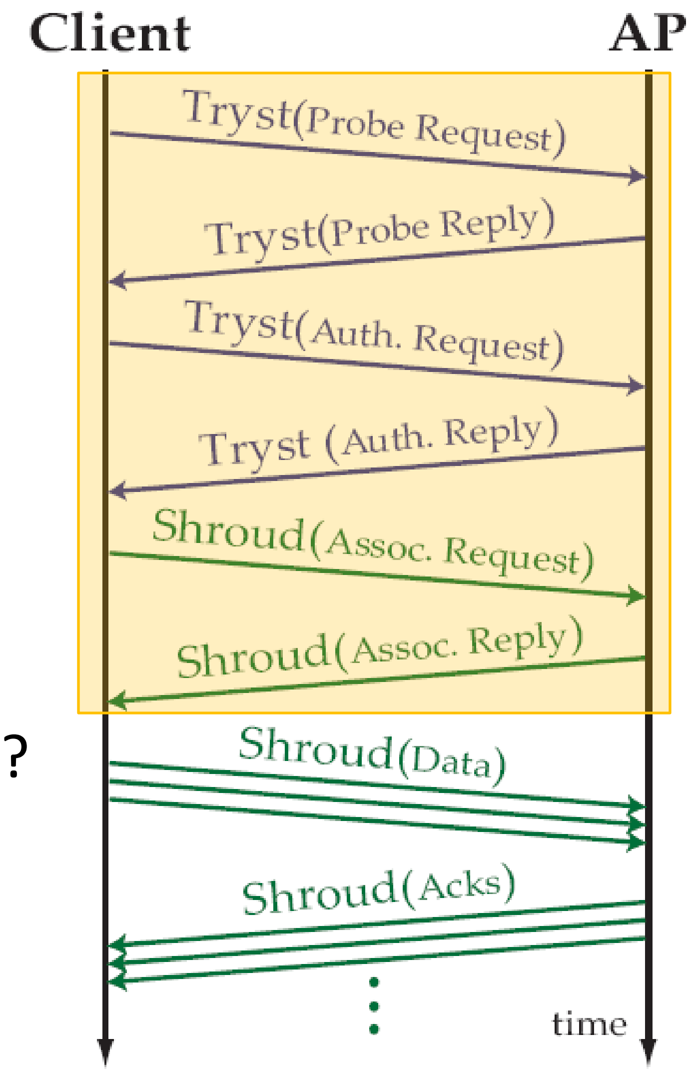
Shroud: Data transport

- On receipt of packet with address A_T, compute next address A_{T+1}
- Handling message loss:
  - Compute $A_{T+1}, \ldots, A_{T+k}$
  - Can progress unless *k* consecutive packets are lost
  - Studies show *k*=50 sufficient for vast majority of cases
  - Common case: compute 1 new address per reception, except first packet, which requires 49 computations

Evaluation

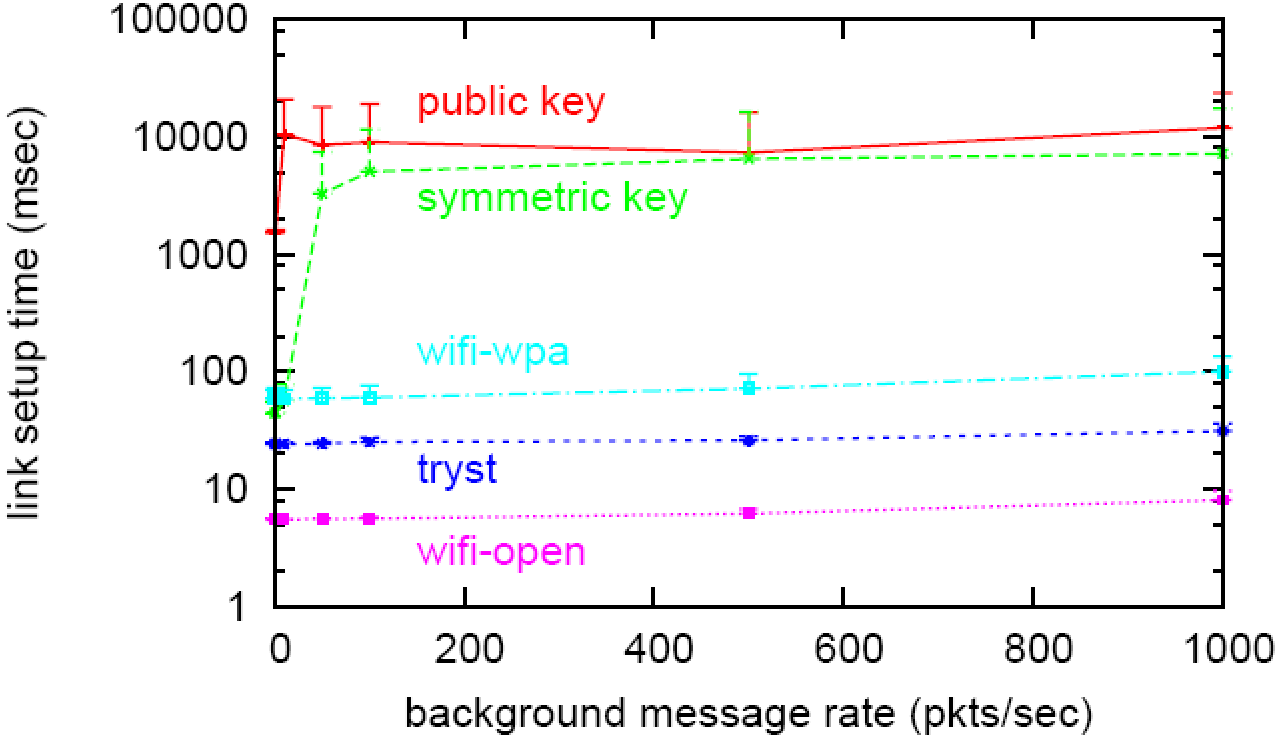
Evaluation Metrics

- **Link setup time**
  - Time to discover and setup a link
  - Lower $\Rightarrow$ shorter wait to deliver data, less interruption when roaming



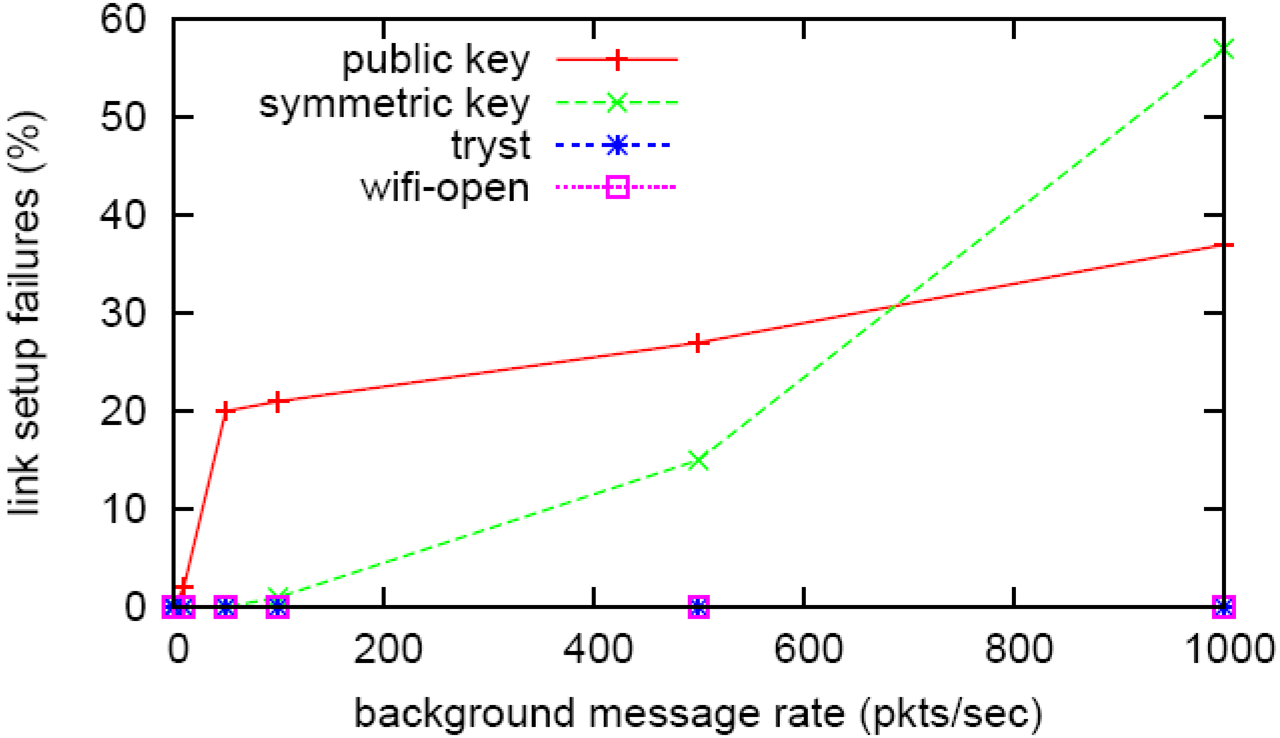
- Key questions:
  - Is address computation overhead large?
  - Can Tryst filter messages efficiently?

Link Setup Time vs. Background Rate



Tryst has less overhead than WPA

Link Setup Failure vs. Background Rate

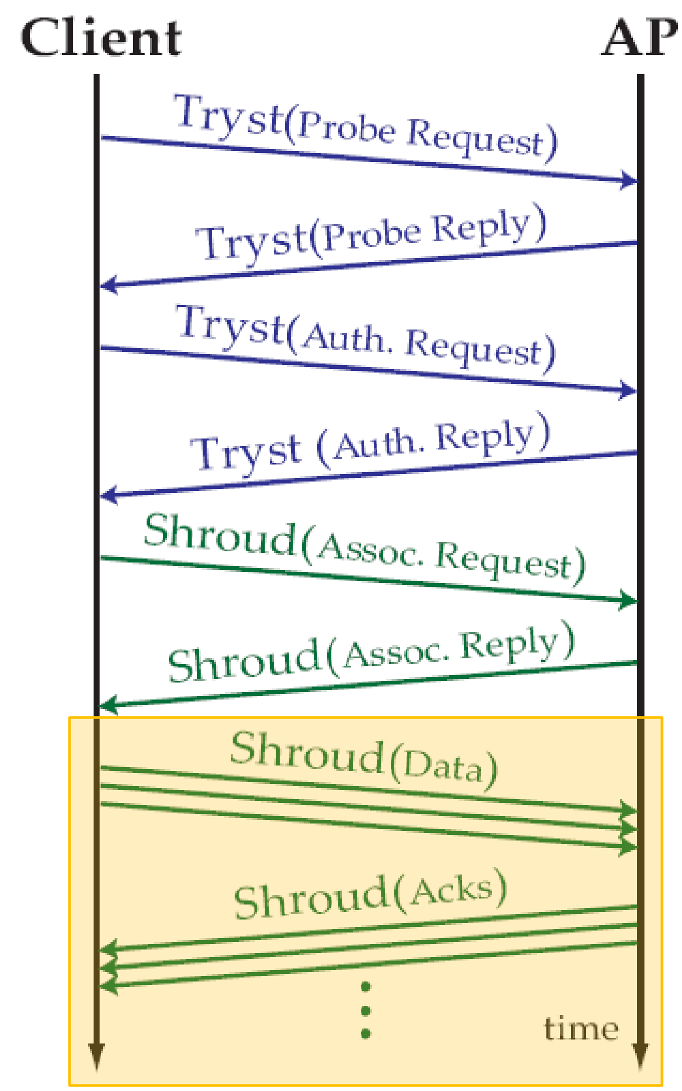


Tryst filtering is much more efficient than straw men

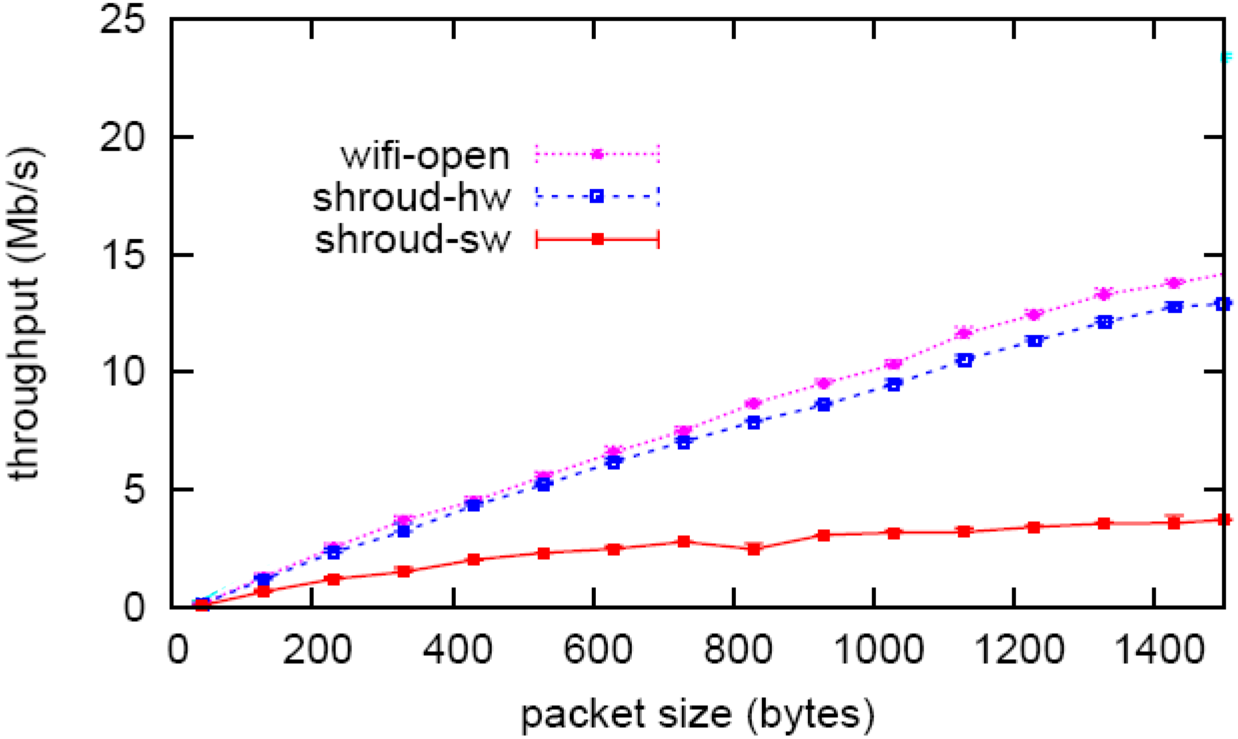
Evaluation Metrics

- **Data throughput**
  - How fast can link deliver data
  - Higher $\Rightarrow$ faster applications

- Key questions:
  - What is Shroud's overhead?
  - Can Shroud filter messages efficiently?

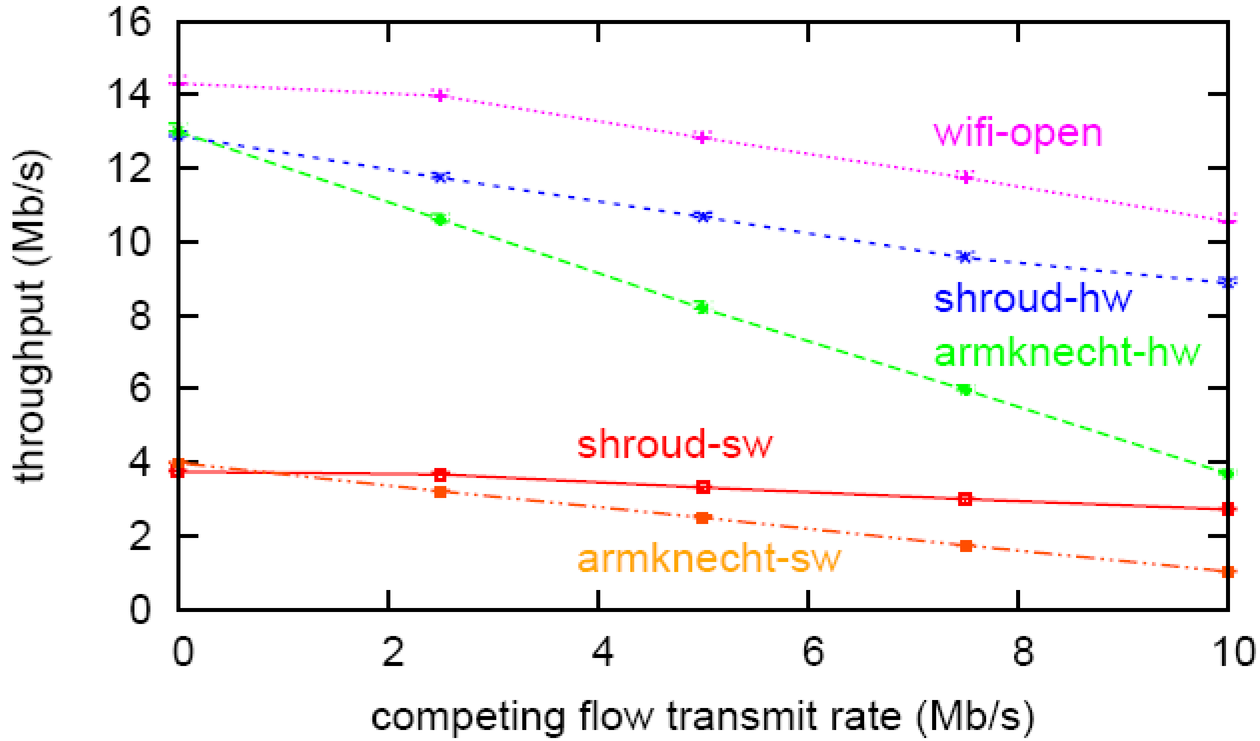


Data Throughput vs. Packet Size



Shroud overhead is similar to WPA

Data Throughput vs. Background Rate



Shroud filtering is almost as efficient as 802.11

Improvements and Open Questions

- Known limitations:
  - Packet sizes, packet timings, and physical layer might still be used to link packets together

- SlyFi can be introduced incrementally because it falls back to normal 802.11 if no SlyFi-enabled access point is found
  - Introduce security risks in the future if SlyFi were to become a more prevalent protocol?