LIPKEY:
A Lower Infrastructure Public Key Mechanism

Mike Eisler
Sun Microsystems, Inc.
mre@Eng.Sun.Com

44th IETF

CAT Working Group

Priorities of the LIPKEY approach:

Familiarity breeds acceptance

- People will adopt what they are used to

On the Internet, authentication of server is more important than the client

- Who exactly am I sending my credit card number to?

There are more clients than servers

- “low infrastructure” considerations should focus minimizing client impact

Virtually every server’s operating system has native user accounting with passwords

- But different operating systems (e.g. UNIX, NT) store passwords differently

Some application protocols cannot or will not use TLS.

Familiarity breeds acceptance

A WG member wrote: “LIPKEY looks a lot like TLS with passwords”.

- That’s good, because people understand that.
- Kerberos V5 is misunderstood.
- A simple plug in to GSS-API today can “easily” be replaced with the less simple (but more versatile) when there is understanding.

Server authentication is more important on Internet

Certificate technology is operationally more secure.

There are more clients than servers

Given the state of PKI deployment (products, lack of ubiquity of smart cards and readers), forcing PKI on users to authenticate them is a non-starter.

Virtually every server's operating system has native user accounting with passwords

A security mechanism that uses existing password infrastructure is low impact.

- Requiring sites to adopt a new set of passwords is an uphill battle.
- Using UNIX password hash as the share secret from which yet another hash is calculated does not work where there is NIS.
    - Telling customers to stop storing hashes in NIS is a non-starter.
        - They'll might get around to stopping, but they don't want to do multiple transitions at once.
    - What happens when customer transitions from say UNIX to NT?

Some application protocols cannot or will not use TLS.

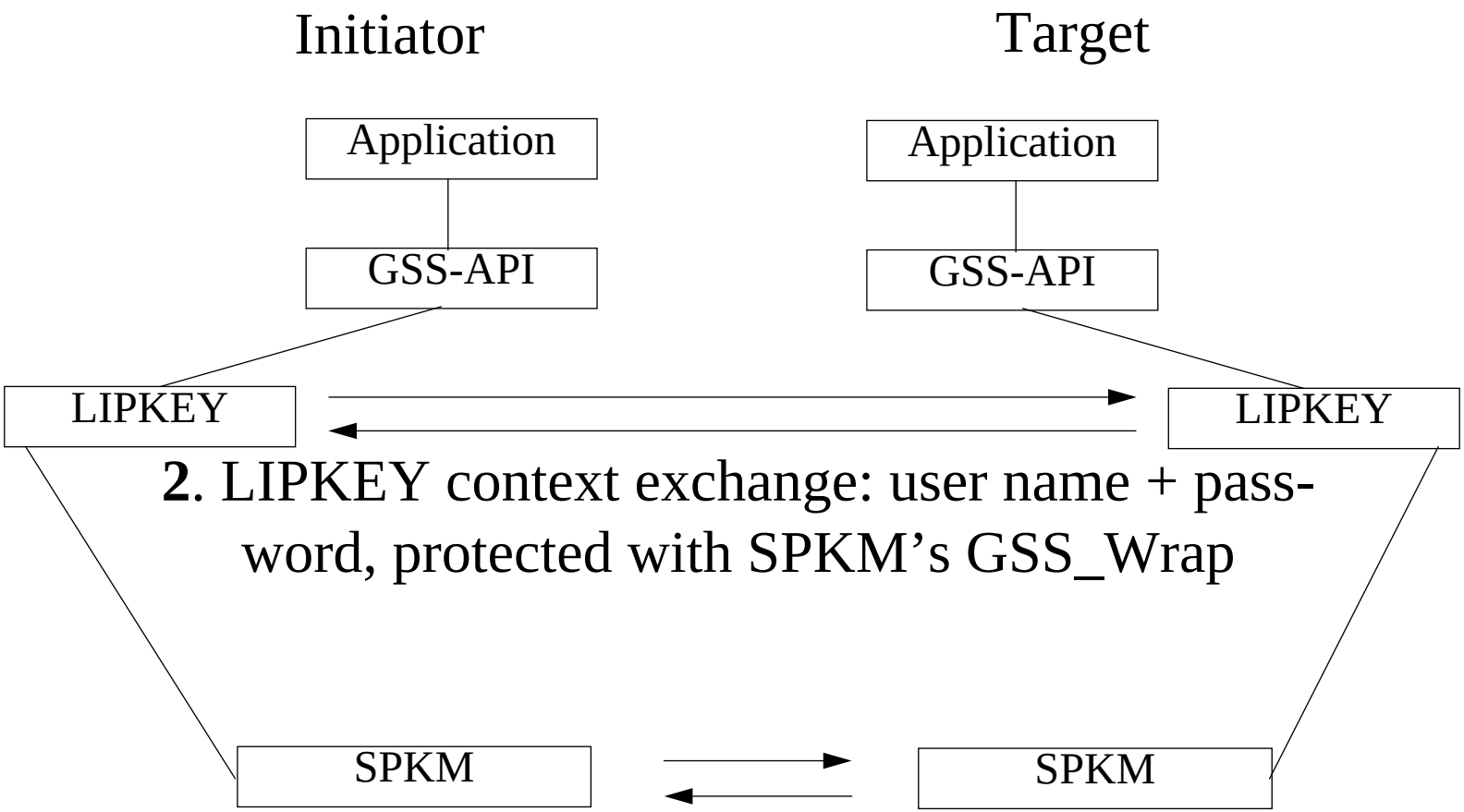
Protocols over UDP lose without something like GSS-API.

- Mandating switch to TCP is a non-starter.

Protocols like ONC RPC that have already adopted GSS-API (RFC 2203) shouldn't have to deal with multiple security architectures.

- One API, many plug ins
- SASL's use over non-TLS frameworks is perhaps not thought through yet.
    - Refer to recent GSSAPI versus GSS-SPNEGO SASL question on CAT WG alias.

Brief overview of LIPKEY



1. SPKM-1 unilateral (acceptor only) authentication context exchange. Initiator is anonymous with no certificate required.

Issues with LIPKEY

RFC 2025 (SPKM) assumes that the anonymous client can obtain (e.g. from directory service) the server's certificate to calculate a MAC on the first context token.

- Required so that the client compute a MAC on the initiator's context establishment request token.
- Considered a high infrastructure requirement.

SPKM lets the initiator request the target's certificate. Useful when the initiator knows its certificate and doesn't have a way to otherwise obtain the target's certificate.

- LIPKEY stretches the interpretation of RFC 2025 by using a "null MAC" in request token which tells the acceptor to ignore MAC.
- Acceptor will return in response token a digital signature based on its certificate.
    - signature is computed on <u>concatenation of context request and response token</u>
- Perhaps an SPKM-3 should be added to make this cleaner?

Issues raised by the working group

How can anything that has certificates be considered “low infrastructure”?

- It may not be “low”, but it’s “lower” than pure SPKM that requires a directory service for server certificates.
- Do web browser users consider IE4 or Navigator to be high infrastructure?
    - Client side infrastructure consists of pre-configured list of trusted Certificate Authorities
- Creating demand for server certificates on Intranets is a good thing.
- Semantics are irrelevant anyway.

CAT WG alias recently discussed the use of ASN.1

- The LIPKEY I-D author is agnostic. XDR works for the framing of LIPKEY tokens.
- LIPKEY uses SPKM, and does not propose to redefine SPKM to use non-ASN.1 encoding.
    - No desire to re-implement SPKM