Perspectives: Improving SSH-style authentication using multi-path probing

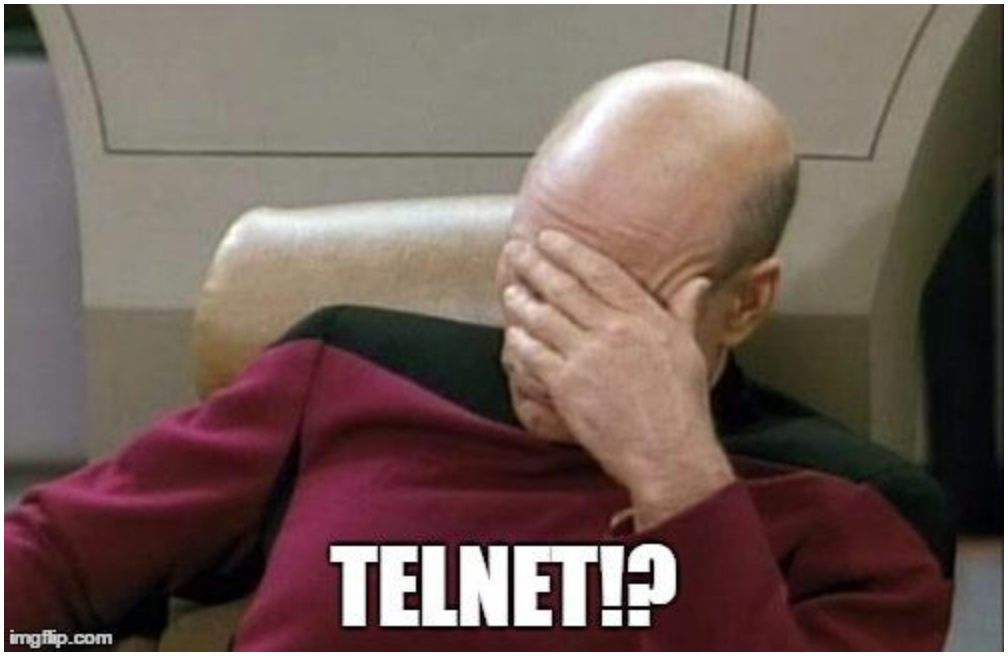
Dan Wendlandt, David G. Andersen, Adrian Perrig
- ATC'08

By Hassan Shahid Khan

CS 598 - COMPUTER SECURITY IN THE PHYSICAL WORLD

In the beginning of times..

- Telnet
- r* services (rlogin, rsh)
- Weak (or no) authentication
- Communication in the clear

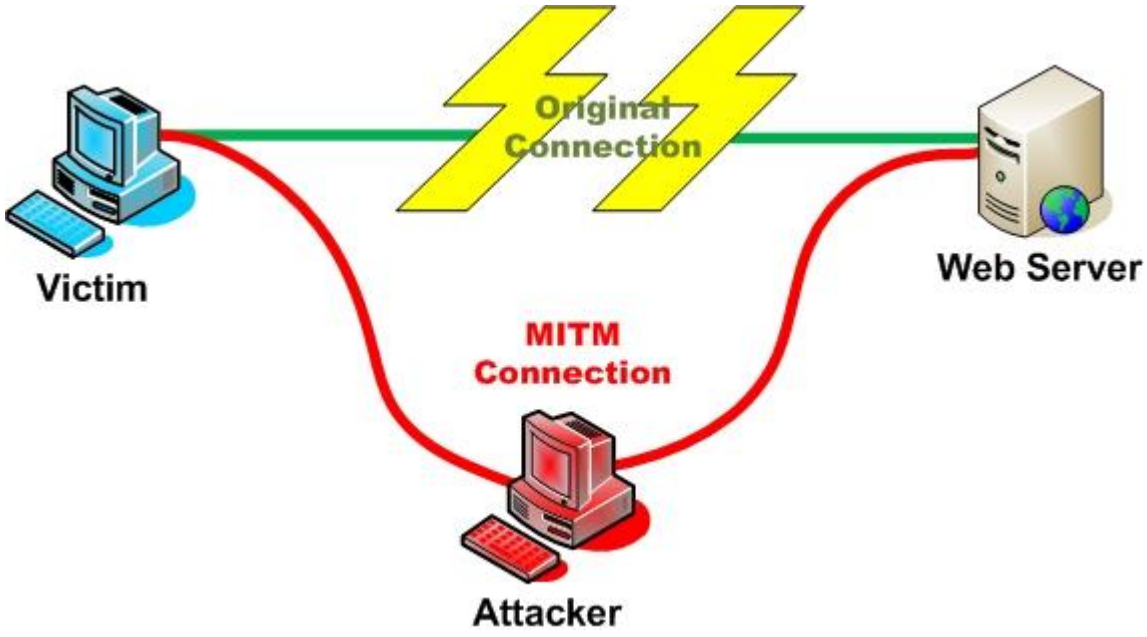


Enter SSH/SSL

- Provided the cryptographic elements to build a tunnel for confidential data transport with checked integrity

However..

- SSH/SSL authentication based on asymmetric cryptography
- Diffie-Hellman key exchange subject to MITM attack.



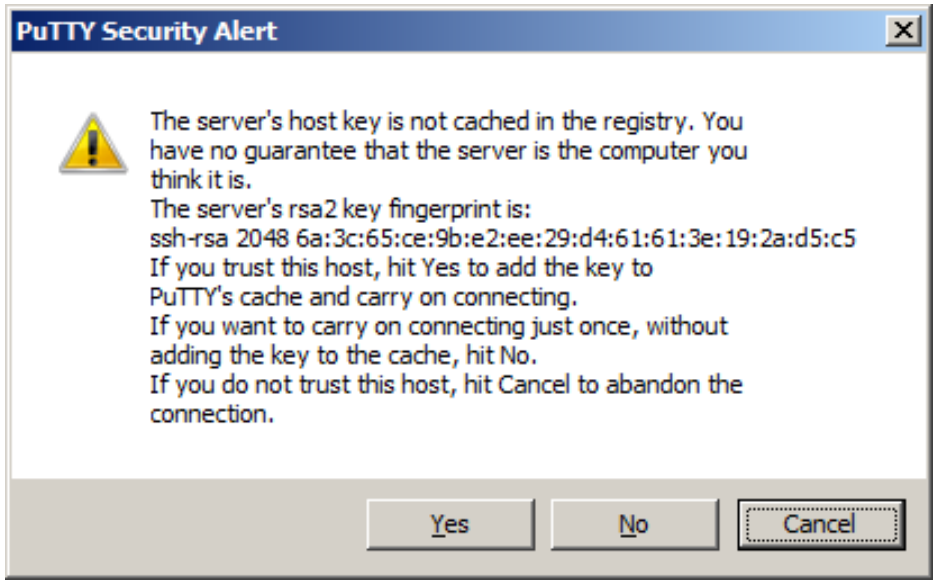
Should I be worried about MitM?

- Recent trends increase MitM vulnerability
  - Other hosts on a wireless can spoof ARP/DNS.
    (e.g., ARPIFrame worm)
  - Access points/home routers may be poorly administered or have known vulnerabilities.
    (e.g., "Pharming" attacks)
- These attacks are automated & profit driven

Obtaining Authentic Public Keys

Two standard approaches to handling MitM attacks:

- Public Key Infrastructure (e.g., Verisign certs)
- Trust on first use (TOFU) mechanism



Trust-on-first-use Authentication

1) Assume no adversary on first connection, cache key
2) If key changes*, panic!

Seems insecure, why use it?

- Unlike PKI, it's simple & cheap.
- No manual work when adding a server, just plug-and-play.

*SSH keys do change legitimately

Goals of this paper

- Significantly improve attack resistance for Tofu

- Keep simple SSH-style deployment model.

Key observation for SSH

With Tofu, clients face a security decision:

- When first connecting to a server.
- Any time a key mismatch is detected.

But Tofu gives little/no helpful information!

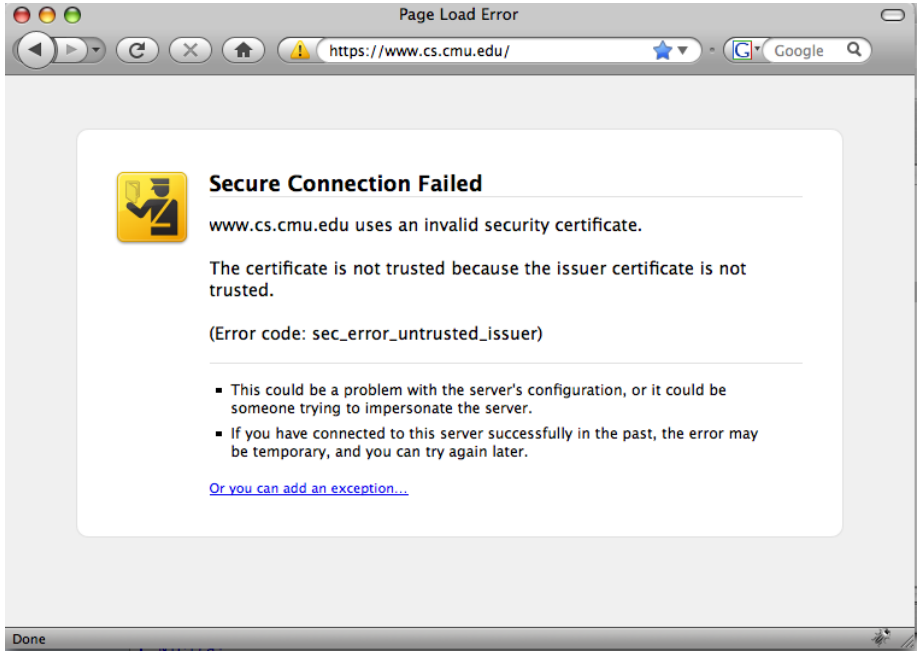
```
The authenticity of host 'host.domain.com (192.168.74.49)' can't be established.
RSA key fingerprint is 07:fd:fb:9b:03:a2:b4:e8:b3:c9:0f:0b:db:43:1c:1a.
Are you sure you want to continue connecting (yes/no)?
```

or

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the DSA host key has just been changed.
The fingerprint for the DSA key sent by the remote host is
4c:68:03:d4:5c:58:a6:1d:bd:17:13:84:14:48:ba:99.
Please contact your system administrator.
```

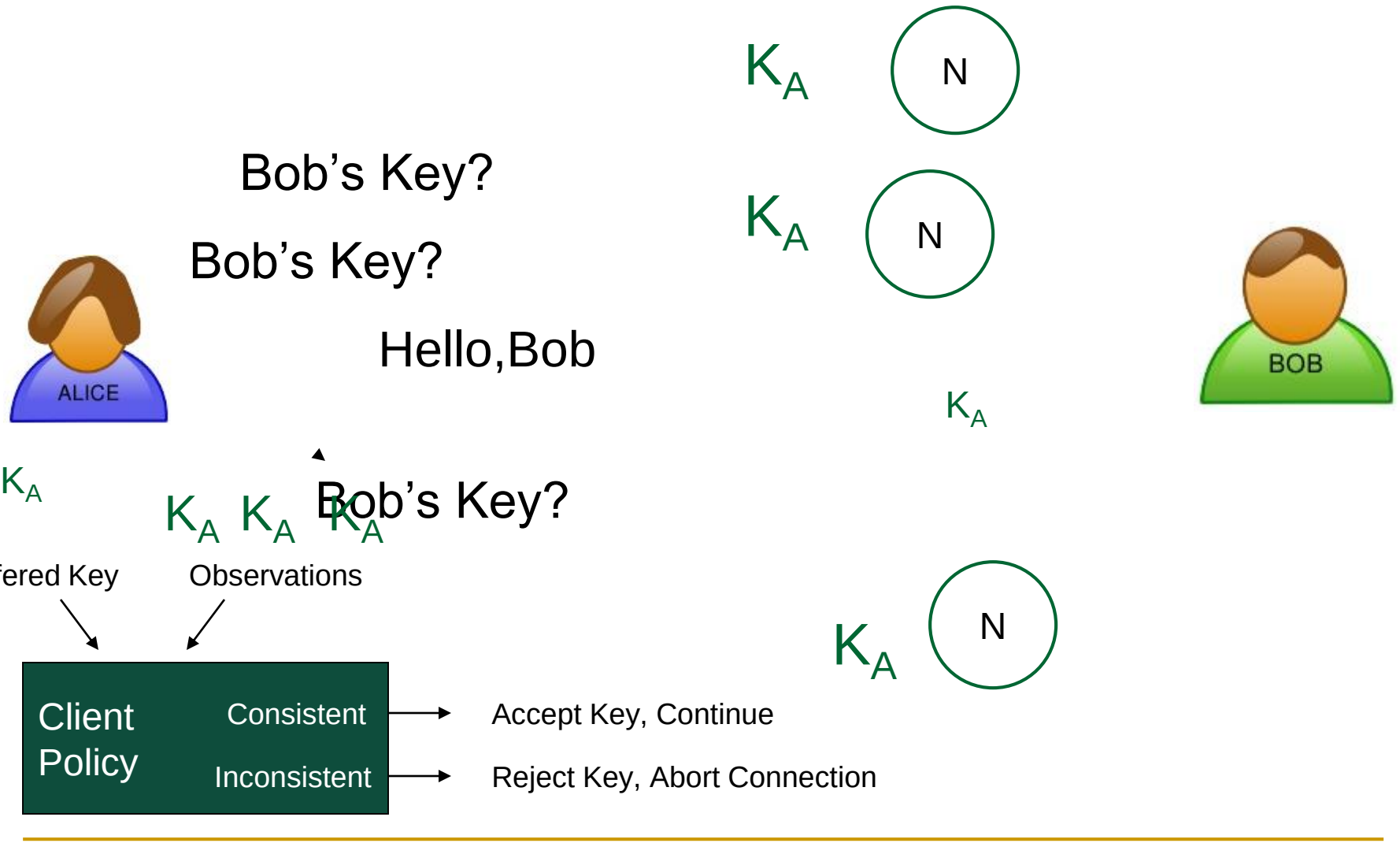
Key observation for SSL

- Difficult for users to validate new/changed keys with self-signed certs.
- Frequent spurious warnings “train” users to ignore ALL warnings



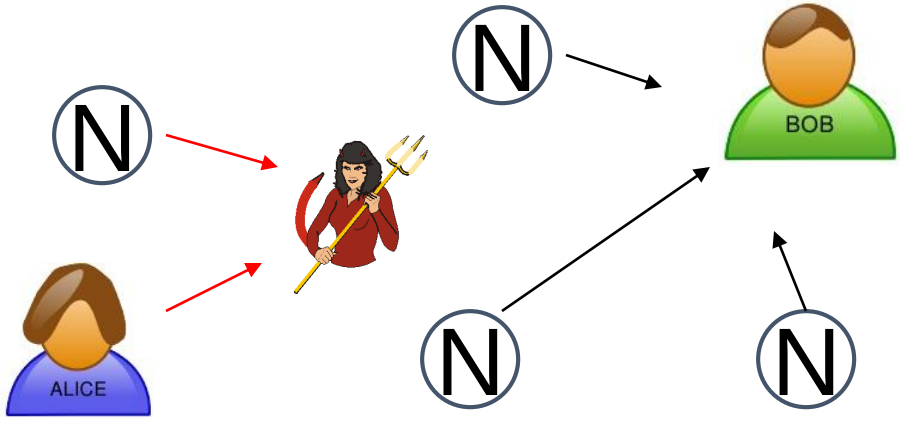
Perspectives provides additional data to distinguish between an attack and a spurious warning.

Perspectives Overview



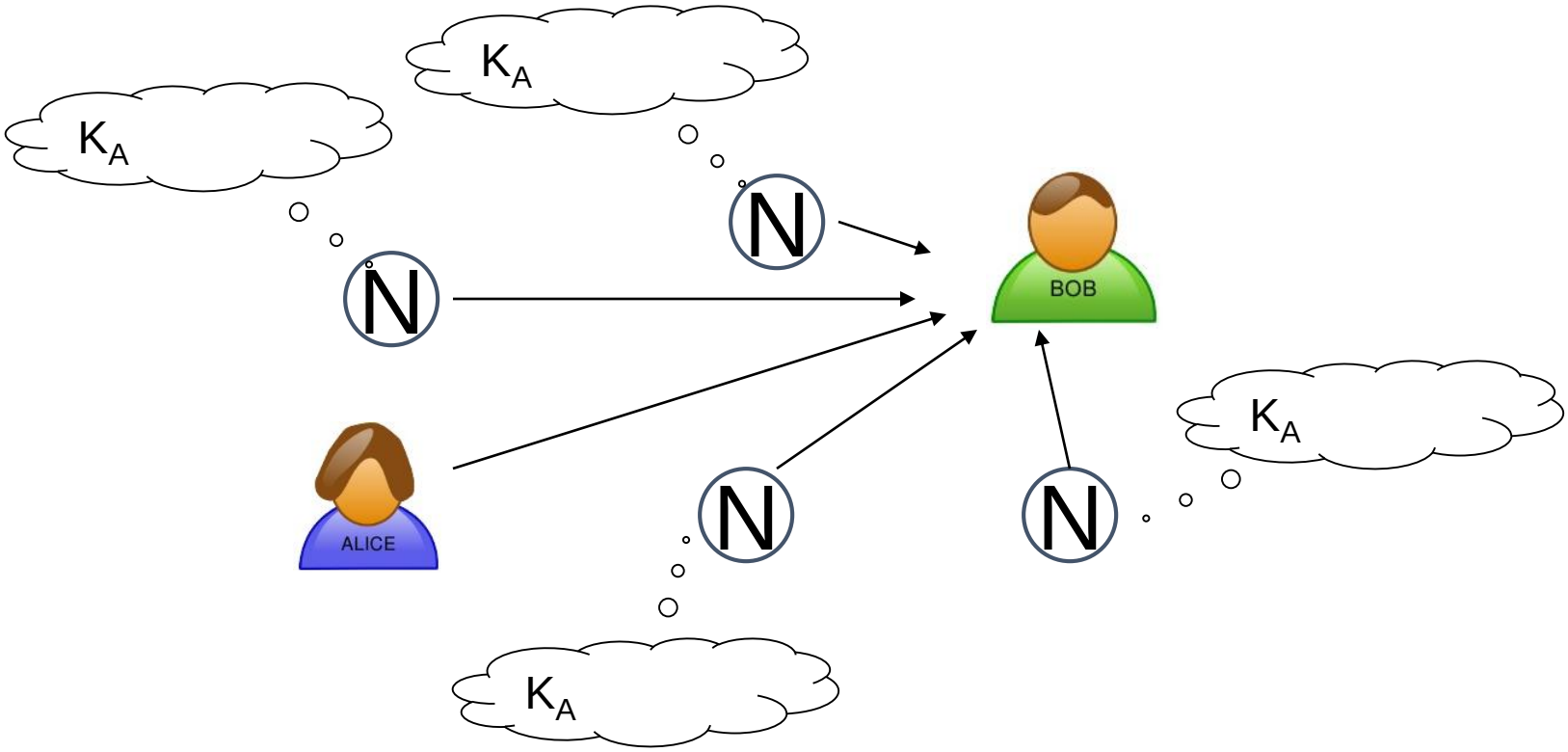
Spatial Resistance

Multiple vantage points to circumvent localized attackers



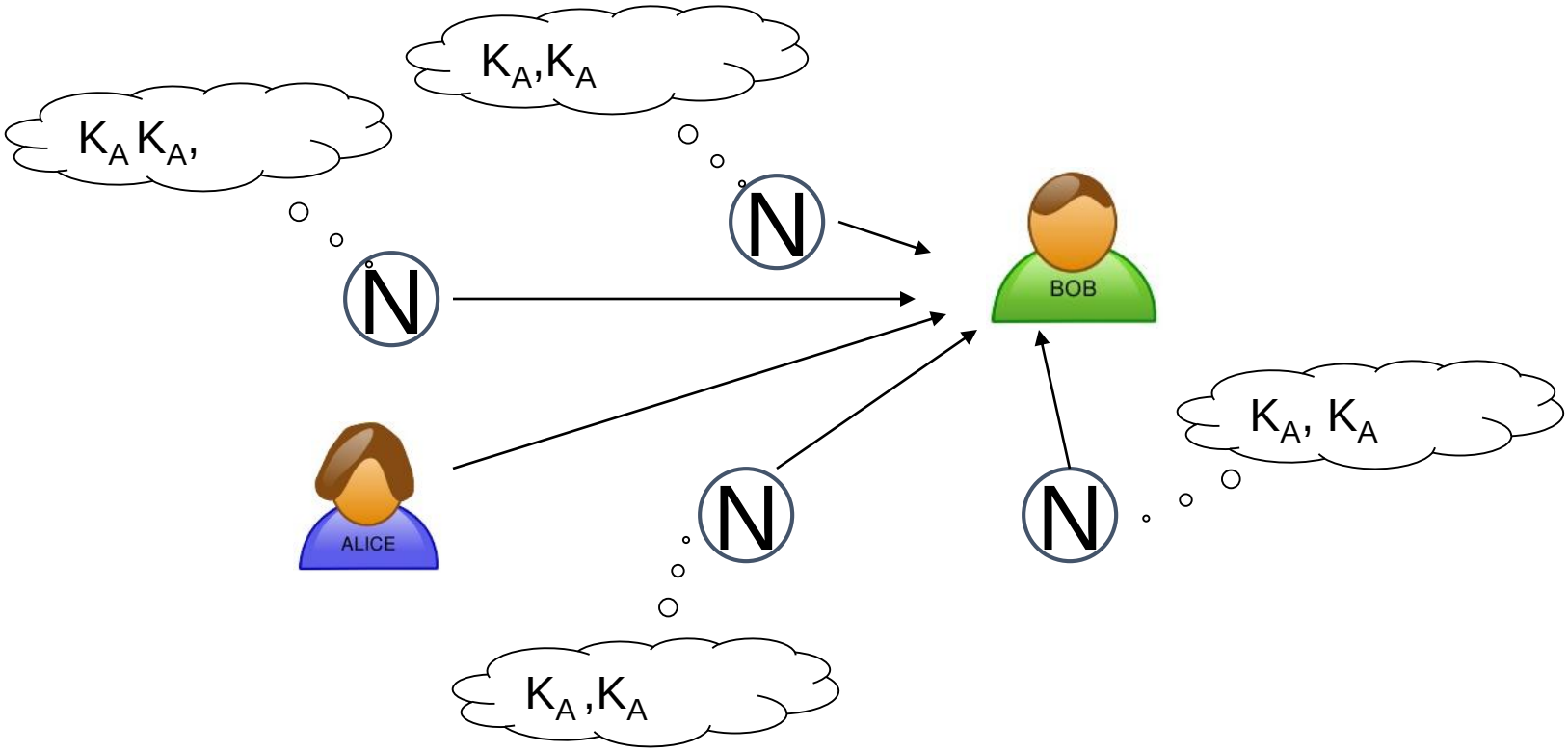
Temporal Resistance

Key history raises alarm even if all paths are compromised.



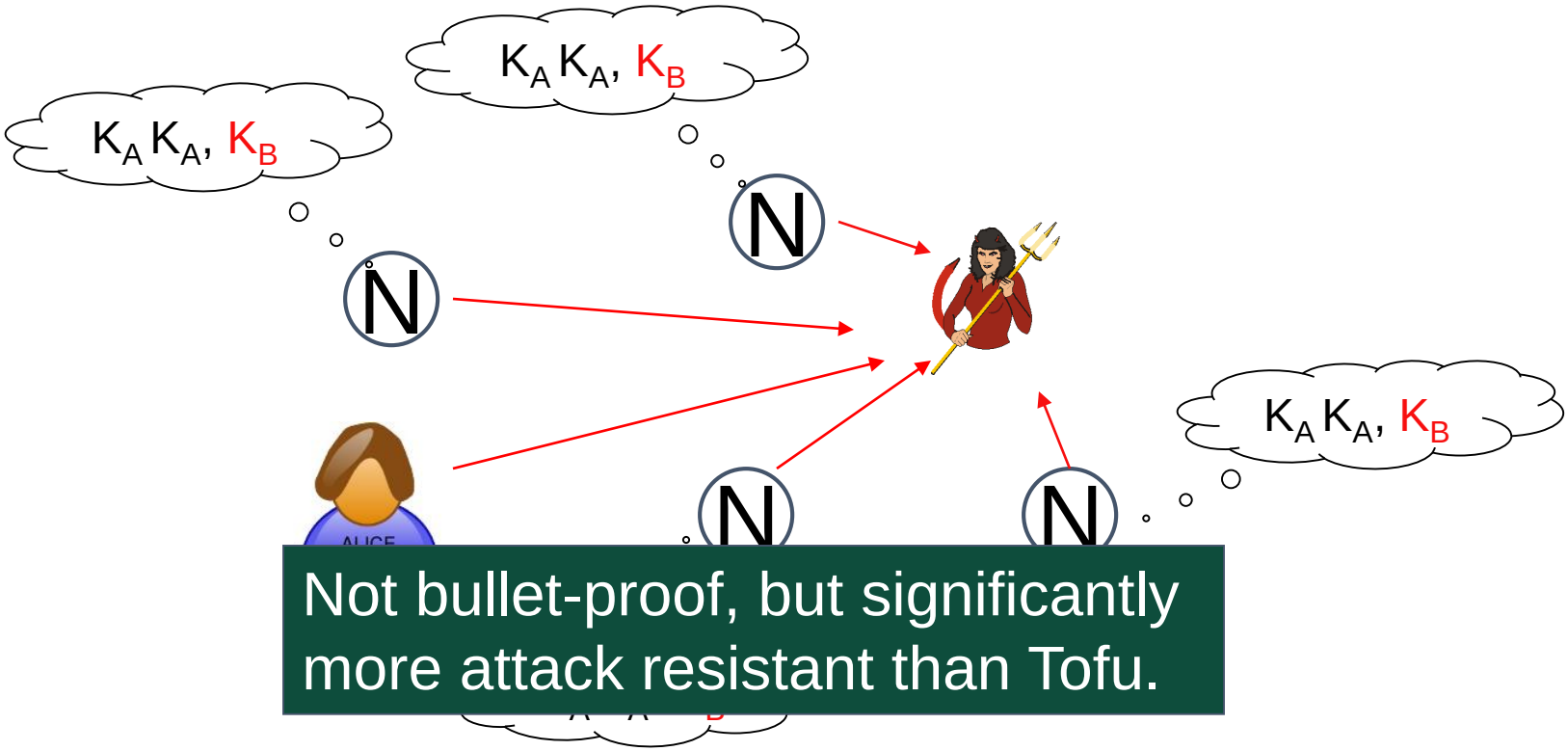
Temporal Resistance

Key history raises alarm even if all paths are compromised.



Temporal Resistance

Key history raises alarm even if all paths are compromised.



Not bullet-proof, but significantly more attack resistant than Tofu.

Perspectives Design

- Who runs these network notaries?
- How do notaries probe servers?
- How do clients use notary data to accept or reject a key?

Who runs notary servers?

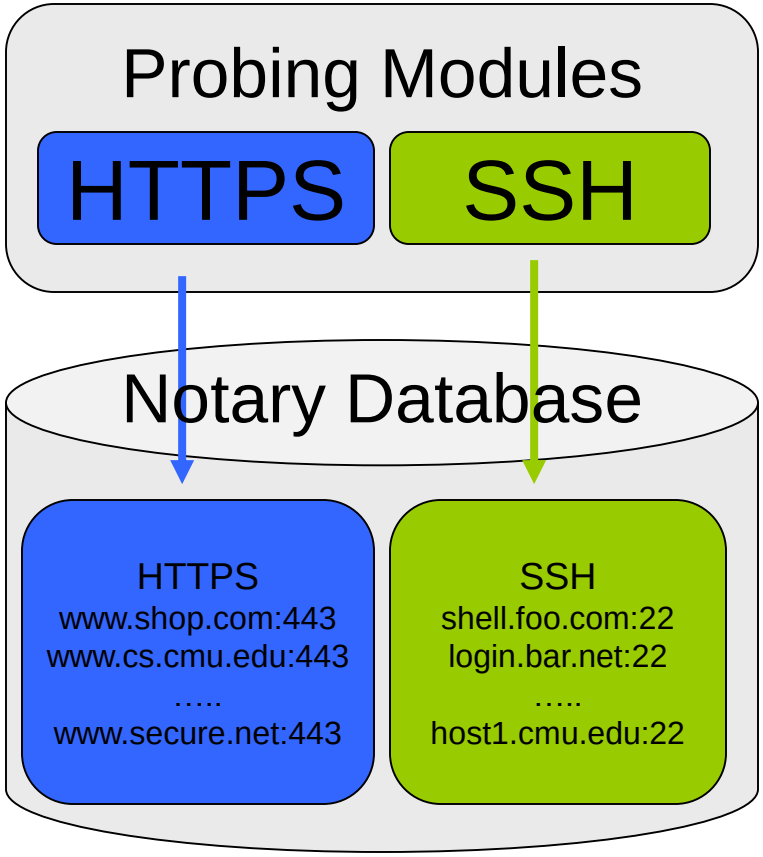
- A “community deployment” with universities, ISPs, or hosting providers volunteering to host a single notary.
  - Public traceroute & looking-glass servers
  - Academic network testbeds like PlanetLab and RON.

- Design assumes notaries are only “semi-trusted”.

- Clients regularly download “notary list” to bootstrap.

  [notary ip, notary public key]
  [notary ip, notary public key]
  ……
  [notary ip, notary public key]

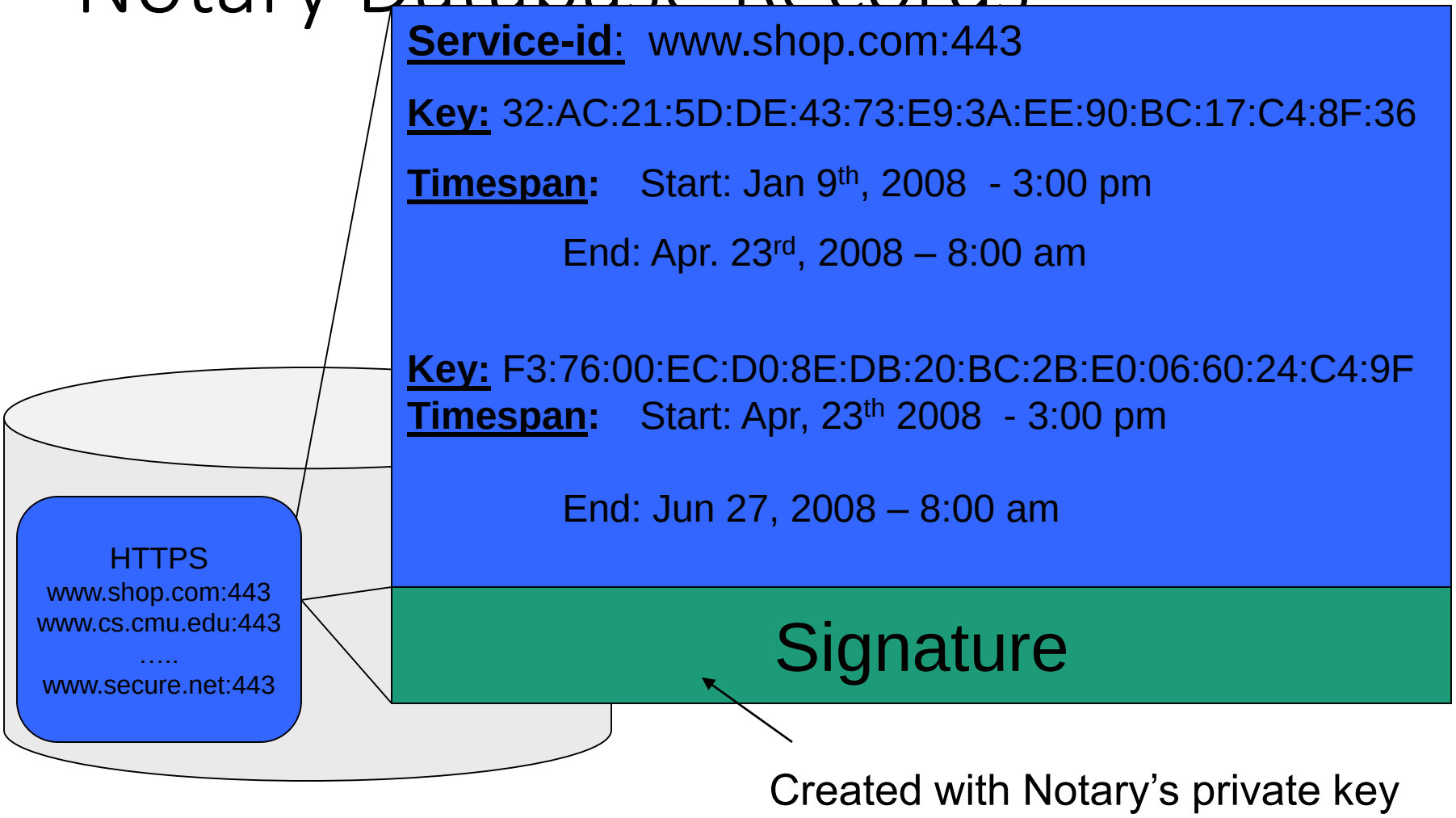
How do notaries monitor keys?



- Probing modules mimic client.

- Notary regularly (e.g. daily) probes each service listed in database and updates its info.

Notary Database Records



Compromised notaries?

Data redundancy

- Each notary acts as a *shadow server* for several other notaries.
- *A shadow server* stores an immutable record of each observation made by another notary.
- Whenever a client receives a query reply from a notary, it can also check and compare reply history with one or more of that notary's shadow servers

Client Policies to accept/reject a key.

- Test spatial and temporal “consistency”.

- Many possible approaches to policies:

  - Manual (power users)

    or

  - Automatic (normal users)

Manual Key Policies: Power Users

Give sophisticated users more detailed info than Tofu.

- 6/6 notaries have consistently seen the offered key from this service over the past 200 days.
- 4/6 notaries currently see a different key!
- All notaries have seen the offered key for the past 8 hours, but previously all consistently saw key Y!

Automated Key Policies: Normal Users

quorum: minimum notary agreement needed to consider a key valid.

Notary #1	Notary #2	Notary #3	Notary #4	Notary #5
K_A	K_A	K_A	K_B	K_A

If offered key is K_A: if Q <= 80% then Accept
else then Reject

Automated Key Policies: Normal Users

Quorum must be a fraction of the total number of queried notaries, not responses received.

Notary #1	Notary #2	Notary #3	Notary #4	Notary #5
K_A	K_A	K_A	K_B	K_A

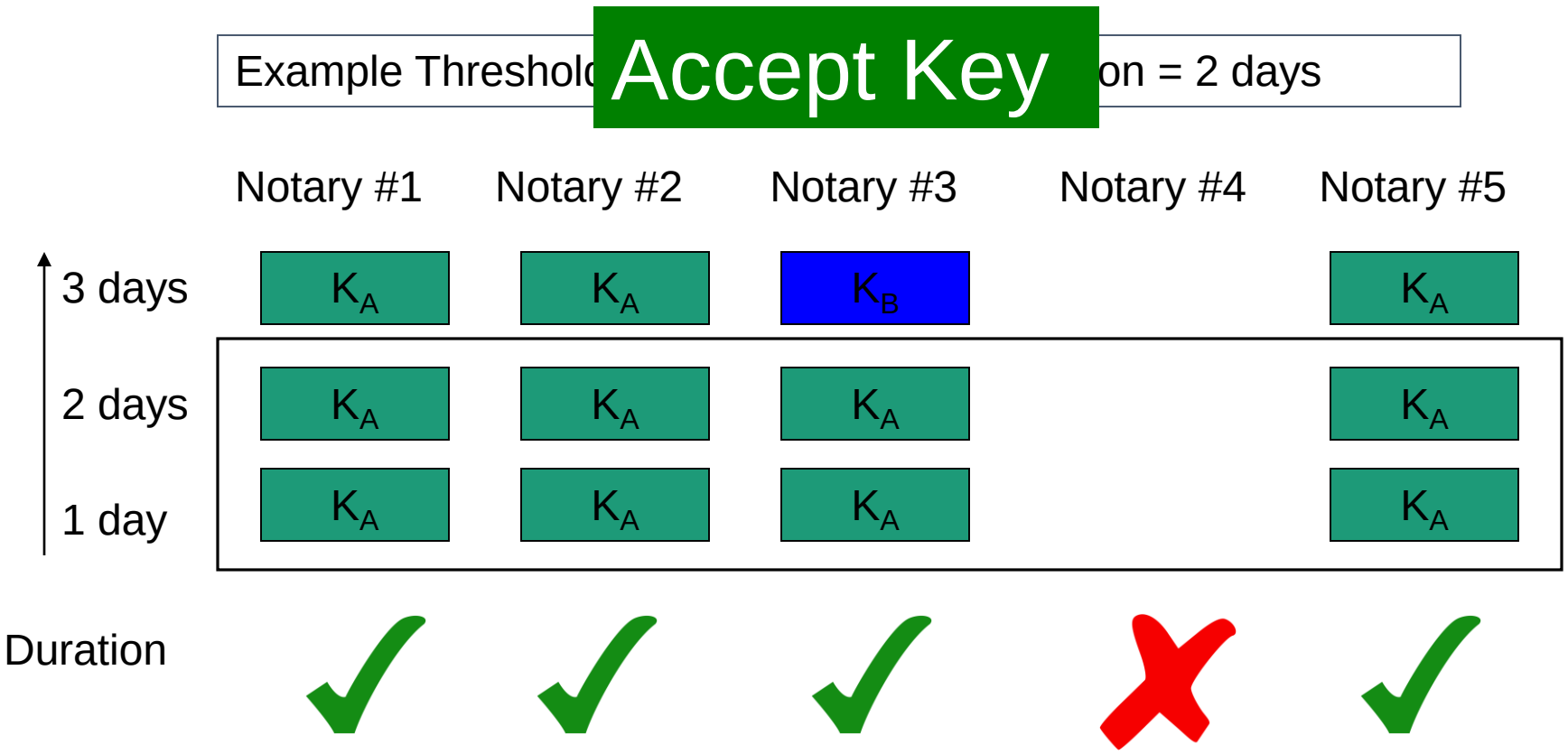
Adversary on client link can selectively drop notary replies.

Automated Key Policies: Normal Users

- Define “quorum duration” : given quorum threshold,
  the length of time a particular key has held quorum.

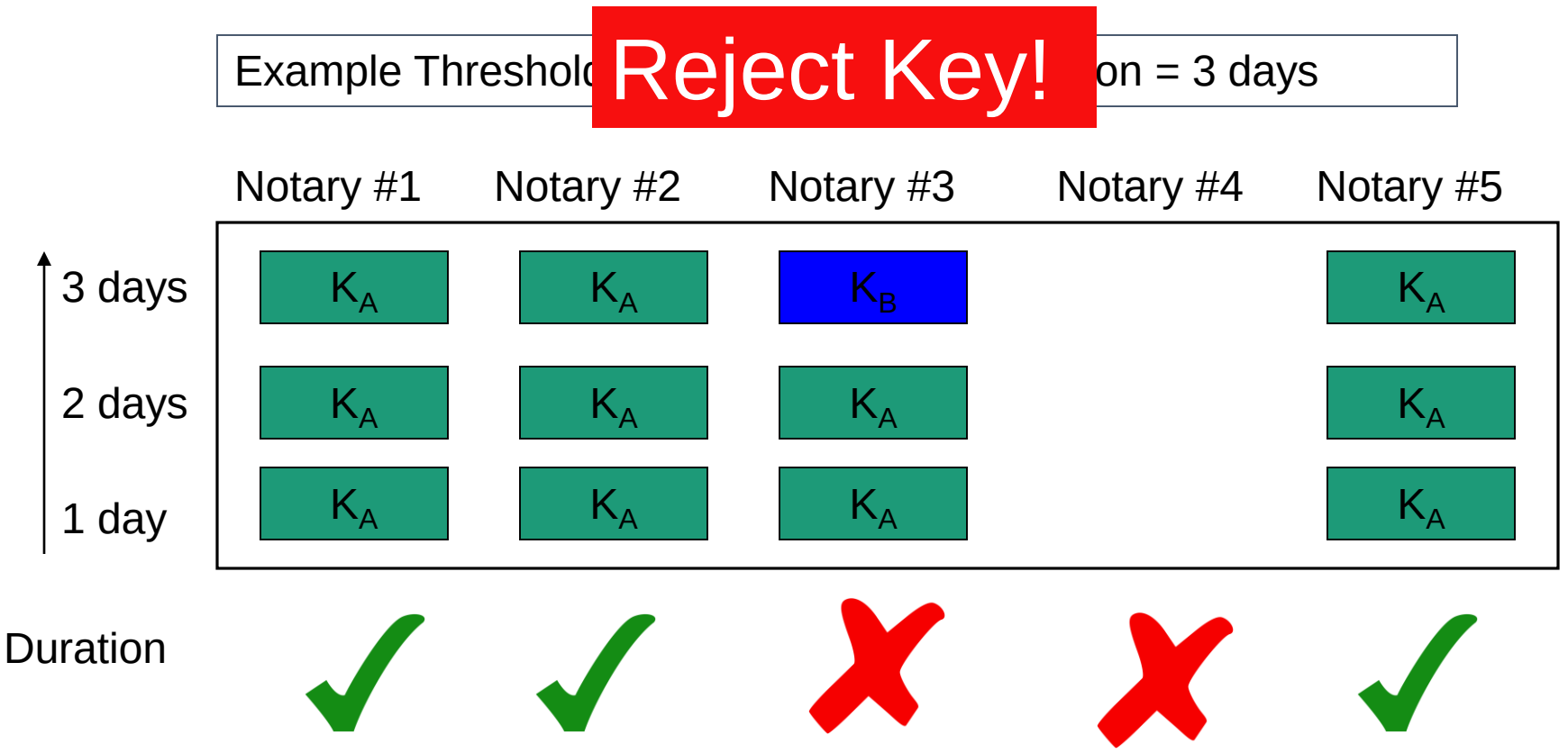
Automated Key Policies: Normal Users

- Define "quorum duration" : given quorum threshold, the length of time a particular key has held quorum.



Key Policies: Normal Users

- Define "quorum duration" : given quorum threshold, the length of time a particular key has held quorum.



Security vs. Availability

- Fundamental network authentication trade-off:

  Clients gain security at the cost of availability (i.e., rejecting a key and disconnecting).

- quorum/quorum duration" encode this trade-off:
  - Higher quorum threshold is more secure:

    => but client is more likely to reject valid key due to notary compromise or failure.
  - Higher quorum duration threshold is more secure:

    => but client rejects valid servers with new keys.

Contrast with PKI

- Perspectives allows each client to individually make a security vs. availability trade-off.

- In contrast a traditional PKI applies a single criteria for all clients.

Security Analysis

		Tofu		Perspectives
Compromise	**DoS**	**MitM**	**DoS**	**MitM**
L_{client}	**X**	**X**	**X**	**safe**
L_{server}	**X**	**X**	**X**	**temporal safe**
$k \cdot n_m$	**safe**	**safe**	$k > (n - q)$: **X** $k \leq (n - q)$: **safe**	**safe**
$L_{server} + L_{client}$	**X**	**X**	**X**	**temporal safe**
$L_{client} + k \cdot n_m$	**X**	**X**	**X**	$k \geq (q + q \cdot r)$: **X** $k \geq q$: **temporal safe** $k < q$: **safe**
$L_{server} + k \cdot n_m$	**X**	**X**	**X**	$k \geq (q + q \cdot r)$: **X** $k < (q + q \cdot r)$: **temporal safe**
$L_{server} + L_{client} + k \cdot n_m$	**X**	**X**	**X**	$k \geq (q + q \cdot r)$: **X** $k < (q + q \cdot r)$: **temporal safe**

Discussion Questions

- Contributions?
  - Do you think something like this can be deployed currently?
- Limitations?
- Thoughts on scalability?
- Thoughts on notaries impacting user privacy? They are still ‘semi-trusted’
  - Factor in proxies, DNS?
- If you really care about privacy, why not choose the PKI path (it’s worth the hassle!)