



```
opaque verify_data[verify_data_length];  
} Finished;
```

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 74]
August 2008

A.5. The Cipher Suite

The following values define the cipher suite codes used in the ClientHello and ServerHello messages.

A cipher suite defines a cipher specification supported in TLS Version 1.2.

TLS_NULL_WITH_NULL_NULL is specified and is the initial state of a TLS connection during the first handshake on that channel, but MUST NOT be negotiated, as it provides no more protection than an unsecured connection.

```
CipherSuite TLS_NULL_WITH_NULL_NULL = { 0x00,0x00 };
```

The following CipherSuite definitions require that the server provide an RSA certificate that can be used for key exchange. The server may request any signature-capable certificate in the certificate request message.

```
CipherSuite TLS_RSA_WITH_NULL_MD5 = { 0x00,0x01 };  
CipherSuite TLS_RSA_WITH_NULL_SHA = { 0x00,0x02 };  
CipherSuite TLS_RSA_WITH_NULL_SHA256 = { 0x00,0x3B };  
CipherSuite TLS_RSA_WITH_RC4_128_MD5 = { 0x00,0x04 };  
CipherSuite TLS_RSA_WITH_RC4_128_SHA = { 0x00,0x05 };  
CipherSuite TLS_RSA_WITH_3DES_EDE_CBC_SHA = { 0x00,0x0A };  
CipherSuite TLS_RSA_WITH_AES_128_CBC_SHA = { 0x00,0x2F };  
CipherSuite TLS_RSA_WITH_AES_256_CBC_SHA = { 0x00,0x35 };  
CipherSuite TLS_RSA_WITH_AES_128_CBC_SHA256 = { 0x00,0x3C };  
CipherSuite TLS_RSA_WITH_AES_256_CBC_SHA256 = { 0x00,0x3D };
```

The following cipher suite definitions are used for server-authenticated (and optionally client-authenticated) Diffie-Hellman. DH denotes cipher suites in which the server's certificate contains the Diffie-Hellman parameters signed by the certificate authority (CA). DHE denotes ephemeral Diffie-Hellman, where the Diffie-Hellman parameters are signed by a signature-capable certificate, which has been signed by the CA. The signing algorithm used by the server is specified after the DHE component of the CipherSuite name. The server can request any signature-capable certificate from the client for client authentication, or it may request a Diffie-Hellman certificate. Any Diffie-Hellman certificate provided by the client must use the parameters (group and generator) described by the server.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 75]
August 2008

```
CipherSuite TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA = { 0x00,0x0D };  
CipherSuite TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA = { 0x00,0x10 };  
CipherSuite TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA = { 0x00,0x13 };  
CipherSuite TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA = { 0x00,0x16 };  
CipherSuite TLS_DH_DSS_WITH_AES_128_CBC_SHA = { 0x00,0x30 };  
CipherSuite TLS_DH_RSA_WITH_AES_128_CBC_SHA = { 0x00,0x31 };  
CipherSuite TLS_DHE_DSS_WITH_AES_128_CBC_SHA = { 0x00,0x32 };  
CipherSuite TLS_DHE_RSA_WITH_AES_128_CBC_SHA = { 0x00,0x33 };  
CipherSuite TLS_DH_DSS_WITH_AES_256_CBC_SHA = { 0x00,0x36 };  
CipherSuite TLS_DH_RSA_WITH_AES_256_CBC_SHA = { 0x00,0x37 };  
CipherSuite TLS_DHE_DSS_WITH_AES_256_CBC_SHA = { 0x00,0x38 };  
CipherSuite TLS_DHE_RSA_WITH_AES_256_CBC_SHA = { 0x00,0x39 };  
CipherSuite TLS_DH_DSS_WITH_AES_128_CBC_SHA256 = { 0x00,0x3E };
```



```

CipherSuite TLS_DH_DSS_WITH_AES_256_CBC_SHA256 = { 0x00,0x68 };
CipherSuite TLS_DH_RSA_WITH_AES_256_CBC_SHA256 = { 0x00,0x69 };
CipherSuite TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 = { 0x00,0x6A };
CipherSuite TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 = { 0x00,0x6B };

```

The following cipher suites are used for completely anonymous Diffie-Hellman communications in which neither party is authenticated. Note that this mode is vulnerable to man-in-the-middle attacks. Using this mode therefore is of limited use: These cipher suites MUST NOT be used by TLS 1.2 implementations unless the application layer has specifically requested to allow anonymous key exchange. (Anonymous key exchange may sometimes be acceptable, for example, to support opportunistic encryption when no set-up for authentication is in place, or when TLS is used as part of more complex security protocols that have other means to ensure authentication.)

```

CipherSuite TLS_DH_anon_WITH_RC4_128_MD5 = { 0x00,0x18 };
CipherSuite TLS_DH_anon_WITH_3DES_EDE_CBC_SHA = { 0x00,0x1B };
CipherSuite TLS_DH_anon_WITH_AES_128_CBC_SHA = { 0x00,0x34 };
CipherSuite TLS_DH_anon_WITH_AES_256_CBC_SHA = { 0x00,0x3A };
CipherSuite TLS_DH_anon_WITH_AES_128_CBC_SHA256 = { 0x00,0x6C };
CipherSuite TLS_DH_anon_WITH_AES_256_CBC_SHA256 = { 0x00,0x6D };

```

Note that using non-anonymous key exchange without actually verifying the key exchange is essentially equivalent to anonymous key exchange, and the same precautions apply. While non-anonymous key exchange will generally involve a higher computational and communicational cost than anonymous key exchange, it may be in the interest of interoperability not to disable non-anonymous key exchange when the application layer is allowing anonymous key exchange.

Dierks & Rescorla Standards Track [Page 76]
RFC 5246 TLS August 2008

New cipher suite values have been assigned by IANA as described in Section 12.

Note: The cipher suite values { 0x00, 0x1C } and { 0x00, 0x1D } are reserved to avoid collision with Fortezza-based cipher suites in SSL 3.

A.6. The Security Parameters

These security parameters are determined by the TLS Handshake Protocol and provided as parameters to the TLS record layer in order to initialize a connection state. SecurityParameters includes:

```

enum { null(0), (255) } CompressionMethod;

enum { server, client } ConnectionEnd;

enum { tls_prf_sha256 } PRFAlgorithm;

enum { null, rc4, 3des, aes } BulkCipherAlgorithm;

enum { stream, block, aead } CipherType;

enum { null, hmac_md5, hmac_sha1, hmac_sha256, hmac_sha384,
       hmac_sha512 } MACAlgorithm;

/* Other values may be added to the algorithms specified in
   CompressionMethod, PRFAlgorithm, BulkCipherAlgorithm, and
   MACAlgorithm. */

```

```

struct {
    ConnectionEnd                      entity;

```



```
uint8      enc_key_length;
uint8      block_length;
uint8      fixed_iv_length;
uint8      record_iv_length;
MACAlgorithm mac_algorithm;
uint8      mac_length;
uint8      mac_key_length;
CompressionMethod compression_algorithm;
opaque     master_secret[48];
opaque     client_random[32];
opaque     server_random[32];
} SecurityParameters;
```

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 77]
August 2008

A.7. Changes to RFC 4492

RFC 4492 [TLSECC] adds Elliptic Curve cipher suites to TLS. This document changes some of the structures used in that document. This section details the required changes for implementors of both RFC 4492 and TLS 1.2. Implementors of TLS 1.2 who are not implementing RFC 4492 do not need to read this section.

This document adds a "signature_algorithm" field to the digitally-signed element in order to identify the signature and digest algorithms used to create a signature. This change applies to digital signatures formed using ECDSA as well, thus allowing ECDSA signatures to be used with digest algorithms other than SHA-1, provided such use is compatible with the certificate and any restrictions imposed by future revisions of [PKIX].

As described in Sections 7.4.2 and 7.4.6, the restrictions on the signature algorithms used to sign certificates are no longer tied to the cipher suite (when used by the server) or the ClientCertificateType (when used by the client). Thus, the restrictions on the algorithm used to sign certificates specified in Sections 2 and 3 of RFC 4492 are also relaxed. As in this document, the restrictions on the keys in the end-entity certificate remain.

Appendix B. Glossary

Advanced Encryption Standard (AES)

AES [AES] is a widely used symmetric encryption algorithm. AES is a block cipher with a 128-, 192-, or 256-bit keys and a 16-byte block size. TLS currently only supports the 128- and 256-bit key sizes.

application protocol

An application protocol is a protocol that normally layers directly on top of the transport layer (e.g., TCP/IP). Examples include HTTP, TELNET, FTP, and SMTP.

asymmetric cipher

See public key cryptography.

authenticated encryption with additional data (AEAD)

A symmetric encryption algorithm that simultaneously provides confidentiality and message integrity.

authentication

Authentication is the ability of one entity to determine the identity of another entity.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 78]
August 2008



common block size.

bulk cipher

A symmetric encryption algorithm used to encrypt large quantities of data.

cipher block chaining (CBC)

CBC is a mode in which every plaintext block encrypted with a block cipher is first exclusive-ORed with the previous ciphertext block (or, in the case of the first block, with the initialization vector). For decryption, every block is first decrypted, then exclusive-ORed with the previous ciphertext block (or IV).

certificate

As part of the X.509 protocol (a.k.a. ISO Authentication framework), certificates are assigned by a trusted Certificate Authority and provide a strong binding between a party's identity or some other attributes and its public key.

client

The application entity that initiates a TLS connection to a server. This may or may not imply that the client initiated the underlying transport connection. The primary operational difference between the server and client is that the server is generally authenticated, while the client is only optionally authenticated.

client write key

The key used to encrypt data written by the client.

client write MAC key

The secret data used to authenticate data written by the client.

connection

A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For TLS, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.

Data Encryption Standard

DES [DES] still is a very widely used symmetric encryption algorithm although it is considered as rather weak now. DES is a block cipher with a 56-bit key and an 8-byte block size. Note that in TLS, for key generation purposes, DES is treated as having an 8-byte key length (64 bits), but it still only provides 56 bits

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 79]
August 2008

of protection. (The low bit of each key byte is presumed to be set to produce odd parity in that key byte.) DES can also be operated in a mode [3DES] where three independent keys and three encryptions are used for each block of data; this uses 168 bits of key (24 bytes in the TLS key generation method) and provides the equivalent of 112 bits of security.

Digital Signature Standard (DSS)

A standard for digital signing, including the Digital Signing Algorithm, approved by the National Institute of Standards and Technology, defined in NIST FIPS PUB 186-2, "Digital Signature Standard", published January 2000 by the U.S. Department of Commerce [DSS]. A significant update [DSS-3] has been drafted and was published in March 2006.

digital signatures

Digital signatures utilize public key cryptography and one-way hash functions to produce a signature of the data that can be



establishes the parameters of their transactions.

Initialization Vector (IV)

When a block cipher is used in CBC mode, the initialization vector is exclusive-ORed with the first plaintext block prior to encryption.

Message Authentication Code (MAC)

A Message Authentication Code is a one-way hash computed from a message and some secret data. It is difficult to forge without knowing the secret data. Its purpose is to detect if the message has been altered.

master secret

Secure secret data used for generating encryption keys, MAC secrets, and IVs.

MD5

MD5 [MD5] is a hashing function that converts an arbitrarily long data stream into a hash of fixed size (16 bytes). Due to significant progress in cryptanalysis, at the time of publication of this document, MD5 no longer can be considered a 'secure' hashing function.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 80]
August 2008

public key cryptography

A class of cryptographic techniques employing two-key ciphers. Messages encrypted with the public key can only be decrypted with the associated private key. Conversely, messages signed with the private key can be verified with the public key.

one-way hash function

A one-way transformation that converts an arbitrary amount of data into a fixed-length hash. It is computationally hard to reverse the transformation or to find collisions. MD5 and SHA are examples of one-way hash functions.

RC4

A stream cipher invented by Ron Rivest. A compatible cipher is described in [SCH].

RSA

A very widely used public key algorithm that can be used for either encryption or digital signing. [RSA]

server

The server is the application entity that responds to requests for connections from clients. See also "client".

session

A TLS session is an association between a client and a server. Sessions are created by the handshake protocol. Sessions define a set of cryptographic security parameters that can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

session identifier

A session identifier is a value generated by a server that identifies a particular session.

server write key

The key used to encrypt data written by the server.

server write MAC key

The secret data used to authenticate data written by the server.



produces a 20-byte output. Note that all references to SHA (without a numerical suffix) actually use the modified SHA-1 algorithm.

Dierks & Rescorla Standards Track [Page 81]
RFC 5246 TLS August 2008

SHA-256

The 256-bit Secure Hash Algorithm is defined in FIPS PUB 180-2. It produces a 32-byte output.

SSL

Netscape's Secure Socket Layer protocol [SSL3]. TLS is based on SSL Version 3.0.

stream cipher

An encryption algorithm that converts a key into a cryptographically strong keystream, which is then exclusive-ORed with the plaintext.

symmetric cipher

See bulk cipher.

Transport Layer Security (TLS)

This protocol; also, the Transport Layer Security working group of the Internet Engineering Task Force (IETF). See "Working Group Information" at the end of this document (see page 99).

Dierks & Rescorla Standards Track [Page 82]
RFC 5246 TLS August 2008

Appendix C. Cipher Suite Definitions

Cipher Suite	Key Exchange	Cipher	Mac
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA
TLS_RSA_WITH_NULL_SHA256	RSA	NULL	SHA256
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4_128	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4_128	SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	AES_128_CBC	SHA
TLS_RSA_WITH_AES_256_CBC_SHA	RSA	AES_256_CBC	SHA
TLS_RSA_WITH_AES_128_CBC_SHA256	RSA	AES_128_CBC	SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256	RSA	AES_256_CBC	SHA256
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE_CBC	SHA
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE_CBC	SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES_EDE_CBC	SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE_CBC	SHA
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4_128	MD5
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES_EDE_CBC	SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA	DH_DSS	AES_128_CBC	SHA
TLS_DH_RSA_WITH_AES_128_CBC_SHA	DH_RSA	AES_128_CBC	SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	DHE_DSS	AES_128_CBC	SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	DHE_RSA	AES_128_CBC	SHA
TLS_DH_anon_WITH_AES_128_CBC_SHA	DH_anon	AES_128_CBC	SHA
TLS_DH_DSS_WITH_AES_256_CBC_SHA	DH_DSS	AES_256_CBC	SHA
TLS_DH_RSA_WITH_AES_256_CBC_SHA	DH_RSA	AES_256_CBC	SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	DHE_DSS	AES_256_CBC	SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	DHE_RSA	AES_256_CBC	SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA	DH_anon	AES_256_CBC	SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	DH_DSS	AES_128_CBC	SHA256
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	DH_RSA	AES_128_CBC	SHA256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	DHE_DSS	AES_128_CBC	SHA256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	DHE_RSA	AES_128_CBC	SHA256



TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	DHE_DSS	AES_256_CBC	SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	DHE_RSA	AES_256_CBC	SHA256
TLS_DH_anon_WITH_AES_256_CBC_SHA256	DH_anon	AES_256_CBC	SHA256

Dierks & Rescorla	Standards Track	[Page 83]
RFC 5246	TLS	August 2008

Cipher	Type	Key Material	IV Size	Block Size
NULL	Stream	0	0	N/A
RC4_128	Stream	16	0	N/A
3DES_EDE_CBC	Block	24	8	8
AES_128_CBC	Block	16	16	16
AES_256_CBC	Block	32	16	16

MAC	Algorithm	mac_length	mac_key_length
NULL	N/A	0	0
MD5	HMAC-MD5	16	16
SHA	HMAC-SHA1	20	20
SHA256	HMAC-SHA256	32	32

Type

Indicates whether this is a stream cipher or a block cipher running in CBC mode.

Key Material

The number of bytes from the key_block that are used for generating the write keys.

IV Size

The amount of data needed to be generated for the initialization vector. Zero for stream ciphers; equal to the block size for block ciphers (this is equal to SecurityParameters.record_iv_length).

Block Size

The amount of data a block cipher enciphers in one chunk; a block cipher running in CBC mode can only encrypt an even multiple of its block size.

Dierks & Rescorla	Standards Track	[Page 84]
RFC 5246	TLS	August 2008

Appendix D. Implementation Notes

The TLS protocol cannot prevent many common security mistakes. This section provides several recommendations to assist implementors.

D.1. Random Number Generation and Seeding

TLS requires a cryptographically secure pseudorandom number generator (PRNG). Care must be taken in designing and seeding PRNGs. PRNGs based on secure hash operations, most notably SHA-1, are acceptable, but cannot provide more security than the size of the random number generator state.

To estimate the amount of seed material being produced, add the number of bits of unpredictable information in each seed byte. For example, keystroke timing values taken from a PC compatible's 18.2 Hz timer provide 1 or 2 secure bits each, even though the total size of the counter value is 16 bits or more. Seeding a 128-bit PRNG would thus require approximately 100 such timer values.

[RANDOM] provides guidance on the generation of random values.



certificates and should generally support certificate revocation messages. Certificates should always be verified to ensure proper signing by a trusted Certificate Authority (CA). The selection and addition of trusted CAs should be done very carefully. Users should be able to view information about the certificate and root CA.

D.3. Cipher Suites

TLS supports a range of key sizes and security levels, including some that provide no or minimal security. A proper implementation will probably not support many cipher suites. For instance, anonymous Diffie-Hellman is strongly discouraged because it cannot prevent man-in-the-middle attacks. Applications should also enforce minimum and maximum key sizes. For example, certificate chains containing 512-bit RSA keys or signatures are not appropriate for high-security applications.

D.4. Implementation Pitfalls

Implementation experience has shown that certain parts of earlier TLS specifications are not easy to understand, and have been a source of interoperability and security problems. Many of these areas have

Dierks & Rescorla Standards Track [Page 85]
RFC 5246 TLS August 2008

been clarified in this document, but this appendix contains a short list of the most important things that require special attention from implementors.

TLS protocol issues:

- Do you correctly handle handshake messages that are fragmented to multiple TLS records (see Section 6.2.1)? Including corner cases like a ClientHello that is split to several small fragments? Do you fragment handshake messages that exceed the maximum fragment size? In particular, the certificate and certificate request handshake messages can be large enough to require fragmentation.
- Do you ignore the TLS record layer version number in all TLS records before ServerHello (see Appendix E.1)?
- Do you handle TLS extensions in ClientHello correctly, including omitting the extensions field completely?
- Do you support renegotiation, both client and server initiated? While renegotiation is an optional feature, supporting it is highly recommended.
- When the server has requested a client certificate, but no suitable certificate is available, do you correctly send an empty Certificate message, instead of omitting the whole message (see Section 7.4.6)?

Cryptographic details:

- In the RSA-encrypted Premaster Secret, do you correctly send and verify the version number? When an error is encountered, do you continue the handshake to avoid the Bleichenbacher attack (see Section 7.4.7.1)?
- What countermeasures do you use to prevent timing attacks against RSA decryption and signing operations (see Section 7.4.7.1)?
- When verifying RSA signatures, do you accept both NULL and missing parameters (see Section 4.7)? Do you verify that the RSA padding doesn't have additional data after the hash value? [FI06]

- Does your TLS client check that the Diffie-Hellman parameters sent by the server are acceptable (see Section F.1.1.3)?

Dierks & Rescorla Standards Track [Page 86]
RFC 5246 TLS August 2008

- How do you generate unpredictable IVs for CBC mode ciphers (see Section 6.2.3.2)?
- Do you accept long CBC mode padding (up to 255 bytes; see Section 6.2.3.2)?
- How do you address CBC mode timing attacks (Section 6.2.3.2)?
- Do you use a strong and, most importantly, properly seeded random number generator (see Appendix D.1) for generating the premaster secret (for RSA key exchange), Diffie-Hellman private values, the DSA "k" parameter, and other security-critical values?

Appendix E. Backward Compatibility

E.1. Compatibility with TLS 1.0/1.1 and SSL 3.0

Since there are various versions of TLS (1.0, 1.1, 1.2, and any future versions) and SSL (2.0 and 3.0), means are needed to negotiate the specific protocol version to use. The TLS protocol provides a built-in mechanism for version negotiation so as not to bother other protocol components with the complexities of version selection.

TLS versions 1.0, 1.1, and 1.2, and SSL 3.0 are very similar, and use compatible ClientHello messages; thus, supporting all of them is relatively easy. Similarly, servers can easily handle clients trying to use future versions of TLS as long as the ClientHello format remains compatible, and the client supports the highest protocol version available in the server.

A TLS 1.2 client who wishes to negotiate with such older servers will send a normal TLS 1.2 ClientHello, containing { 3, 3 } (TLS 1.2) in ClientHello.client_version. If the server does not support this version, it will respond with a ServerHello containing an older version number. If the client agrees to use this version, the negotiation will proceed as appropriate for the negotiated protocol.

If the version chosen by the server is not supported by the client (or not acceptable), the client MUST send a "protocol_version" alert message and close the connection.

If a TLS server receives a ClientHello containing a version number greater than the highest version supported by the server, it MUST reply according to the highest version supported by the server.

A TLS server can also receive a ClientHello containing a version number smaller than the highest supported version. If the server wishes to negotiate with old clients, it will proceed as appropriate

Dierks & Rescorla Standards Track [Page 87]
RFC 5246 TLS August 2008

for the highest version supported by the server that is not greater than ClientHello.client_version. For example, if the server supports TLS 1.0, 1.1, and 1.2, and client_version is TLS 1.0, the server will proceed with a TLS 1.0 ServerHello. If server supports (or is willing to use) only versions greater than client_version, it MUST send a "protocol_version" alert message and close the connection.

Whenever a client already knows the highest protocol version known to



Note: some server implementations are known to implement version negotiation incorrectly. For example, there are buggy TLS 1.0 servers that simply close the connection when the client offers a version newer than TLS 1.0. Also, it is known that some servers will refuse the connection if any TLS extensions are included in ClientHello. Interoperability with such buggy servers is a complex topic beyond the scope of this document, and may require multiple connection attempts by the client.

Earlier versions of the TLS specification were not fully clear on what the record layer version number (TLSPlaintext.version) should contain when sending ClientHello (i.e., before it is known which version of the protocol will be employed). Thus, TLS servers compliant with this specification MUST accept any value {03,XX} as the record layer version number for ClientHello.

TLS clients that wish to negotiate with older servers MAY send any value {03,XX} as the record layer version number. Typical values would be {03,00}, the lowest version number supported by the client, and the value of ClientHello.client_version. No single value will guarantee interoperability with all old servers, but this is a complex topic beyond the scope of this document.

E.2. Compatibility with SSL 2.0

TLS 1.2 clients that wish to support SSL 2.0 servers MUST send version 2.0 CLIENT-HELLO messages defined in [SSL2]. The message MUST contain the same version number as would be used for ordinary ClientHello, and MUST encode the supported TLS cipher suites in the CIPHER-SPECS-DATA field as described below.

Warning: The ability to send version 2.0 CLIENT-HELLO messages will be phased out with all due haste, since the newer ClientHello format provides better mechanisms for moving to newer versions and negotiating extensions. TLS 1.2 clients SHOULD NOT support SSL 2.0.

Dierks & Rescorla Standards Track [Page 88]
RFC 5246 TLS August 2008

However, even TLS servers that do not support SSL 2.0 MAY accept version 2.0 CLIENT-HELLO messages. The message is presented below in sufficient detail for TLS server implementors; the true definition is still assumed to be [SSL2].

For negotiation purposes, 2.0 CLIENT-HELLO is interpreted the same way as a ClientHello with a "null" compression method and no extensions. Note that this message MUST be sent directly on the wire, not wrapped as a TLS record. For the purposes of calculating Finished and CertificateVerify, the msg_length field is not considered to be a part of the handshake message.

```
uint8 V2CipherSpec[3];
struct {
    uint16 msg_length;
    uint8 msg_type;
    Version version;
    uint16 cipher_spec_length;
    uint16 session_id_length;
    uint16 challenge_length;
    V2CipherSpec cipher_specs[V2ClientHello.cipher_spec_length];
    opaque session_id[V2ClientHello.session_id_length];
    opaque challenge[V2ClientHello.challenge_length];
} V2ClientHello;
```

msg_length

The highest bit MUST be 1; the remaining bits contain the length of the following data in bytes.



version 2 ClientHello message. The value MUST be 1.

version

Equal to ClientHello.client_version.

cipher_spec_length

This field is the total length of the field cipher_specs. It cannot be zero and MUST be a multiple of the V2CipherSpec length (3).

session_id_length

This field MUST have a value of zero for a client that claims to support TLS 1.2.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 89]
August 2008

challenge_length

The length in bytes of the client's challenge to the server to authenticate itself. Historically, permissible values are between 16 and 32 bytes inclusive. When using the SSLv2 backward-compatible handshake the client SHOULD use a 32-byte challenge.

cipher_specs

This is a list of all CipherSpecs the client is willing and able to use. In addition to the 2.0 cipher specs defined in [SSL2], this includes the TLS cipher suites normally sent in ClientHello.cipher_suites, with each cipher suite prefixed by a zero byte. For example, the TLS cipher suite {0x00,0x0A} would be sent as {0x00,0x00,0x0A}.

session_id

This field MUST be empty.

challenge

Corresponds to ClientHello.random. If the challenge length is less than 32, the TLS server will pad the data with leading (note: not trailing) zero bytes to make it 32 bytes long.

Note: Requests to resume a TLS session MUST use a TLS client hello.

E.3. Avoiding Man-in-the-Middle Version Rollback

When TLS clients fall back to Version 2.0 compatibility mode, they MUST use special PKCS#1 block formatting. This is done so that TLS servers will reject Version 2.0 sessions with TLS-capable clients.

When a client negotiates SSL 2.0 but also supports TLS, it MUST set the right-hand (least-significant) 8 random bytes of the PKCS padding (not including the terminal null of the padding) for the RSA encryption of the ENCRYPTED-KEY-DATA field of the CLIENT-MASTER-KEY to 0x03 (the other padding bytes are random).

When a TLS-capable server negotiates SSL 2.0 it SHOULD, after decrypting the ENCRYPTED-KEY-DATA field, check that these 8 padding bytes are 0x03. If they are not, the server SHOULD generate a random value for SECRET-KEY-DATA, and continue the handshake (which will eventually fail since the keys will not match). Note that reporting the error situation to the client could make the server vulnerable to attacks described in [BLEI].

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 90]
August 2008

Appendix F. Security Analysis

The TLS protocol is designed to establish a secure connection between



secret information from sources outside the protocol. Attackers are assumed to have the ability to capture, modify, delete, replay, and otherwise tamper with messages sent over the communication channel. This appendix outlines how TLS has been designed to resist a variety of attacks.

F.1. Handshake Protocol

The handshake protocol is responsible for selecting a cipher spec and generating a master secret, which together comprise the primary cryptographic parameters associated with a secure session. The handshake protocol can also optionally authenticate parties who have certificates signed by a trusted certificate authority.

F.1.1. Authentication and Key Exchange

TLS supports three authentication modes: authentication of both parties, server authentication with an unauthenticated client, and total anonymity. Whenever the server is authenticated, the channel is secure against man-in-the-middle attacks, but completely anonymous sessions are inherently vulnerable to such attacks. Anonymous servers cannot authenticate clients. If the server is authenticated, its certificate message must provide a valid certificate chain leading to an acceptable certificate authority. Similarly, authenticated clients must supply an acceptable certificate to the server. Each party is responsible for verifying that the other's certificate is valid and has not expired or been revoked.

The general goal of the key exchange process is to create a `pre_master_secret` known to the communicating parties and not to attackers. The `pre_master_secret` will be used to generate the `master_secret` (see Section 8.1). The `master_secret` is required to generate the Finished messages, encryption keys, and MAC keys (see Sections 7.4.9 and 6.3). By sending a correct Finished message, parties thus prove that they know the correct `pre_master_secret`.

F.1.1.1. Anonymous Key Exchange

Completely anonymous sessions can be established using Diffie-Hellman for key exchange. The server's public parameters are contained in the server key exchange message, and the client's are sent in the

Dierks & Rescorla Standards Track [Page 91]
RFC 5246 TLS August 2008

client key exchange message. Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e., the `pre_master_secret`).

Warning: Completely anonymous connections only provide protection against passive eavesdropping. Unless an independent tamper-proof channel is used to verify that the Finished messages were not replaced by an attacker, server authentication is required in environments where active man-in-the-middle attacks are a concern.

F.1.1.2. RSA Key Exchange and Authentication

With RSA, key exchange and server authentication are combined. The public key is contained in the server's certificate. Note that compromise of the server's static RSA key results in a loss of confidentiality for all sessions protected under that static key. TLS users desiring Perfect Forward Secrecy should use DHE cipher suites. The damage done by exposure of a private key can be limited by changing one's private key (and certificate) frequently.

After verifying the server's certificate, the client encrypts a `pre_master_secret` with the server's public key. By successfully

When RSA is used for key exchange, clients are authenticated using the certificate verify message (see Section 7.4.8). The client signs a value derived from all preceding handshake messages. These handshake messages include the server certificate, which binds the signature to the server, and ServerHello.random, which binds the signature to the current handshake process.

F.1.1.3. Diffie-Hellman Key Exchange with Authentication

When Diffie-Hellman key exchange is used, the server can either supply a certificate containing fixed Diffie-Hellman parameters or use the server key exchange message to send a set of temporary Diffie-Hellman parameters signed with a DSA or RSA certificate. Temporary parameters are hashed with the hello.random values before signing to ensure that attackers do not replay old parameters. In either case, the client can verify the certificate or signature to ensure that the parameters belong to the server.

If the client has a certificate containing fixed Diffie-Hellman parameters, its certificate contains the information required to complete the key exchange. Note that in this case the client and server will generate the same Diffie-Hellman result (i.e.,

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 92]
August 2008

pre_master_secret) every time they communicate. To prevent the pre_master_secret from staying in memory any longer than necessary, it should be converted into the master_secret as soon as possible. Client Diffie-Hellman parameters must be compatible with those supplied by the server for the key exchange to work.

If the client has a standard DSA or RSA certificate or is unauthenticated, it sends a set of temporary parameters to the server in the client key exchange message, then optionally uses a certificate verify message to authenticate itself.

If the same DH keypair is to be used for multiple handshakes, either because the client or server has a certificate containing a fixed DH keypair or because the server is reusing DH keys, care must be taken to prevent small subgroup attacks. Implementations SHOULD follow the guidelines found in [SUBGROUP].

Small subgroup attacks are most easily avoided by using one of the DHE cipher suites and generating a fresh DH private key (X) for each handshake. If a suitable base (such as 2) is chosen, $g^X \bmod p$ can be computed very quickly; therefore, the performance cost is minimized. Additionally, using a fresh key for each handshake provides Perfect Forward Secrecy. Implementations SHOULD generate a new X for each handshake when using DHE cipher suites.

Because TLS allows the server to provide arbitrary DH groups, the client should verify that the DH group is of suitable size as defined by local policy. The client SHOULD also verify that the DH public exponent appears to be of adequate size. [KEYSIZ] provides a useful guide to the strength of various group sizes. The server MAY choose to assist the client by providing a known group, such as those defined in [IKEALG] or [MODP]. These can be verified by simple comparison.

F.1.2. Version Rollback Attacks

Because TLS includes substantial improvements over SSL Version 2.0, attackers may try to make TLS-capable clients and servers fall back to Version 2.0. This attack can occur if (and only if) two TLS-capable parties use an SSL 2.0 handshake.



3.0 servers to detect the attack. This solution is not secure against attackers who can brute-force the key and substitute a new ENCRYPTED-KEY-DATA message containing the same key (but with normal padding) before the application-specified wait threshold has expired. Altering the padding of the least-significant 8 bytes of the PKCS

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 93]
August 2008

padding does not impact security for the size of the signed hashes and RSA key lengths used in the protocol, since this is essentially equivalent to increasing the input block size by 8 bytes.

F.1.3. Detecting Attacks Against the Handshake Protocol

An attacker might try to influence the handshake exchange to make the parties select different encryption algorithms than they would normally choose.

For this attack, an attacker must actively change one or more handshake messages. If this occurs, the client and server will compute different values for the handshake message hashes. As a result, the parties will not accept each others' Finished messages. Without the master_secret, the attacker cannot repair the Finished messages, so the attack will be discovered.

F.1.4. Resuming Sessions

When a connection is established by resuming a session, new ClientHello.random and ServerHello.random values are hashed with the session's master_secret. Provided that the master_secret has not been compromised and that the secure hash operations used to produce the encryption keys and MAC keys are secure, the connection should be secure and effectively independent from previous connections. Attackers cannot use known encryption keys or MAC secrets to compromise the master_secret without breaking the secure hash operations.

Sessions cannot be resumed unless both the client and server agree. If either party suspects that the session may have been compromised, or that certificates may have expired or been revoked, it should force a full handshake. An upper limit of 24 hours is suggested for session ID lifetimes, since an attacker who obtains a master_secret may be able to impersonate the compromised party until the corresponding session ID is retired. Applications that may be run in relatively insecure environments should not write session IDs to stable storage.

F.2. Protecting Application Data

The master_secret is hashed with the ClientHello.random and ServerHello.random to produce unique data encryption keys and MAC secrets for each connection.

Outgoing data is protected with a MAC before transmission. To prevent message replay or modification attacks, the MAC is computed from the MAC key, the sequence number, the message length, the

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 94]
August 2008

message contents, and two fixed character strings. The message type field is necessary to ensure that messages intended for one TLS record layer client are not redirected to another. The sequence number ensures that attempts to delete or reorder messages will be detected. Since sequence numbers are 64 bits long, they should never overflow. Messages from one party cannot be inserted into the



If an attacker does break an encryption key, all messages encrypted with it can be read. Similarly, compromise of a MAC key can make message-modification attacks possible. Because MACs are also encrypted, message-alteration attacks generally require breaking the encryption algorithm as well as the MAC.

Note: MAC keys may be larger than encryption keys, so messages can remain tamper resistant even if encryption keys are broken.

F.3. Explicit IVs

[CBCATT] describes a chosen plaintext attack on TLS that depends on knowing the IV for a record. Previous versions of TLS [TLS1.0] used the CBC residue of the previous record as the IV and therefore enabled this attack. This version uses an explicit IV in order to protect against this attack.

F.4. Security of Composite Cipher Modes

TLS secures transmitted application data via the use of symmetric encryption and authentication functions defined in the negotiated cipher suite. The objective is to protect both the integrity and confidentiality of the transmitted data from malicious actions by active attackers in the network. It turns out that the order in which encryption and authentication functions are applied to the data plays an important role for achieving this goal [ENCAUTH].

The most robust method, called encrypt-then-authenticate, first applies encryption to the data and then applies a MAC to the ciphertext. This method ensures that the integrity and confidentiality goals are obtained with ANY pair of encryption and MAC functions, provided that the former is secure against chosen plaintext attacks and that the MAC is secure against chosen-message attacks. TLS uses another method, called authenticate-then-encrypt, in which first a MAC is computed on the plaintext and then the concatenation of plaintext and MAC is encrypted. This method has been proven secure for CERTAIN combinations of encryption functions and MAC functions, but it is not guaranteed to be secure in general.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 95]
August 2008

In particular, it has been shown that there exist perfectly secure encryption functions (secure even in the information-theoretic sense) that combined with any secure MAC function, fail to provide the confidentiality goal against an active attack. Therefore, new cipher suites and operation modes adopted into TLS need to be analyzed under the authenticate-then-encrypt method to verify that they achieve the stated integrity and confidentiality goals.

Currently, the security of the authenticate-then-encrypt method has been proven for some important cases. One is the case of stream ciphers in which a computationally unpredictable pad of the length of the message, plus the length of the MAC tag, is produced using a pseudorandom generator and this pad is exclusive-ORed with the concatenation of plaintext and MAC tag. The other is the case of CBC mode using a secure block cipher. In this case, security can be shown if one applies one CBC encryption pass to the concatenation of plaintext and MAC and uses a new, independent, and unpredictable IV for each new pair of plaintext and MAC. In versions of TLS prior to 1.1, CBC mode was used properly EXCEPT that it used a predictable IV in the form of the last block of the previous ciphertext. This made TLS open to chosen plaintext attacks. This version of the protocol is immune to those attacks. For exact details in the encryption modes proven secure, see [ENCAUTH].



In particular, an attacker who initiates a large number of TCP connections can cause a server to consume large amounts of CPU for doing RSA decryption. However, because TLS is generally used over TCP, it is difficult for the attacker to hide his point of origin if proper TCP SYN randomization is used [SEQNUM] by the TCP stack.

Because TLS runs over TCP, it is also susceptible to a number of DoS attacks on individual connections. In particular, attackers can forge RSTs, thereby terminating connections, or forge partial TLS records, thereby causing the connection to stall. These attacks cannot in general be defended against by a TCP-using protocol. Implementors or users who are concerned with this class of attack should use IPsec AH [AH] or ESP [ESP].

F.6. Final Notes

For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.

Dierks & Rescorla Standards Track [Page 96]
RFC 5246 TLS August 2008

The system is only as strong as the weakest key exchange and authentication algorithm supported, and only trustworthy cryptographic functions should be used. Short public keys and anonymous servers should be used with great caution. Implementations and users must be careful when deciding which certificates and certificate authorities are acceptable; a dishonest certificate authority can do tremendous damage.

Normative References

- [AES] National Institute of Standards and Technology,
 "Specification for the Advanced Encryption Standard (AES)"
 FIPS 197. November 26, 2001.
- [3DES] National Institute of Standards and Technology,
 "Recommendation for the Triple Data Encryption Algorithm
 (TDEA) Block Cipher", NIST Special Publication 800-67, May
 2004.
- [DSS] NIST FIPS PUB 186-2, "Digital Signature Standard",
 National Institute of Standards and Technology, U.S.
 Department of Commerce, 2000.
- [HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-
 Hashing for Message Authentication", RFC 2104, February
 1997.
- [MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321,
 April 1992.
- [PKCS1] Jonsson, J. and B. Kaliski, "Public-Key Cryptography
 Standards (PKCS) #1: RSA Cryptography Specifications
 Version 2.1", RFC 3447, February 2003.
- [PKIX] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet
 X.509 Public Key Infrastructure Certificate and
 Certificate Revocation List (CRL) Profile", RFC 3280,
 April 2002.
- [SCH] B. Schneier. "Applied Cryptography: Protocols, Algorithms,
 and Source Code in C, 2nd ed.", Published by John Wiley &
 Sons, Inc. 1996.
- [SHS] NIST FIPS PUB 180-2, "Secure Hash Standard", National



Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 97]
August 2008

- [REQ] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [X680] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.
- [X690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002, Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

Informative References

- [AEAD] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.
- [AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005.
- [BLEI] Bleichenbacher D., "Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1" in Advances in Cryptology -- CRYPTO'98, LNCS vol. 1462, pages: 1-12, 1998.
- [CBCATT] Moeller, B., "Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures", <http://www.openssl.org/~bodo/tls-cbc.txt>.
- [CBCTIME] Canvel, B., Hiltgen, A., Vaudenay, S., and M. Vuagnoux, "Password Interception in a SSL/TLS Channel", Advances in Cryptology -- CRYPTO 2003, LNCS vol. 2729, 2003.
- [CCM] "NIST Special Publication 800-38C: The CCM Mode for Authentication and Confidentiality", <http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf>
- [DES] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, October 1999.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 98]
August 2008

- [DSS-3] NIST FIPS PUB 186-3 Draft, "Digital Signature Standard", National Institute of Standards and Technology, U.S. Department of Commerce, 2006.
- [ECDSA] American National Standards Institute, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANS X9.62-2005, November 2005.
- [ENCAUTH] Krawczyk, H., "The Order of Encryption and Authentication for Protecting Communications (Or: How Secure is SSL?)", Crypto 2001.
- [ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.



list, 27 August 2006, <http://www.imc.org/ietf-openpgp/mail-archive/msg14307.html>.

- [GCM] Dworkin, M., NIST Special Publication 800-38D, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", November 2007.
- [IKEALG] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, December 2005.
- [KEYSIZ] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.
- [KPR03] Klima, V., Pokorny, O., Rosa, T., "Attacking RSA-based Sessions in SSL/TLS", <http://eprint.iacr.org/2003/052/>, March 2003.
- [MODP] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.
- [PKCS6] RSA Laboratories, "PKCS #6: RSA Extended Certificate Syntax Standard", version 1.5, November 1993.
- [PKCS7] RSA Laboratories, "PKCS #7: RSA Cryptographic Message Syntax Standard", version 1.5, November 1993.

Dierks & Rescorla
RFC 5246

Standards Track
TLS

[Page 99]
August 2008

- [RANDOM] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC3749] Hollenbeck, S., "Transport Layer Security Protocol Compression Methods", RFC 3749, May 2004.
- [RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.
- [RSA] R. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, v. 21, n. 2, Feb 1978, pp. 120-126.
- [SEQNUM] Bellare, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.
- [SSL2] Hickman, Kipp, "The SSL Protocol", Netscape Communications Corp., Feb 9, 1995.
- [SSL3] A. Freier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.
- [SUBGROUP] Zuccherato, R., "Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME", RFC 2785, March 2000.
- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [TIMING] Boneh, D., Brumley, D., "Remote timing attacks are practical", USENIX Security Symposium 2003.



[TLSECC] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[TLSEXT] Eastlake, D., 3rd, "Transport Layer Security (TLS) Extensions: Extension Definitions", Work in Progress, February 2008.

Dierks & Rescorla Standards Track [Page 100]
RFC 5246 TLS August 2008

[TLSPGP] Mavrogiannopoulos, N., "Using OpenPGP Keys for Transport Layer Security (TLS) Authentication", RFC 5081, November 2007.

[TLSPSK] Eronen, P., Ed., and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[TLS1.0] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[TLS1.1] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[X501] ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models, 1993.

[XDR] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006.

Working Group Information

The discussion list for the IETF TLS working group is located at the e-mail address <tls@ietf.org>. Information on the group and information on how to subscribe to the list is at <<https://www1.ietf.org/mailman/listinfo/tls>>

Archives of the list can be found at:
<<http://www.ietf.org/mail-archive/web/tls/current/index.html>>

Contributors

Christopher Allen (co-editor of TLS 1.0)
Alacrity Ventures
ChristopherA@AlacrityManagement.com

Martin Abadi
University of California, Santa Cruz
abadi@cs.ucsc.edu

Steven M. Bellovin
Columbia University
smb@cs.columbia.edu

Simon Blake-Wilson
BCI
sblakewilson@bcisse.com

Dierks & Rescorla Standards Track [Page 101]
RFC 5246 TLS August 2008

Ran Canetti
IBM
canetti@watson.ibm.com



Taher Elgamal
taher@securify.com
Securify

Pasi Eronen
pasi.eronen@nokia.com
Nokia

Anil Gangolli
anil@busybuddha.org

Kipp Hickman

Alfred Hoenes

David Hopwood
Independent Consultant
david.hopwood@blueyonder.co.uk

Phil Karlton (co-author of SSLv3)

Paul Kocher (co-author of SSLv3)
Cryptography Research
paul@cryptography.com

Hugo Krawczyk
IBM
hugo@ee.technion.ac.il

Jan Mikkelsen
Transactionware
janm@transactionware.com

Magnus Nystrom
RSA Security
magnus@rsasecurity.com

Robert Relyea
Netscape Communications
relyea@netscape.com

Dierks & Rescorla Standards Track
RFC 5246 TLS

[Page 102]
August 2008

Jim Roskind
Netscape Communications
jar@netscape.com

Michael Sabin

Dan Simon
Microsoft, Inc.
dansimon@microsoft.com

Tom Weinstein

Tim Wright
Vodafone
timothy.wright@vodafone.com

Editors' Addresses

Tim Dierks
Independent
EMail: tim@dierks.org



Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

EXHIBIT B

Working Groups

Applications
Internet
Ops & Mgmt
RAI
Routing
Security
Transport
Active WGs
Concluded WGs
Non-WG Lists

Drafts & RFCs

Search

Submit a draft

Meetings

Agenda
Materials
Past Proceedings
Upcoming

Other Documents

IPR Disclosures
Liaison Statements
IESG Agenda

Related Sites

Main IETF site
ETF tools
IAB
RFC Editor
IASA/IAOC/Trust
IANA
IRTF

Version 3.12, 2011-02-02

The Transport Layer Security (TLS) Protocol Version 1.2 RFC 5246

Document IESG Evaluation Record IESG Writeups History

Document type: RFC - Proposed Standard (IETF Stream)
Updated by [RFC 5746](#), [RFC 5878](#)
Obsoletes [RFC 3268](#), [RFC 4346](#), [RFC 4366](#)
Updates [RFC 4492](#)
Was [draft-ietf-tls-rfc4346-bis](#)
[Errata](#)

Published: 2008-08
Other versions: [plain text](#), [pdf](#), [html](#)
[IPR Disclosures](#) | [Dependencies to this RFC](#)

Network Working Group
Request for Comments: 5246
Obsoletes: 3268, 4346, 4366
Updates: 4492
Category: Standards Track

T. Dierks
Independent
E. Rescorla
RTFM, Inc.
August 2008

The Transport Layer Security (TLS) Protocol
Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1. Introduction	4
1.1. Requirements Terminology	5
1.2. Major Differences from TLS 1.1	5
2. Goals	6
3. Goals of This Document	7
4. Presentation Language	7
4.1. Basic Block Size	7
4.2. Miscellaneous	8
4.3. Vectors	8
4.4. Numbers	9
4.5. Enumerateds	9
4.6. Constructed Types	10
4.6.1. Variants	10
4.7. Cryptographic Attributes	12
4.8. Constants	14

5. HMAC and the Pseudorandom Function	14
6. The TLS Record Protocol	15
6.1. Connection States	16
6.2. Record Layer	19
6.2.1. Fragmentation	19

6.2.2. Record Compression and Decompression	20
6.2.3. Record Payload Protection	21
6.2.3.1. Null or Standard Stream Cipher	22
6.2.3.2. CBC Block Cipher	22
6.2.3.3. AEAD Ciphers	24
6.3. Key Calculation	25
7. The TLS Handshaking Protocols	26
7.1. Change Cipher Spec Protocol	27
7.2. Alert Protocol	28
7.2.1. Closure Alerts	29
7.2.2. Error Alerts	30
7.3. Handshake Protocol Overview	33
7.4. Handshake Protocol	37
7.4.1. Hello Messages	38
7.4.1.1. Hello Request	38
7.4.1.2. Client Hello	39
7.4.1.3. Server Hello	42
7.4.1.4. Hello Extensions	44
7.4.1.4.1. Signature Algorithms	45
7.4.2. Server Certificate	47
7.4.3. Server Key Exchange Message	50
7.4.4. Certificate Request	53
7.4.5. Server Hello Done	55
7.4.6. Client Certificate	55
7.4.7. Client Key Exchange Message	57
7.4.7.1. RSA-Encrypted Premaster Secret Message	58
7.4.7.2. Client Diffie-Hellman Public Value	61
7.4.8. Certificate Verify	62
7.4.9. Finished	63

[\[include full document text\]](#)

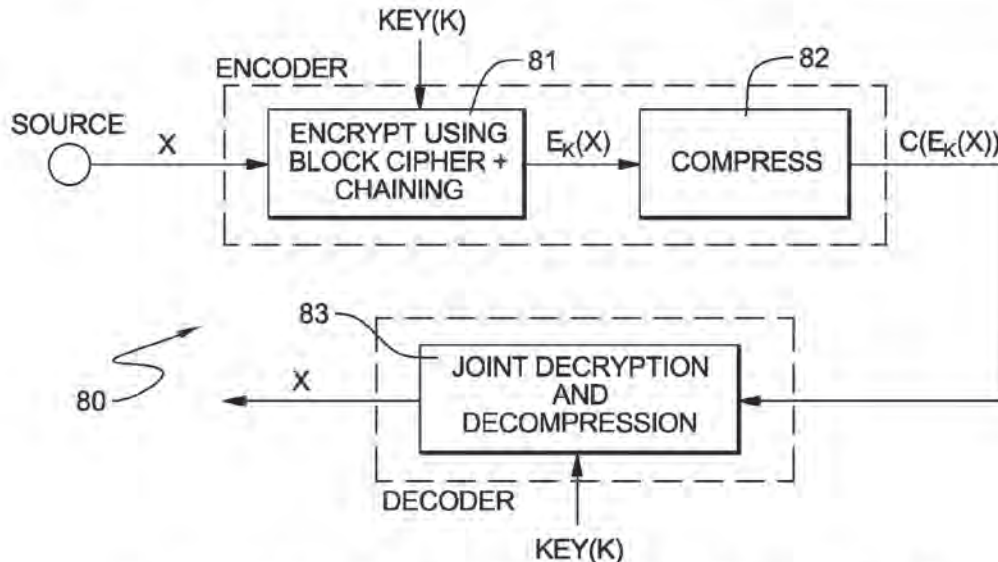
EXHIBIT C



US 20110103580A1

(19) **United States**(12) **Patent Application Publication**
Hazay et al.(10) **Pub. No.: US 2011/0103580 A1**(43) **Pub. Date: May 5, 2011**(54) **COMPRESSING ENCRYPTED DATA
WITHOUT THE ENCRYPTION KEY****Publication Classification**(51) **Int. Cl.**
H04L 9/28 (2006.01)
G06F 17/30 (2006.01)
(52) **U.S. Cl.** **380/28; 707/802; 707/E17.002**(57) **ABSTRACT**

A method, system and computer program product are disclosed for compressing encrypted data, wherein the data is encrypted by using a block encryption algorithm in a chained mode of operation, and the encrypted data is comprised of a set of N encrypted blocks, $C_1 \dots C_N$. In one embodiment, the method comprises leaving block C_N uncompressed, and compressing all of the blocks $C_1 \dots C_N$ in a defined sequence using a Slepian-Wolf code. In an embodiment, the data is encrypted using an encryption key K , and the compressing includes compressing all of the blocks $C_1 \dots C_N$ without using the encryption key. In one embodiment, the compressing includes outputting the blocks $C_1 \dots C_N$ as a set of compressed blocks $\text{Cmpr}C_1 \dots \text{Cmpr}C_{N-1}$; and the method further comprises decrypting C_N to generate a reconstructed block X_N , and decrypting and decompressing the set of compressed blocks using X_N .

(75) **Inventors:** **Camit Hazay**, Yorktown Heights, NY (US); **Ashish Jagmohan**, Yorktown Heights, NY (US); **Demijan Kline**, Yorktown Heights, NY (US); **Hugo M. Krawczyk**, Yorktown Heights, NY (US); **Tal Rabin**, Yorktown Heights, NY (US)(73) **Assignee:** **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)(21) **Appl. No.:** **12/610,754**(22) **Filed:** **Nov. 2, 2009**

**PREFERRED EMBODIMENT OF SYSTEM FOR COMPRESSION OF
DATA ENCRYPTED USING BLOCK CIPHER AND CHAINING**

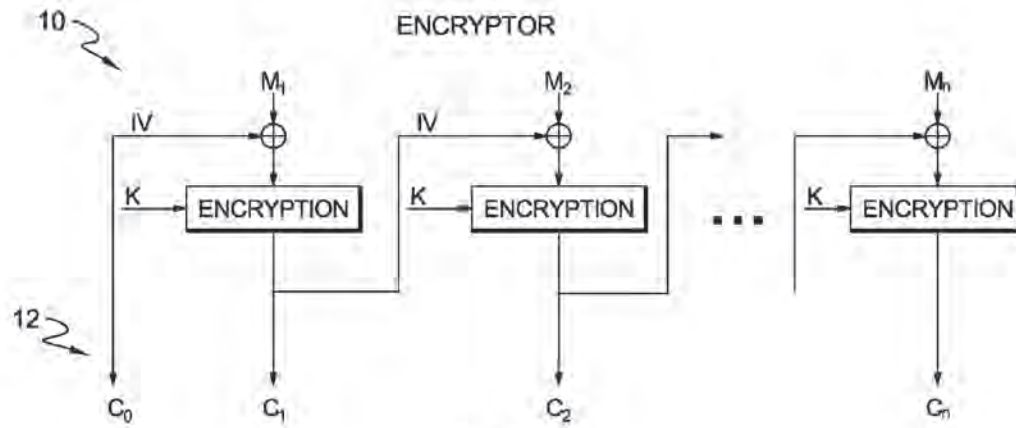


FIG. 1

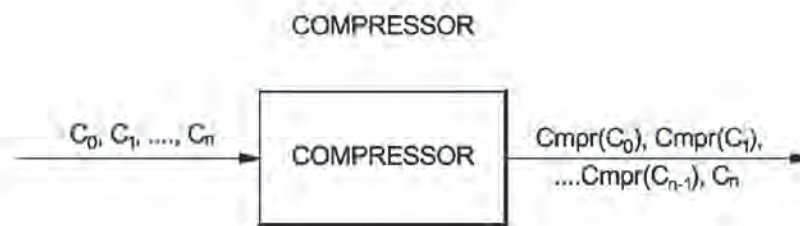


FIG. 2

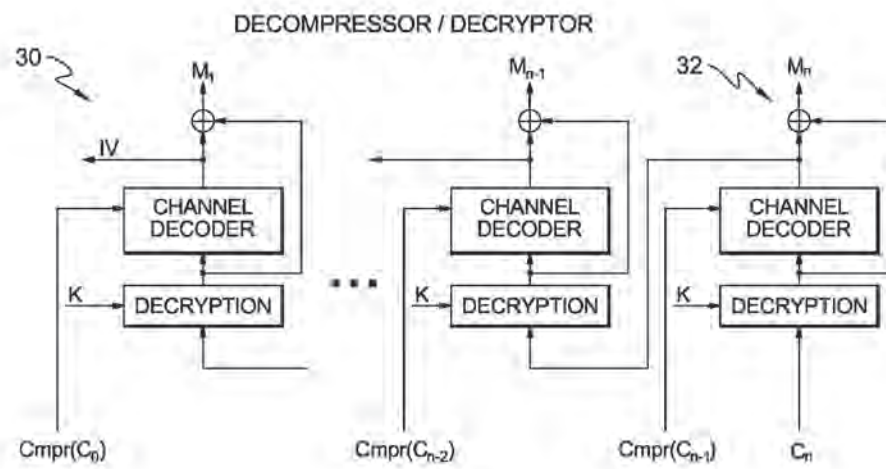
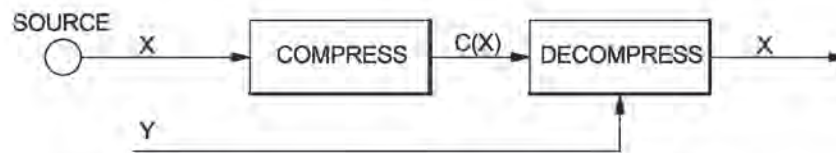
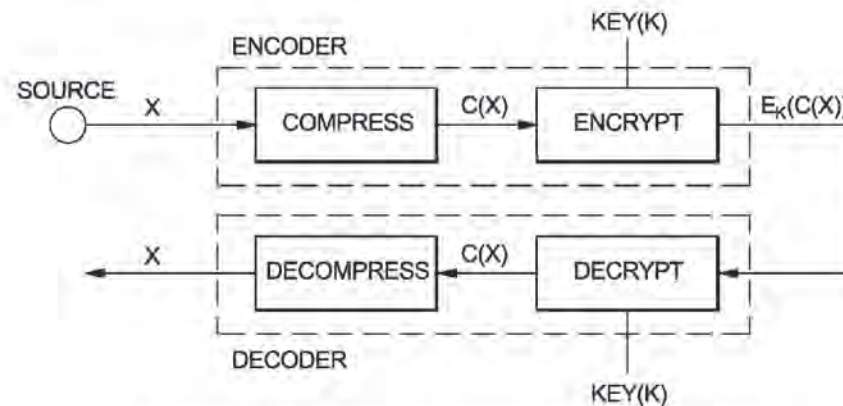


FIG. 3



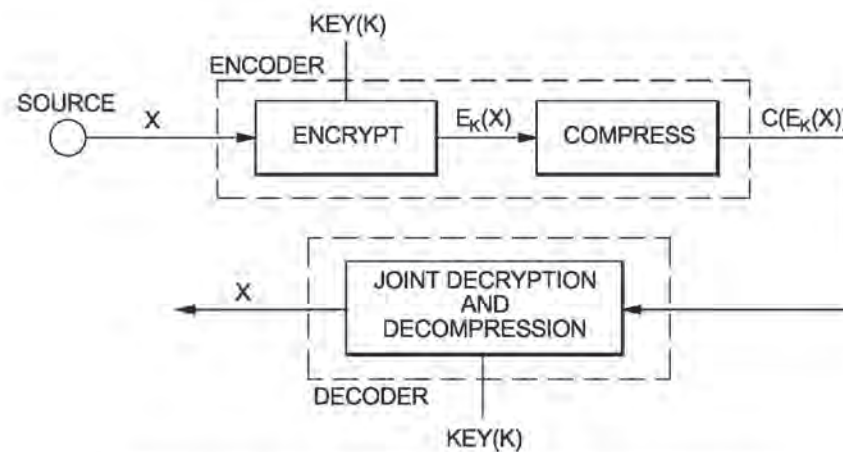
LOSSLESS SOURCE CODE WITH DECODER SIDE-INFORMATION

FIG. 4



TRADITIONAL SYSTEM WITH COMPRESSION DONE FIRST

FIG. 5(a)



SYSTEM WITH ENCRYPTION DONE BEFORE COMPRESSION

FIG. 5(b)

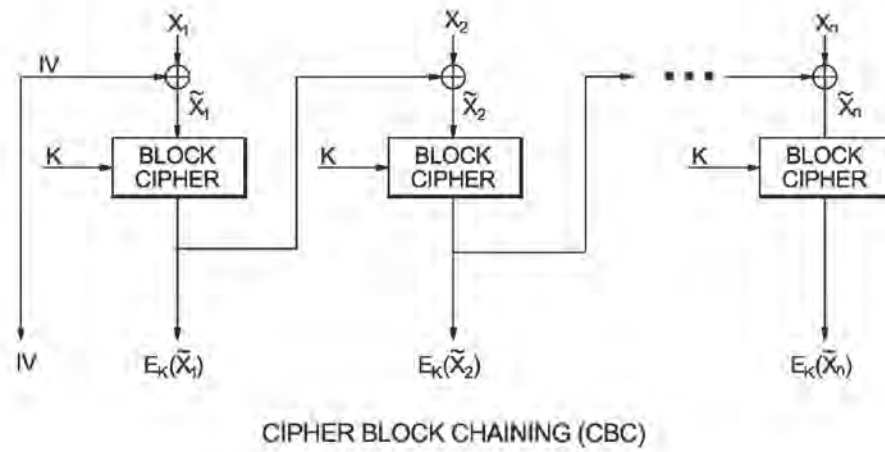


FIG. 6

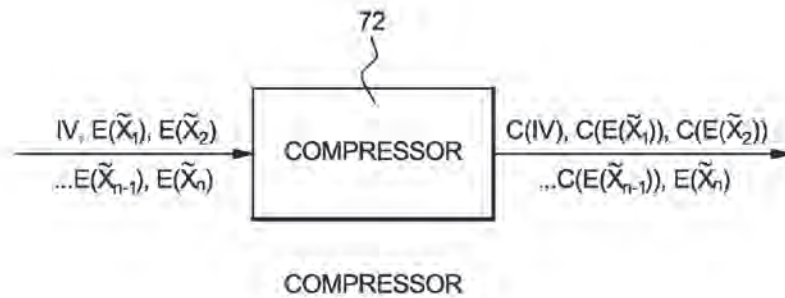
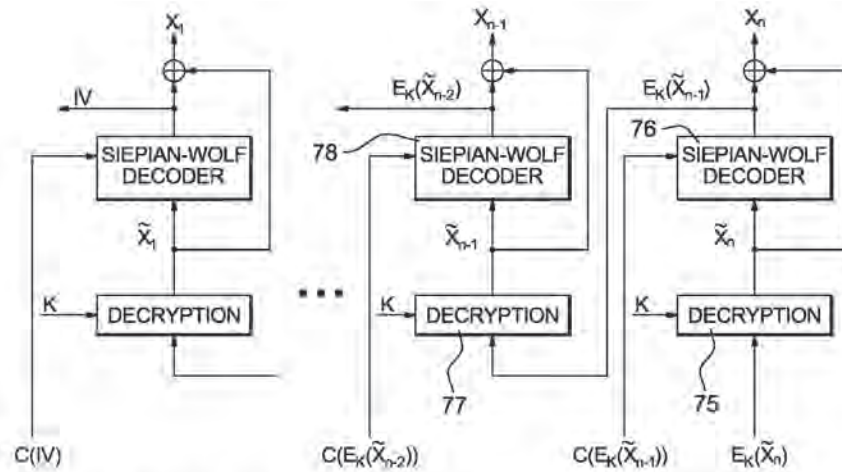
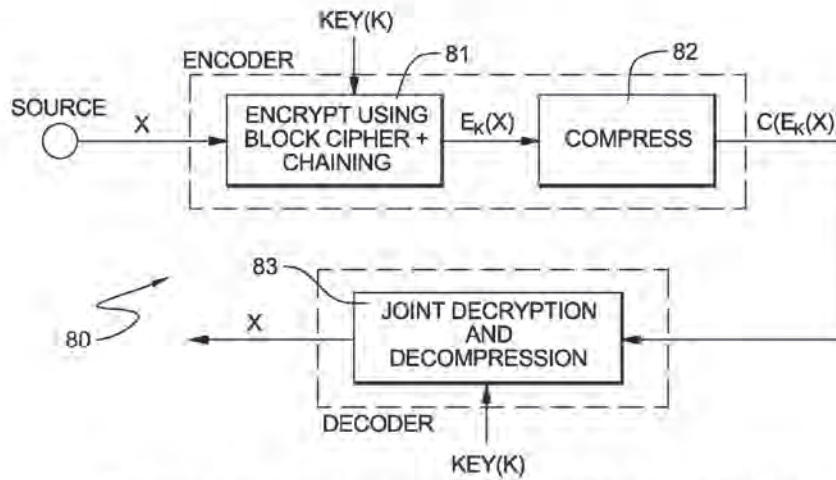


FIG. 7



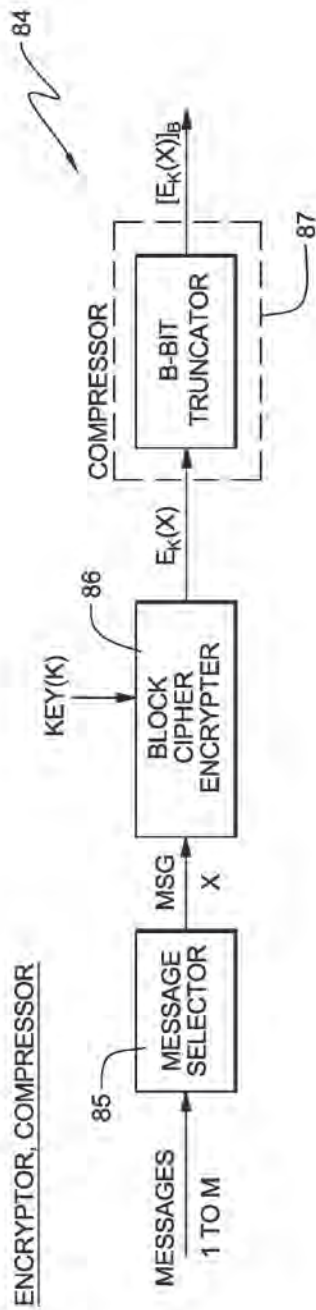
JOINT DECRYPTION AND DECODING AT THE RECEIVER. IT IS PERFORMED SERIALY FROM RIGHT TO LEFT

FIG. 8

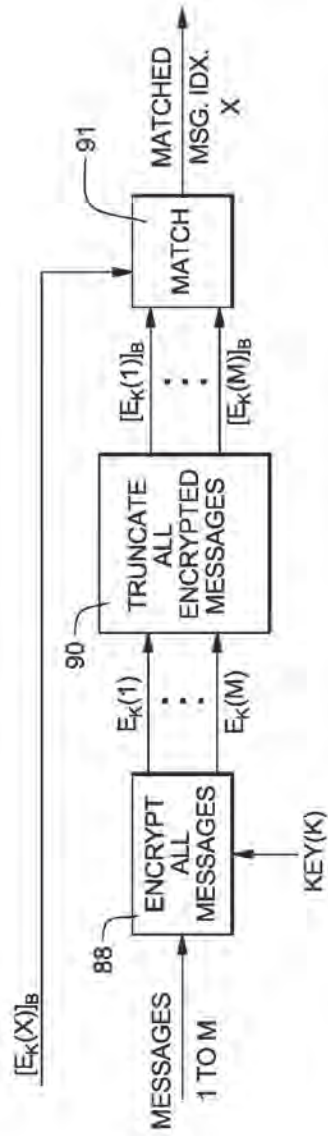


PREFERRED EMBODIMENT OF SYSTEM FOR COMPRESSION OF DATA ENCRYPTED USING BLOCK CIPHER AND CHAINING

FIG. 9

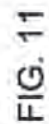


JOINT DECRYPTION AND DECOMPRESSION



EMBODIMENT OF SYSTEM FOR COMPRESSION OF DATA ENCRYPTED USING BLOCK CIPHER AND CORRESPONDING SYSTEM FOR DECOMPRESSION AND DECRYPTION

FIG. 10



COMPRESSING ENCRYPTED DATA WITHOUT THE ENCRYPTION KEY

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] This invention, generally, relates to compressing encrypted data, and more specifically, to compressing encrypted data without using or requiring knowledge of the encryption key.

[0003] 2. Background Art

[0004] Traditionally in communication systems, data from a source is first compressed and then encrypted before it is transmitted over a channel to the receiver. While in many cases this approach is befitting, there exist scenarios where there is a need to reverse the order in which data encryption and compression are performed. Consider for instance a network of low-cost sensor nodes that transmit sensitive information over the internet to a recipient.

[0005] The sensor nodes need to encrypt data to hide it from potential eavesdroppers, but they do not necessarily want to compress it as that would require additional hardware and thus higher implementation cost. On the other hand, the network operator that is responsible for transferring the data to the recipient wants to compress the data to maximize the utilization of its resources. It is important to note that the network operator is not trusted and hence does not have access to the key used for encryption and decryption of data. If it had the key, it could simply decrypt data, compress and encrypt again.

[0006] Related work in the area of compression and encryption can be classified into three main categories. The first category includes systems and methods for compressing encrypted data i.e. systems in which compression is performed prior to encryption. This category includes the systems/methods described in U.S. Pat. No. 6,122,378 ('Data compression/encryption method and system'), U.S. Patent Application Publication No. 2007/0263876A1 ('In-memory compression and encryption') and U.S. Pat. No. 7,295,673 ('Method and system for securing compressed digital video'). The second category includes systems and methods for simultaneously performing compression and encryption, wherein the encryption key (or a constant value, repeating cipher-text) is assumed known during compression. This category includes the systems/methods described in U.S. Patent Application Publication No. 2004/0136566A1 ('Method and apparatus for encrypting and compressing multimedia data'), U.S. Pat. No. 6,122,379 ('Method and apparatus for performing simultaneous data compression and encryption'), and U.S. Patent Application Publication No. 2008/0162521 ('Compression of encrypted data in database management systems'). The main shortcoming of the systems in these two categories is that they do not allow encryption after compression and without knowledge of the encryption key.

[0007] The third category includes the systems/methods described in the papers 'On Compressing Encrypted Data,' M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg and K. Ramchandran, IEEE Transactions on Signal Processing, October 2004 (Johnson et al. I), and 'On Compressing Encrypted Data without the Encryption Key', M. Johnson, D. Wagner and K. Ramchandran, Theory of Cryptography Conference 2004. In these methods Slepian-Wolf coding principles are used to compress data encrypted with a one-time pad and with a stream cipher. These procedures, however, do

not do compression of data encrypted with block ciphers in chaining modes, which are commonly used for most data.

[0008] Block ciphers with a fixed key are a bijection, therefore the entropy of an input is the same as that of the output. It follows that it is theoretically possible to compress the source to the same level as before encryption. However, in practice, encrypted data appears to be random and the conventional compression techniques do not yield desirable results. Consequently, it was long believed that encrypted data is practically incompressible. In the above-mentioned Johnson et al. I paper, the authors break that paradigm and show that the problem of compressing one-time pad encrypted data translates to the problem of compressing correlated sources, which was solved by Slepian and Wolf (see D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," IEEE Trans. Info. Theory, vol. 19, pp. 471-480, July 1973) and for which practical and efficient codes are known. Compression is practically achievable due to a simple symbol-wise correlation between the key (one-time pad) and the encrypted message. However, when such correlation is more complex, as is the case with block ciphers, the approach to Slepian-Wolf coding utilized in Slepian et al. is not directly applicable.

[0009] Therefore, a need exists for a method for compressing encrypted data without knowledge of the compression key, wherein the encryption of the data has been performed by one of the popularly used block ciphers.

BRIEF SUMMARY

[0010] Embodiments of the invention provide a method, system and computer program product for compressing encrypted data, wherein said data is encrypted by using a block encryption algorithm in a chained mode of operation, and said encrypted data is comprised of a set of N encrypted blocks, $C_1, \dots, C_N$. In one embodiment, the method comprises leaving block C_N uncompressed; and compressing all of the blocks $C_1, \dots, C_{N-1}$ in a defined sequence. In an embodiment, said data is encrypted using an encryption key K, and said compressing includes compressing all of the blocks $C_1, \dots, C_{N-1}$ without using the encryption key and with using a Slepian-Wolf code.

[0011] In one embodiment, said compressing includes outputting the blocks $C_1, \dots, C_{N-1}$ as a set of compressed blocks $\text{Cmpr}C_1, \dots, \text{Cmpr}C_{N-1}$, and the method further comprises decrypting C_N to generate a reconstructed block X_N , and decrypting and decompressing said set of compressed blocks using X_N .

[0012] In an embodiment, the decrypting and decompressing includes performing Slepian-Wolf decoding to decompress said set of compressed blocks. In one embodiment, said decompressing includes decompressing said set of compressed blocks in a given sequence to generate a series of reconstructed blocks, including using X_N as side information to decompress $\text{Cmpr}C_{N-1}$ in said Slepian-Wolf decoding. In an embodiment, said decompressing includes, when decompressing each of the compressed blocks $\text{Cmpr}C_1, \dots, \text{Cmpr}C_{N-1}$, using information from a previously generated one of the reconstructed blocks as side information in said Slepian-Wolf decoding.

[0013] In one embodiment, said data is encrypted using an initial vector IV, and said compressing includes compressing said initial vector using the Slepian-Wolf code. In an embodi-

ment, the defined sequence is $C_{N-1}, C_{N-2}, C_{N-3} \dots, C_1$, and said compressing includes compressing said initial vector after compressing C_1 .

[0014] Embodiments of the invention can be used to encrypt data produced by many block-ciphers (such as the popular AES cipher) when used in a chained mode of operation. Chained modes of operation such as CBC, CFB and OFB are the most commonly used modes in practice. Thus embodiments of the invention allow compression of data produced by commonly used encryption algorithms without requiring the encryption key.

[0015] In contrast to stream ciphers, such as the one-time pad, block ciphers are highly nonlinear and the correlation between the key and the ciphertext is, by design, hard to characterize. The present invention effectively addresses this difficulty can be circumvented and compresses block ciphers when they are used in conjunction with chaining modes. We note that AES is used as an example of a block cipher, but the techniques of this invention also apply to other block ciphers.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0016] FIG. 1 shows an Encryptor that may be used with this invention.

[0017] FIG. 2 illustrates a Compressor that may be used in the present invention.

[0018] FIG. 3 depicts a Decompressor/decryptor that may be used with the present invention.

[0019] FIG. 4 illustrates lossless source coding with decoder side-information.

[0020] FIG. 5(a) shows a traditional compression/encryption system, with compression done before encryption.

[0021] FIG. 5(b) shows a system with encryption done before compression as described, for example, in Johnson I.

[0022] FIG. 6 depicts the Cipher Block Chaining mode of operation used, for example, by the AES cryptosystem.

[0023] FIG. 7 illustrates another view of a compressor that may be used in the implementation of the invention.

[0024] FIG. 8 shows joint decryption and decoding in accordance with embodiments of this invention.

[0025] FIG. 9 shows an embodiment of the system for compressing data encrypted with a block cipher and chaining, where compression is done without knowledge of the encryption key.

[0026] FIG. 10 shows another embodiment of a system where data encrypted with a block cipher is compressed without knowledge of the encryption key, and of the corresponding decompression and decryption system.

[0027] FIG. 11 illustrates a computing environment that may be used in embodiments of the invention.

DETAILED DESCRIPTION

[0028] As will be appreciated by one skilled in the art, the present invention may be embodied as a system, method or computer program product. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, the present invention may take the form of a com-

puter program product embodied in any tangible medium of expression having computer usable program code embodied in the medium.

[0029] Any combination of one or more computer usable or computer readable medium(s) may be utilized. The computer-usable or computer-readable medium may be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a non-exhaustive list) of the computer-readable medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CDROM), an optical storage device, a transmission media such as those supporting the Internet or an intranet, or a magnetic storage device. Note that the computer-usable or computer-readable medium could even be paper or another suitable medium, upon which the program is printed, as the program can be electronically captured, via, for instance, optical scanning of the paper or other medium, then compiled, interpreted, or otherwise processed in a suitable manner, if necessary, and then stored in a computer memory. In the context of this document, a computer-usable or computer-readable medium may be any medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device. The computer-usable medium may include a propagated data signal with the computer-usable program code embodied therewith, either in baseband or as part of a carrier wave. The computer usable program code may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc.

[0030] Computer program code for carrying out operations of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[0031] The present invention is described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data pro-

cessing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer program instructions may also be stored in a computer-readable medium that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable medium produce an article of manufacture including instruction means which implement the function/act specified in the flowchart and/or block diagram block or blocks.

[0032] The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0033] The present invention relates to compressing encrypted data without requiring access to a secure cryptographic key. Embodiments of the invention apply to block ciphers that are used in one of the applicable chaining modes. The description below and the diagrams are based on Cipher Block-Chaining (CBC), but the same principles are applicable to other modes that use an XOR operation for chaining.

[0034] FIG. 1 depicts a generic block encryption algorithm 10 employed in the CBC mode. Assuming that the plaintext message is comprised of N blocks M_1 through M_N , the output 12 of the encryptor is a set of N encrypted blocks C_0 through C_N , where C_0 is the initialization vector IV.

[0035] The compressor is depicted in FIG. 2. The compressor does not have access to the cryptographic key used to encrypt and decrypt messages. Even without the knowledge of the cryptographic key, the compressor can compress encrypted and seemingly random data, thereby counting on the fact that the receiver, knowing the encryption key, will be able to jointly decompress and decrypt. It is assumed, though, that the compressor has knowledge about the source statistics. Generally, compression is performed as follows: the last block C_N is left in uncompressed form, while all the remaining encrypted block are compressed using an appropriate Slepian-Wolf code of a matching size.

[0036] The decompression/decryption part of the receiver is depicted at 30 in FIG. 3. It is assumed, that the receiver has knowledge of the cryptographic key. First the receiver decrypts the uncompressed block C_N . At the output 32 of the decryptor we have $(M_N \text{ XOR } C_{N-1})$. Now the receiver exploits the fact that C_{N-1} and $(M_N \text{ XOR } C_{N-1})$ are strongly correlated. By the Slepian-Wolf theorem, the receiver can recover C_{N-1} using $(M_N \text{ XOR } C_{N-1})$ and $\text{Cmpr}(C_{N-1})$. M_N is then simply $C_{N-1} \text{ XOR } (M_N \text{ XOR } C_{N-1})$.

[0037] With the knowledge of C_{N-1} and $\text{Cmpr}(C_{N-2})$, the receiver can recover M_{N-1} using the exact same procedure as described above. The same is repeated sequentially in backward-to-forward fashion until all message blocks are compressed and decrypted.

[0038] More specific examples of embodiments of the present invention are discussed below.

[0039] A formal definition of an encryption scheme is given in "Introduction To Modern Cryptography," by J. Katz and U. Lindell, Chapman & Hall/CRC, 2007. A private-key encryption scheme is a triple of algorithms (Gen, E, D), where Gen

is a probabilistic algorithm that outputs a key K chosen according to some distribution that is determined by the scheme; the encryption algorithm E takes as input a key K and a plaintext message X and outputs a ciphertext $E_K(X)$; the decryption algorithm D takes as input a key K and a ciphertext $E_K(X)$ and outputs a plaintext X .

[0040] In private-key encryption schemes the same key is used for encryption and decryption algorithms. Private-key encryption schemes can be divided in two categories: block ciphers and stream ciphers. Stream ciphers encrypt plaintext one symbol at a time, typically by summing it with a key (XOR operation for binary alphabets). In contrast, block ciphers represent a different approach where encryption is accomplished by means of nonlinear mappings on input blocks of fixed length. Common examples of block ciphers are the Advanced Encryption Standard (AES) (see W. Mao, *Modern Cryptography: Theory and Practice*, Prentice Hall, 2003), and the Data Encryption Standard (DES) (see N. B. of Standards, *Data Encryption Standard (DES)*, U.S. Department of Commerce, Washington D.C., 1977). Typically, block ciphers are not used as a stand-alone encryption procedure but are rather combined to work on variable length data using composition mechanisms known as chaining modes or modes of operation. The most common mode of operation is cipher-block chaining (CBC), discussed below.

[0041] FIG. 4 illustrates the source coding problem with decoder side information. Consider random variables X (termed the source), and Y (termed the side-information), both over a finite-alphabet and with a joint probability distribution P_{XY} . Consider a sequence of independent realizations of (X, Y) denoted by $\{X_i, Y_i\}_{i=1}^n$.

[0042] The problem at hand is of losslessly encoding $\{X_i\}_{i=1}^n$, with $\{Y_i\}_{i=1}^n$, known only to the decoder. In Slepian et al., Slepian and Wolf showed that, asymptotically in block-length, this can be done at rates arbitrarily close to the conditional entropy $H(X|Y)$. Practical Slepian-Wolf coding schemes use constructions based on good linear error-correcting codes (see A. Aaron and B. Girod, "Compression with side information using turbo codes," in *IEEE Data Compression Conf.*, 2002, pp. 252-261; and J. Garcia-Frias, "Compression of correlated binary sources using turbo codes," *IEEE Communications Letters*, vol. 5, pp. 417-419, October 2001).

[0043] Of interest are systems which perform both compression and encryption, wherein only the encryptor has access to the key. Typically, in such systems, encryption is performed after compression as depicted in FIG. 5(a). This is a consequence of the traditional view which considers ciphertext data hard to compress without knowledge of the key. In the above mentioned Johnson et al. I paper, a system similar to FIG. 5(b) is considered instead, in which the order of the encryption and compression operations at the encoder is reversed. It may be noted that in this system, only the encryptor has access to the key. The authors consider encryption of a plaintext X using a one-time pad scheme, with a finite-alphabet key (pad) K , to generate the ciphertext $E_K(X)$, i.e.

$$E_K(X) = X \oplus K, \forall i \in \mathbb{Z}$$

[0044] This is followed by compression, which is unknown of K , to generate the compressed ciphertext $C(E_K(X))$.

[0045] The key insight underlying the approach in Johnson, et al. I is that the problem of compression in this case can be formulated as a Slepian-Wolf coding problem. In this formulation, the ciphertext $E_K(X)$ is cast as a source, and the shared

key K is cast as the decoder only side-information. The joint distribution of the source and side-information can be determined from the statistics of the source. For example, in the binary case with a uniformly distributed K and X with $\Pr[X=1]=p$,

$$P(E_K(X)) = h(K) = p. \quad (1)$$

[0046] The decoder has knowledge of K , and of the source statistics. It uses this knowledge to reconstruct the ciphertext $E_K(X)$ from the compressed message $C(E_K(X))$, and to subsequently decrypt the plaintext X . This formulation is leveraged in Johnson, et al. I to show that exactly the same lossless compression rate, $H(X)$, can be asymptotically achieved in the system shown in FIG. 5(b), as can be achieved in FIG. 5(a). Further, this can be done while maintaining information-theoretic security.

[0047] The one-time pad and stream ciphers, while convenient for analysis, are not the only forms of encryption in practice. In fact, the prevalent method of encryption uses block ciphers in chaining modes such as CBC. Thus, a desirable extension of the technique in Johnson, et al. I would be to conventional encryption schemes such as the popular AES encryption method. Attempting to do so, however, proves to be problematic. The method in Johnson, et al. I leverages the fact that in a one-time pad encryption scheme a simple symbol-wise correlation exists between the key K and the ciphertext $E_K(X)$, as seen in equation (1). Unfortunately, for block ciphers such as AES no such correlation structure is known. Moreover, any change in the plaintext is diffused in the ciphertext, and quantifying the correlation (or the joint probability distribution) of the key and the ciphertext is believed to be computationally infeasible and a requirement for the security of the block cipher.

[0048] The present invention effectively addresses this problem by exploiting the chaining modes popularly used with block ciphers. Based on this insight, this invention provides an approach for compressing data encrypted with AES, without knowledge of the key. As in Johnson, et al. I, the present invention is based on the use of Slepian-Wolf coding.

[0049] If a block cipher operates on each block of data individually, two identical inputs will produce two identical outputs. While this weakness does not necessarily enable an unauthorized user to understand contents of an individual block it can give him information about frequently occurring data patterns. To address this problem, various chaining modes, also called modes of operation, are used in conjunction with block ciphers. The idea is to randomize each plaintext block, by using a randomization vector derived as a function of previous encryptor inputs or outputs. The randomization prevents two identical plaintext blocks from being encrypted into two identical ciphertext blocks, thus preventing leakage of information about data patterns.

[0050] The CBC mode of operation, depicted in FIG. 6 is a widely used block cipher. In this Figure, X_i denotes the i -th plaintext block. An initial pseudorandom vector, used to initiate the chaining, is denoted as IV . Ciphertext $E_K(X)$ is generated by applying the AES encryption algorithm with key K to the plaintext X . In the CBC mode, each plaintext block X is randomized prior to encryption, by being XOR-ed with the ciphertext block corresponding to the previous plaintext block X_{i-1} , to obtain X_i . Thus, the i -th ciphertext block is generated as

$$E_K(X_i) = E_K(X_i \oplus E_K(X_{i-1})),$$

where the pseudorandom vector IV , assumed to be drawn uniformly from the source alphabet, is used instead of $E_K(X_0)$. This method of chaining ensures that frequently occurring plaintext patterns do not lead to repeating ciphertext patterns. Note that block ciphers in CBC mode are employed as the default mechanism in widespread security standards such as IPsec (see S. Kent and K. Seo, "Security architecture for the internet protocol," in RFC 4301, December 2005) and TLS/SSL (see T. Dierks and E. Rescorla, "The TLS protocol—version 1.2," in RFC 5246, August 2008) and hence it is the prevalent method of encrypting internet traffic. In the discussion below, the length of a plaintext block is represented as m , and X_i and X_j are drawn from the same binary extension field X^m . Further, X_i is generated by an i.i.d. source with marginal distribution P_X .

[0051] The statistical relationship between the key K and the i -th AES encrypted ciphertext $E_K(X_i)$ is hard to characterize. However, the joint distribution of the randomization vector $E_K(X_{i-1})$ and the i -th input to the AES encryptor X_i is easier to characterize, as it is governed by the distribution of the plaintext block X_i . For example, in the i.i.d. source case being considered, $E_K(X_{i-1})$ and X_i are related through a symbol-wise model governed by the distribution P_X . The correlation induced by the use of the chaining mode can be exploited to allow compression of encrypted data using Slepian-Wolf coding shown below.

[0052] Let $\{C_{m,R}, D_{m,R}\}$ denote an order m Slepian-Wolf code with encoding rate R . Here, the Slepian-Wolf encoding function $C_{m,R}$ is a mapping from X^m to the index set $\{1, \dots, 2^{mR}\}$, and the Slepian-Wolf decoding function $D_{m,R}$ is a mapping from $\{1, \dots, 2^{mR}\} \times X^m$ to X^m . The compression method is illustrated in FIG. 7. The input to the compressor **72** is comprised of the IV and n ciphertext blocks $E_K(X_i)$, $1 \leq i \leq n$, generated by the AES cryptosystem running in the CBC mode. Since $E_K(X_i) \in X^m$, the total length of the input sequence is $(n+1)m \log|X|$ bits. The compressor applies the Slepian-Wolf encoder $C_{m,R}$ to the IV and each of the first $n-1$ ciphertext blocks independently, while the n -th block is left unchanged. Thus, the output of the compressor is the sequence $C(IV), C(E_K(X_1)), C(E_K(X_2)), \dots, C(E_K(X_{n-1})), E_K(X_n)$, where the subscripts are omitted for notational simplicity. The length of the output sequence is $n \cdot m \cdot R + m \log|X|$ bits. Thus, the compressor achieves a compression factor of

$$\frac{(n+1) \cdot m \cdot \log|X|}{n \cdot m \cdot R + m \cdot \log|X|} \approx \frac{\log|X|}{R}$$

for large n . Note that the compressor does not need to know the key K . Also, note that this approach only requires a compressed IV , which by itself is incompressible, therefore no performance loss is inflicted by the uncompressed last block.

[0053] The joint decompression and decryption method is shown in FIG. 8. The received compressed sequence is decrypted and decompressed serially, from right to left. In the first step $E_K(X_n)$, which is received without compression, is decrypted at **75** using the key K to generate X_n . Next, Slepian-Wolf decoding is performed at **76** to reconstruct $E_K(X_{n-1})$ using X_n as side-information, and the compressed bits $C(E_K(X_{n-1}))$. The decoder computes $\hat{E}_K(D(C(E_K(X_{n-1})), X_n))$, such that $\hat{E} = E_K(X_{n-1})$ with high-probability if the rate R is high enough. Once $E_K(X_{n-1})$ has been recovered by the Slepian-Wolf decoder, the plaintext block can now be reconstructed as $X_{n-1} = E_K(X_{n-1}) \oplus X_n$. The decoding process now proceeds seri-

ally with $E_K(X_{n-1})$ decrypted at 77 to generate X_{n-1} , which acts as the new Slepian-Wolf side-information at 78. This continues until all plaintext blocks have been reconstructed. [0054] For large m , it follows from the Slepian-Wolf theorem that the rate required to ensure correct reconstruction of the $(i-1)$ -th block with high probability is given as

$$R = H(E_K(\hat{X}_{i-1})|\hat{X}_i) - H(E_K(\hat{X}_{i-1})|E_K(\hat{X}_{i-1}) \oplus \hat{X}_i) \leq H(E_K(\hat{X}_{i-1})|\hat{X}_i) - H(E_K(\hat{X}_{i-1})|E_K(\hat{X}_{i-1}) \oplus \hat{X}_i) = H(\hat{X}_i).$$

[0055] If it is assumed that $E_K(X_{i-1})$ has a uniform distribution, equation (3) becomes an equality. In practice, in is typically small. In this case, the required rate R is a function of P_X , m , the acceptable decoding error probability, and the non-ideal Slepian-Wolf codes used.

[0056] The above description focuses on the CBC mode as the most common form of encryption, but the techniques of this invention can be extended to other CBC-like modes of operation.

[0057] FIG. 9 gives an overview of a system 80 for compression of data encrypted using a block cipher in chaining mode. The source is encrypted using a block cipher 81 in chaining mode, using a key K . The resultant ciphertext $E_K(X)$ is compressed by the compressor 82. The compressed ciphertext $C(E_K(X))$ is transmitted to the joint decoder and decompressor 83 which uses the key K to decode the data X . The compression and decompression/decryption steps are performed as shown in FIGS. 6-8.

[0058] FIG. 10 shows another embodiment of a system 84 for compressing encrypted data without knowledge of the compression key, wherein the encryption of the data has been performed by one of the popularly used block ciphers. In this case, the message to be transmitted, denoted X , is assumed to be one of M messages, selected by the message selector 85. The selected message is encrypted using the block cipher encryptor 86 using the key K . The resultant ciphertext $E_K(X)$ is sent to the compressor 87. The compressor compresses the ciphertext without knowledge of the key K , by truncating the ciphertext to B bits i.e. it discards all bits other than the first B bits of the ciphertext. The value of B is selected as a function of the allowable decoding-failure probability. The compressed ciphertext denoted $[E_K(X)]_B$ is then transmitted. Upon receiving $[E_K(X)]_B$, the joint decryption and decompression is done as follows. All possible messages 1 to M are encrypted, at 88, using the key K , to generate M ciphertexts $E_K(1)$ to $E_K(M)$. These M ciphertexts are each truncated, at 90, to B bits. The received bits $[E_K(X)]_B$ are compared to each of the truncated ciphertexts, and the message index corresponding to the matching ciphertext is, at 91, declared to be the communicated message. This allows simple compression of encrypted data in the case where the number of possible messages is small.

[0059] FIG. 11 illustrates an example of a suitable computing system environment 100 in which the invention may be implemented, although as made clear above, the computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 500.

[0060] With reference to FIG. 11, an exemplary system for implementing the invention includes a general purpose-computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a process-

ing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus).

[0061] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110.

[0062] Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0063] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 9 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0064] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 9 illustrate a hard disk drive 141 that reads from or writes to non-removable, non-volatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable,

volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0065] The drives and their associated computer storage media discussed above and illustrated in FIG. 11 provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 9, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating System 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies.

[0066] A user may enter commands and information into the computer 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB).

[0067] A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. A graphics interface 182, such as Northbridge, may also be connected to the system bus 121. Northbridge is a chipset that communicates with the CPU, or host-processing unit 120, and assumes responsibility for accelerated graphics port (AGP) communications. One or more graphics processing units (GPUs) 184 may communicate with graphics interface 182. In this regard, GPUs 184 generally include on-chip memory storage, such as register storage and GPUs 184 communicate with a video memory 186. GPUs 184, however, are but one example of a coprocessor and thus a variety of co-processing devices may be included in computer 110. A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190, which may in turn communicate with video memory 186. In addition to monitor 191, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0068] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 9. The logical connections depicted in FIG. 9 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also

include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0069] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 11 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0070] One of ordinary skill in the art can appreciate that a computer 110 or other client device can be deployed as part of a computer network. In this regard, the present invention pertains to any computer system having any number of memory or storage units, and any number of applications and processes occurring across any number of storage units or volumes. The present invention may apply to an environment with server computers and client computers deployed in a network environment, having remote or local storage. The present invention may also apply to a standalone computing device, having programming language functionality, interpretation and execution capabilities.

[0071] Thus, methods, systems, and computer program products for inference-driven multi-source semantic search have been described. In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope of the invention as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

1. A method of compressing encrypted data, wherein said data is encrypted by using a block encryption algorithm in a chained mode of operation, and said encrypted data is comprised of a set of N encrypted blocks, $C_1 \dots C_N$, the method comprising:

leaving block C_N uncompressed; and
compressing all of the blocks $C_1 \dots C_{N-1}$ in a defined sequence.

2. The method according to claim 1, wherein said data is encrypted using an encryption key K , and said compressing includes compressing all of the blocks $C_1 \dots C_{N-1}$ using a Slepian-Wolf code without using the encryption key.

3. The method according to claim 1, wherein said data is encrypted using an initial vector IV , and said compressing includes compressing said initial vector using the Slepian-Wolf code.

4. The method according to claim 3, wherein said defined sequence is $C_{N-1}, C_{N-2}, C_{N-3} \dots C_1$.

5. The method according to claim 4, wherein said compressing includes compressing said initial vector after compressing C_1 .

6. The method according to claim 2, wherein the Slepian-Wolf code is a low-density parity-check matrix based code

with block size equal to the encryption block size, and with rate determined as a function of the source statistics, the acceptable decoding error probability and the block length.

7. The method according to claim 1, wherein said compressing includes outputting the blocks $C_1 \dots C_{N-1}$ as a set of compressed blocks $\text{Cmpr}C_1 \dots \text{Cmpr}C_{N-1}$.

8. The method according to claim 7, wherein the decrypting and decompressing includes performing Slepian-Wolf decoding to decompress said set of compressed blocks, and further comprising:

decrypting C_N to generate a reconstructed block X_n ; and
decrypting and decompressing said set of compressed blocks using X_n .

9. The method according to claim 8, wherein said decompressing includes:

decompressing said set of compressed blocks in a given sequence to generate a series of reconstructed blocks, including using X_n as side information to decompress $\text{Cmpr}C_{N-1}$ in said Slepian-Wolf decoding; and
when decompressing each of the compressed blocks $\text{Cmpr}C_1 \dots \text{Cmpr}C_{N-1}$, using information from a previously generated one of the reconstructed blocks as side information in said Slepian-Wolf decoding.

10. The method according to claim 7, wherein the decrypting and decompressing includes using X_n to decode $\text{Cmpr}C_{N-1}$ to generate a decrypted block X_{n-1} .

11. A system for compressing encrypted data, wherein said data is encrypted by using a block encryption algorithm in a chained mode of operation, and said encrypted data is comprised of a set of N encrypted blocks, $C_1 \dots C_N$, the system comprising one or more processing units configured for:

leaving block C_N uncompressed; and
compressing all of the blocks $C_1 \dots C_{N-1}$ in a defined sequence using a Slepian-Wolf code.

12. The system according to claim 11, wherein said data is encrypted using an encryption key K , and said compressing includes compressing all of the blocks $C_1 \dots C_{N-1}$ without using the encryption key.

13. The system according to claim 11, wherein said compressing includes outputting the blocks $C_1 \dots C_{N-1}$ as a set of compressed blocks $\text{Cmpr}C_1 \dots \text{Cmpr}C_{N-1}$.

14. The system according to claim 13 wherein the decrypting and decompressing includes performing Slepian-Wolf decoding to decompress said set of compressed blocks, and said one or more processor units are further configured for:
decrypting C_N to generate a reconstructed block X_n ; and
decrypting and decompressing said set of compressed blocks using X_n .

15. The system according to claim 14 wherein said decompressing includes decompressing said set of compressed blocks in a given sequence to generate a series of reconstructed blocks, including using X_n as side information to decompress $\text{Cmpr}C_{N-1}$ in said Slepian-Wolf decoding.

16. An article of manufacture comprising:

at least one computer usable medium having computer readable program code logic to execute a machine instruction in a processing unit for compressing encrypted data, wherein said data is encrypted by using a block encryption algorithm in a chained mode of operation, and said encrypted data is comprised of a set of N encrypted blocks, $C_1 \dots C_N$, said computer readable program code logic, when executing, performing the following

leaving block C_N uncompressed; and
compressing all of the blocks $C_1 \dots C_{N-1}$ in a defined sequence using a Slepian-Wolf code.

17. The article of manufacture according to claim 16, wherein said data is encrypted using an encryption key K , and said compressing includes compressing all of the blocks $C_1 \dots C_{N-1}$ without using the encryption key.

18. The article of manufacture according to claim 16, wherein said compressing includes outputting the blocks $C_1 \dots C_{N-1}$ as a set of compressed blocks $\text{Cmpr}C_1 \dots \text{Cmpr}C_{N-1}$, and said computer readable program code logic, when executing,

19. The article of manufacture according to claim 18, wherein the decrypting and decompressing includes performing Slepian-Wolf decoding to decompress said set of compressed blocks, and said computer readable program code logic, when executing, performs the following:

decrypting C_N to generate a reconstructed block X_n ; and
decrypting and decompressing said set of compressed blocks using X_n .

20. The article of manufacture according to claim 18, wherein said decompressing includes:

decompressing said set of compressed blocks in a given sequence to generate a series of reconstructed blocks, including using X_n as side information to decompress $\text{Cmpr}C_{N-1}$ in said Slepian-Wolf decoding; and
when decompressing each of the compressed blocks $\text{Cmpr}C_1 \dots \text{Cmpr}C_{N-1}$, using information from a previously generated one of the reconstructed blocks as side information in said Slepian-Wolf decoding.

21. A method of compressing and decrypting encrypted data, wherein said encrypted data is comprised of one or more first encrypted messages obtained by encrypting one or more messages selected from a group of M messages, the method comprising:

compressing each of said one or more first encrypted messages by truncating each of said one or more first encrypted messages to obtain one or more first truncated messages; and

decompressing and decrypting said one or more first truncated messages by

encrypting all of said M messages to obtain a series of second encrypted messages.

truncating said second encrypted messages to obtain a series of second truncated messages, and

comparing each of said one or more first truncated messages to one or more of said series of second truncated messages, using defined criteria, to match each of said one or more first truncated messages to one of said M messages.

22. The method according to claim 21, wherein said first encrypted messages are encrypted using an encryption key K , and said compressing includes compressing each of said one or more first encrypted messages without using said encryption key K .

23. The method according to claim 22, wherein the encrypting all of said M messages includes encrypting all of said M messages using said encryption key K .

24. A system for compressing and decrypting encrypted data, wherein said encrypted data is comprised of one or more first encrypted messages obtained by encrypting one or more messages selected from a group of M messages, the system comprising one or more processing units configured for:

compressing each of said one or more first encrypted messages by truncating each of said one or more first encrypted messages to obtain one or more first truncated messages; and
decompressing and decrypting said one or more first truncated messages by
encrypting all of said M messages to obtain a series of second encrypted messages,
truncating said second encrypted messages to obtain a series of second truncated messages, and
comparing each of said one or more first truncated messages to one or more of said series of second truncated

messages, using defined criteria, to match each of said one or more first truncated messages to one of said M messages.

25. The system according to claim 24, wherein:
said first encrypted messages are encrypted using an encryption key K, and said compressing includes compressing each of said one or more first encrypted messages without using said encryption key K; and
the encrypting all of said M messages includes encrypting all of said M messages using said encryption key K.

* * * * *

EXHIBIT D



US 20100281519A1

(19) **United States**(12) **Patent Application Publication**
Das et al.(10) **Pub. No.: US 2010/0281519 A1**(43) **Pub. Date: Nov. 4, 2010**(54) **PROACTIVE AUTHENTICATION****Publication Classification**(75) **Inventors:** **Subir Das**, Belle Mead, NJ (US);
Yoshihiro Oba, Kawasaki (JP);
Ashutosh Dutta, Bridgewater, NJ (US)(51) **Int. Cl.**
H04L 29/06 (2006.01)
H04W 12/06 (2009.01)(52) **U.S. Cl.** **726/3; 370/331**

Correspondence Address:

ATTN: Stephen B. Parker (WPD)
Westerman, Hattori, Daniels & Adrian, LLP
1250 Connecticut Ave., Suite 700
Washington, DC 20036 (US)(73) **Assignees:** **KABUSHIKI KAISHA**
TOSHIBA, Tokyo (JP);
TELCORDIA
TECHNOLOGIES, INC.,
Piscataway, NJ (US)(21) **Appl. No.: 12/772,632**(22) **Filed: May 3, 2010****Related U.S. Application Data**

(60) Provisional application No. 61/175,016, filed on May 3, 2009, provisional application No. 61/221,551, filed on Jun. 29, 2009.

(57) **ABSTRACT**

A system for proactively authenticating includes a server having media independent access functions including media independent authentication functions that authenticates other entities attached via an interface to an end of a link specific to a media. A plurality of heterogeneous networks each having media specific access functions including authentication functions corresponding to the other entities attached via the interface to the end of the link specific to the media and mobile devices connected to the plurality of heterogeneous networks, and the server having predefined media independent handover protocols and media independent handover identities based on the media independent functions related to handover, in which the server authenticates candidate access networks prior to the handover of the mobile devices from serving access networks to the candidate access networks each of which belonging to the plurality of heterogeneous access networks having the link specific to the media.

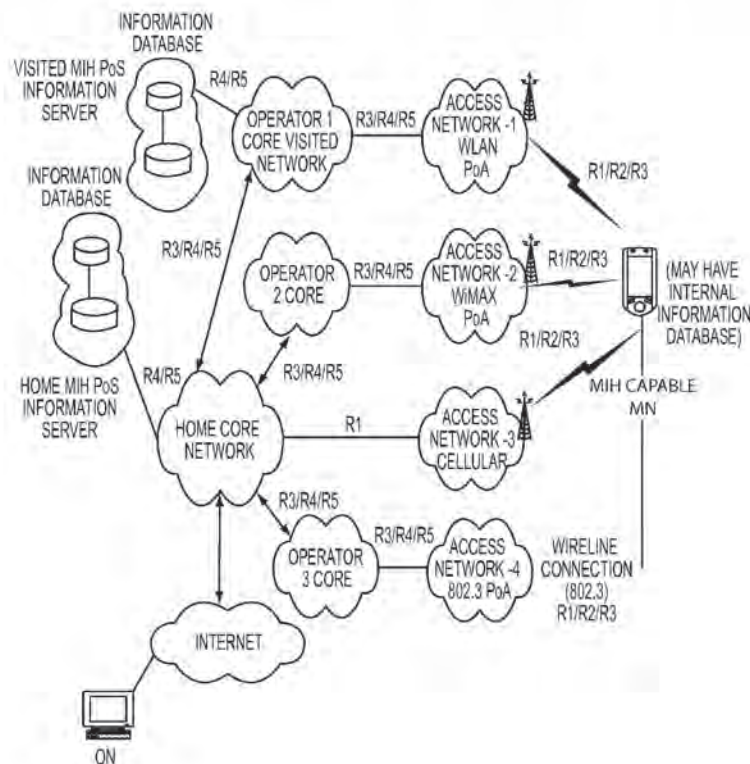
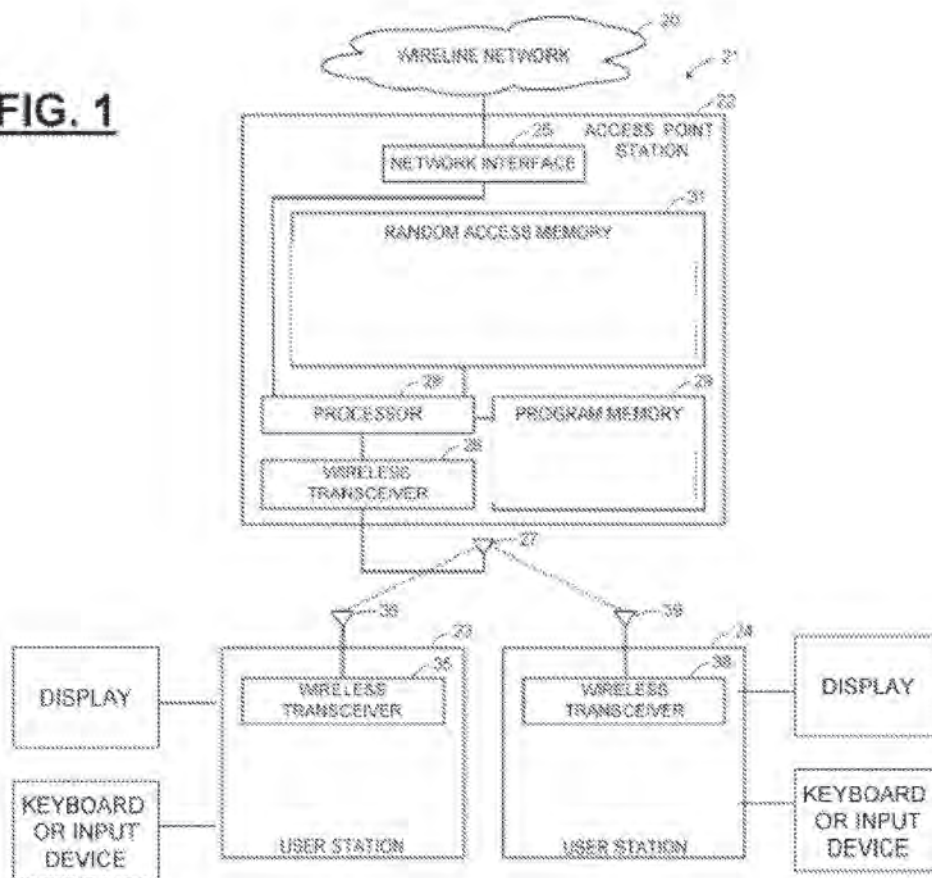


FIG. 1

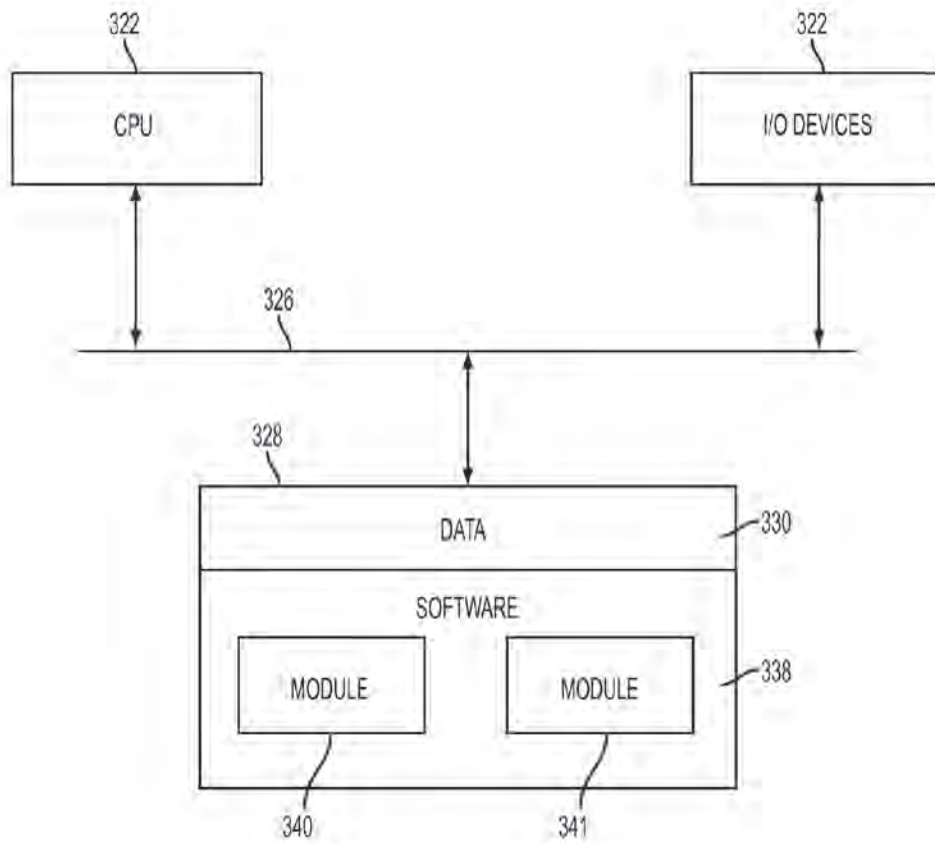


FIG. 2

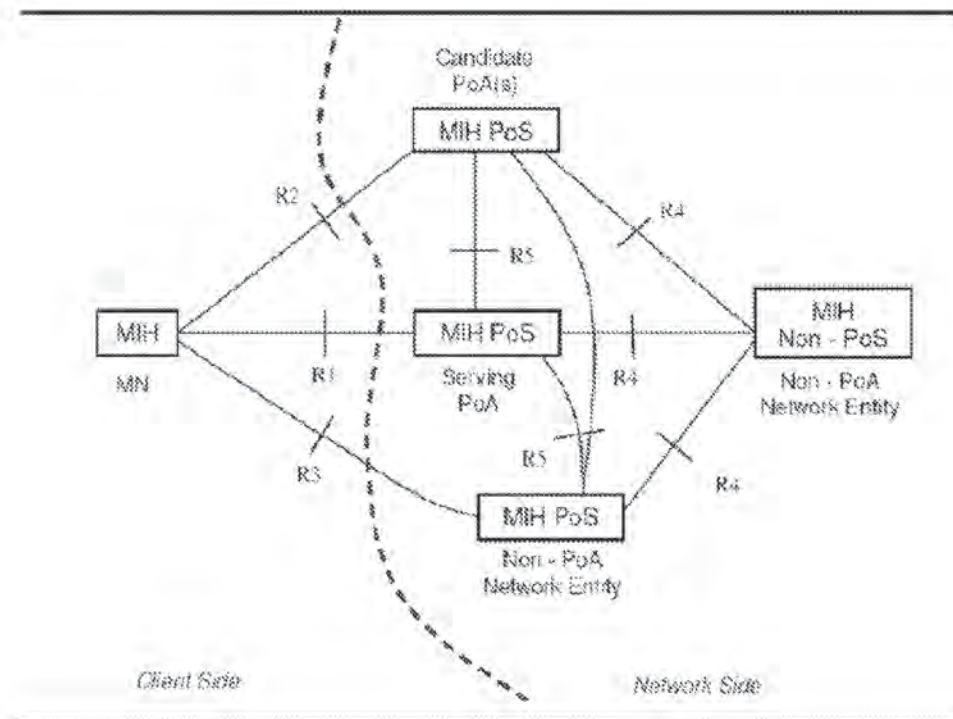


FIG. 3

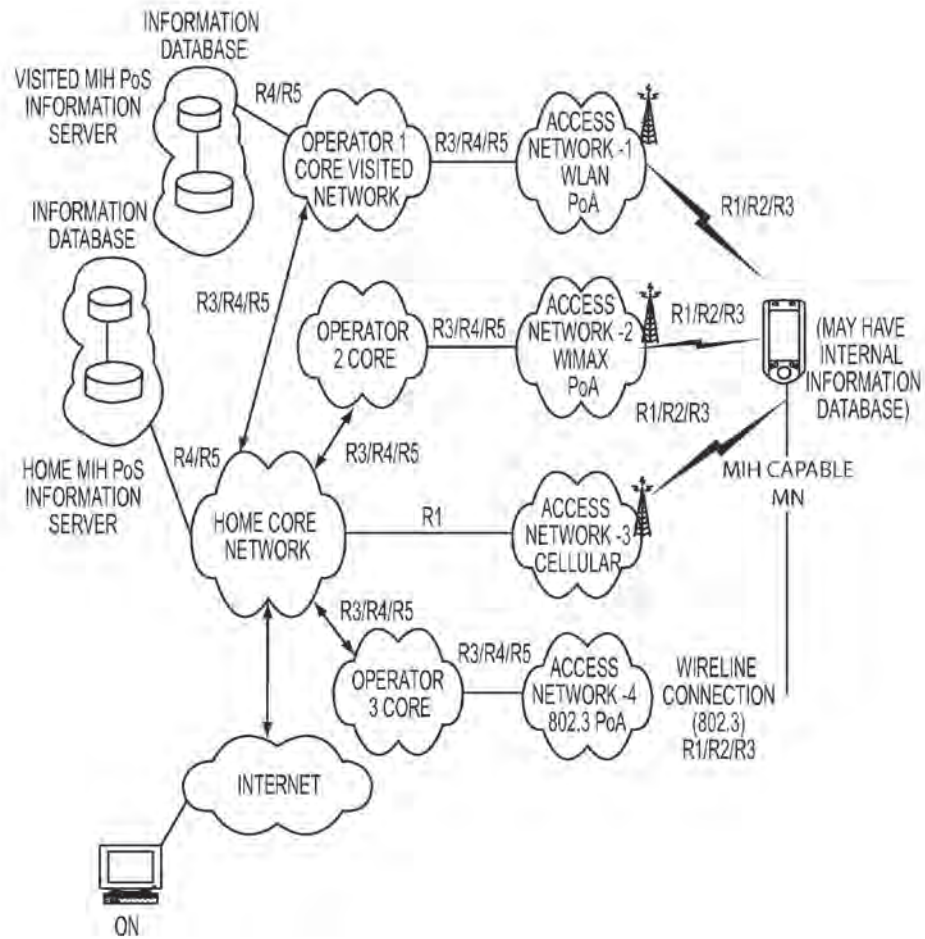


FIG. 4

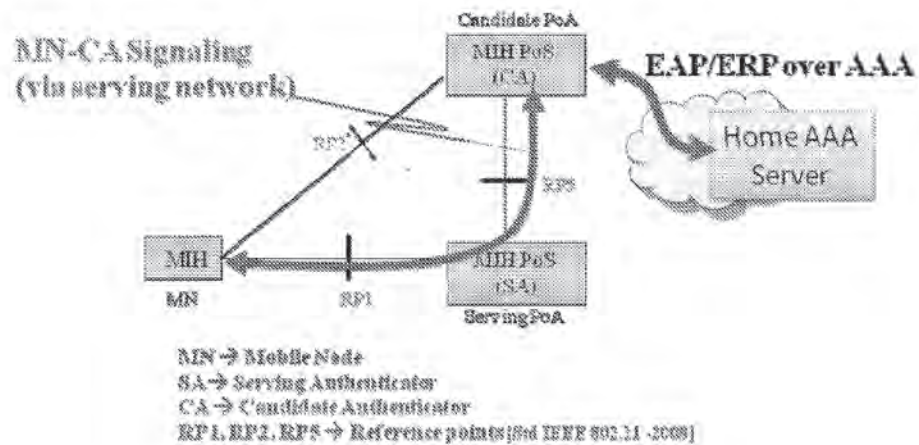


FIG. 5

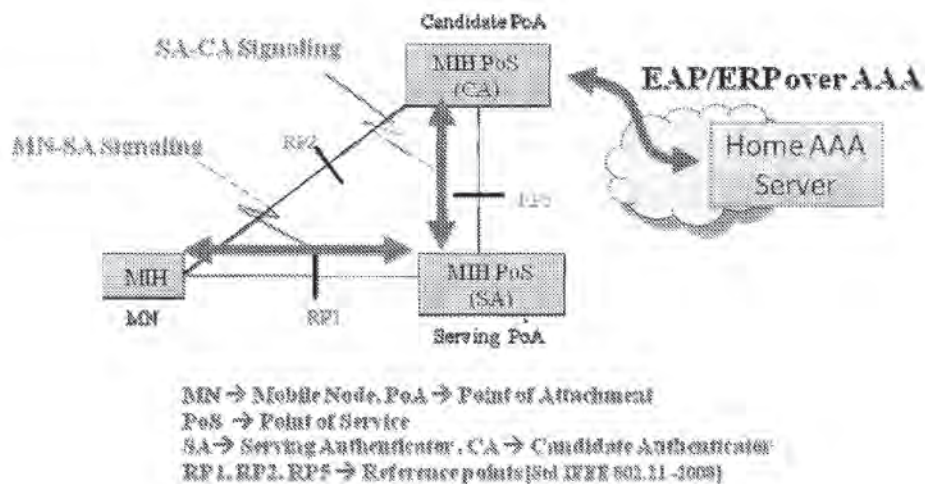


FIG. 6

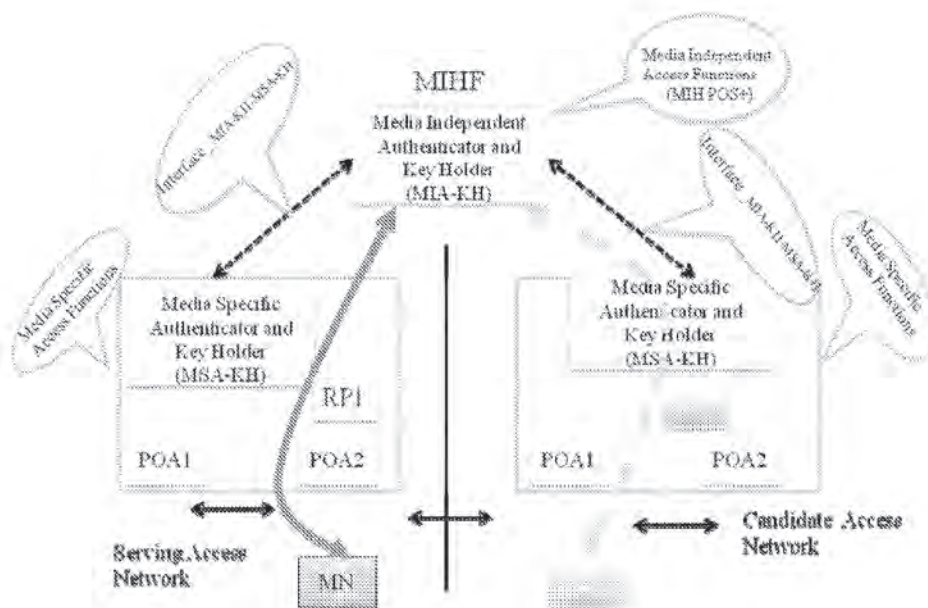


FIG. 7

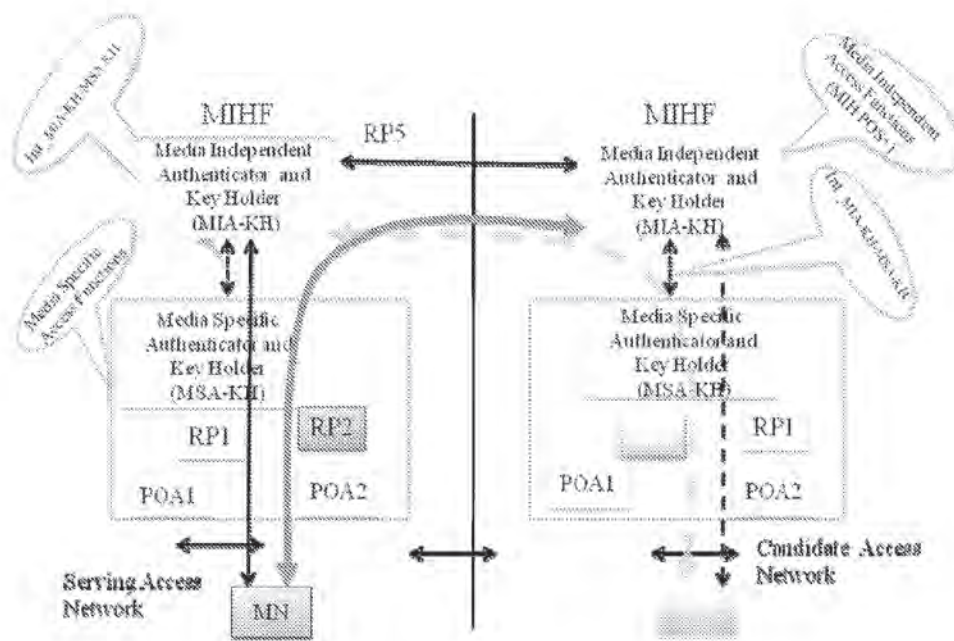


FIG. 8

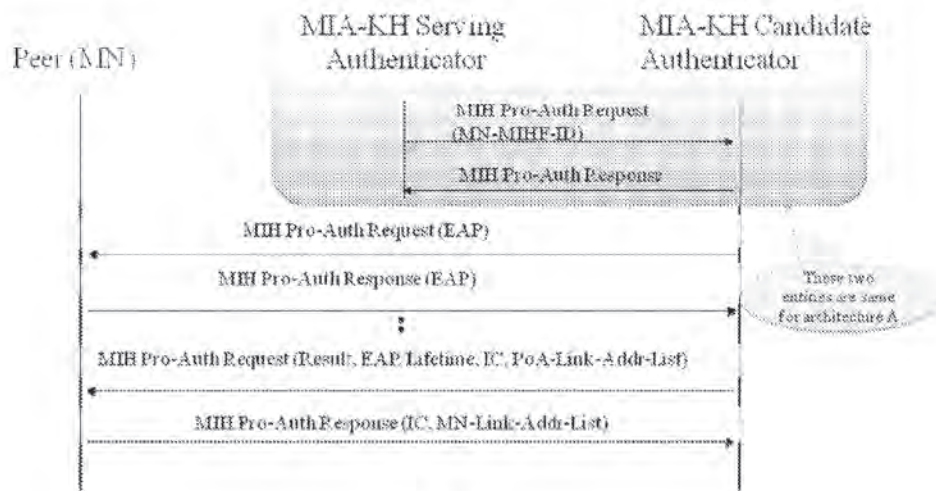


FIG. 9

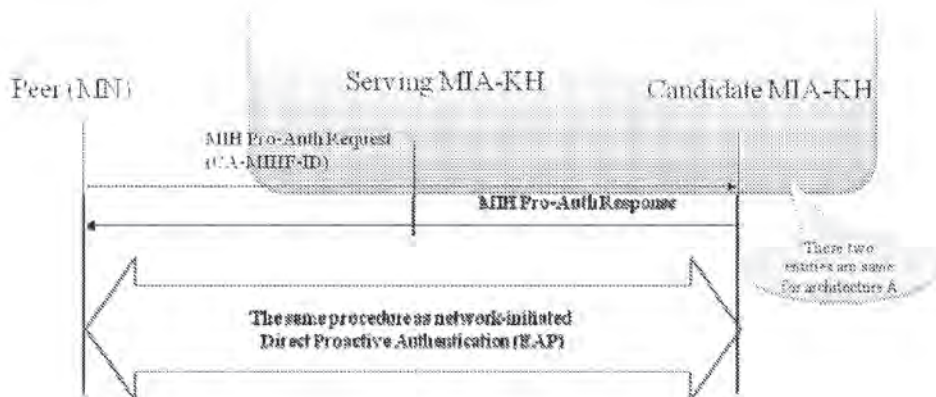


FIG. 10

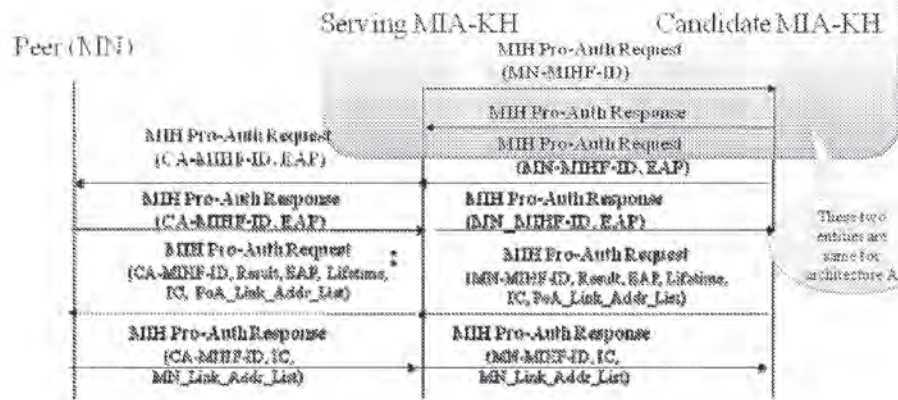


FIG. 11

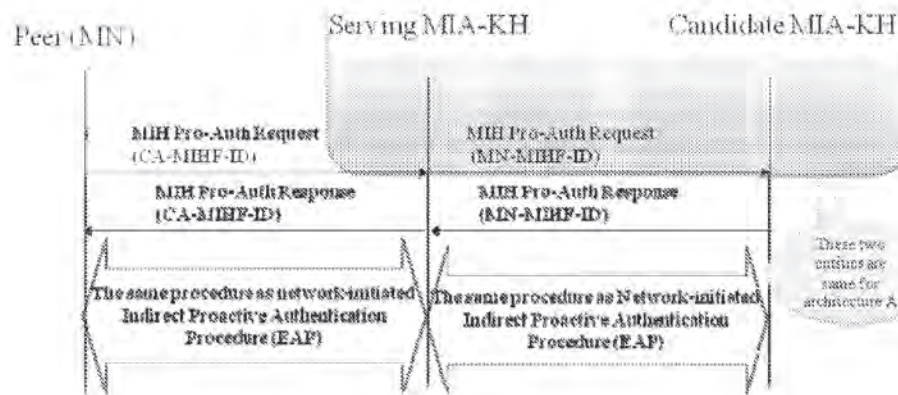


FIG. 12

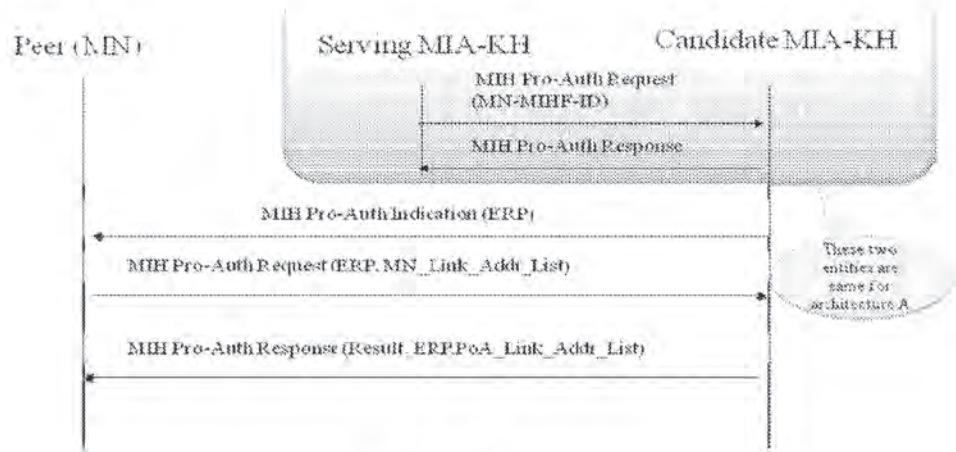


FIG. 13

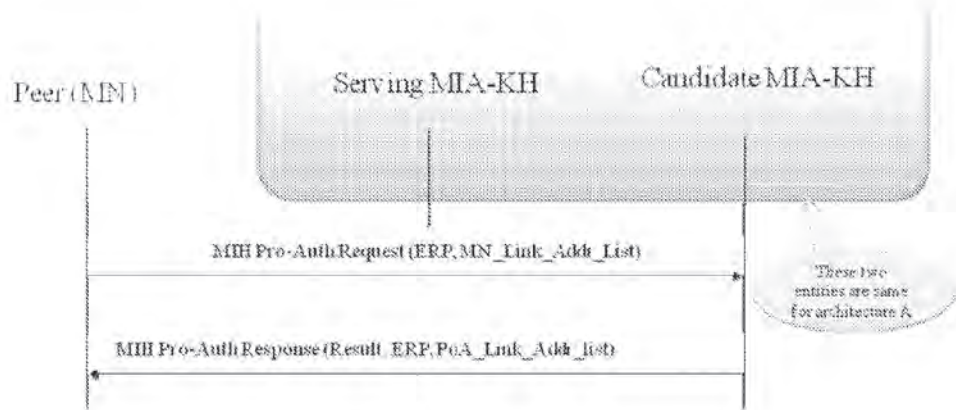


FIG. 14

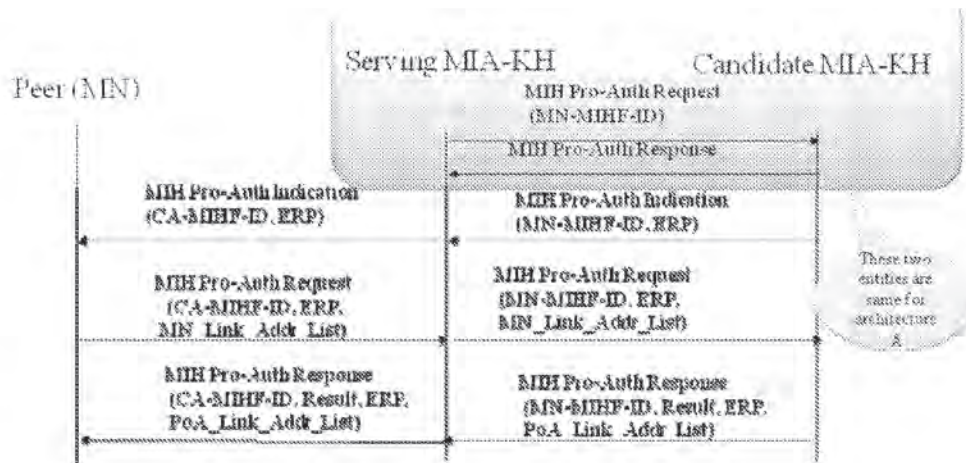


FIG. 15

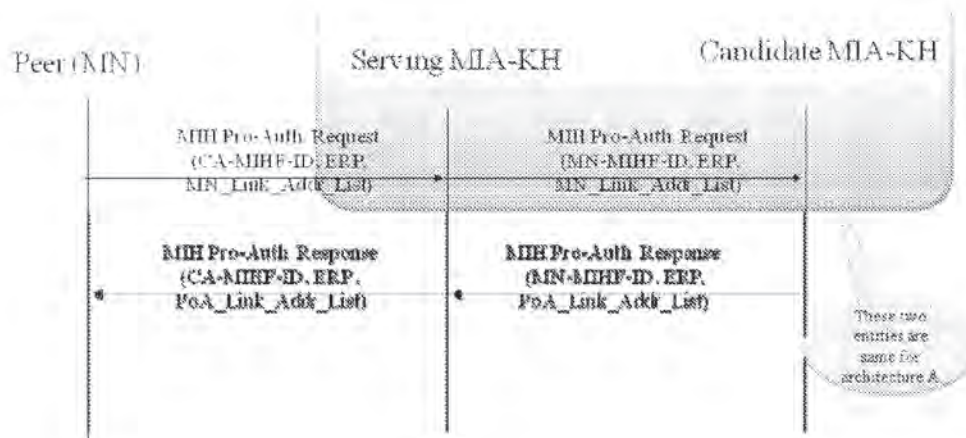


FIG. 16

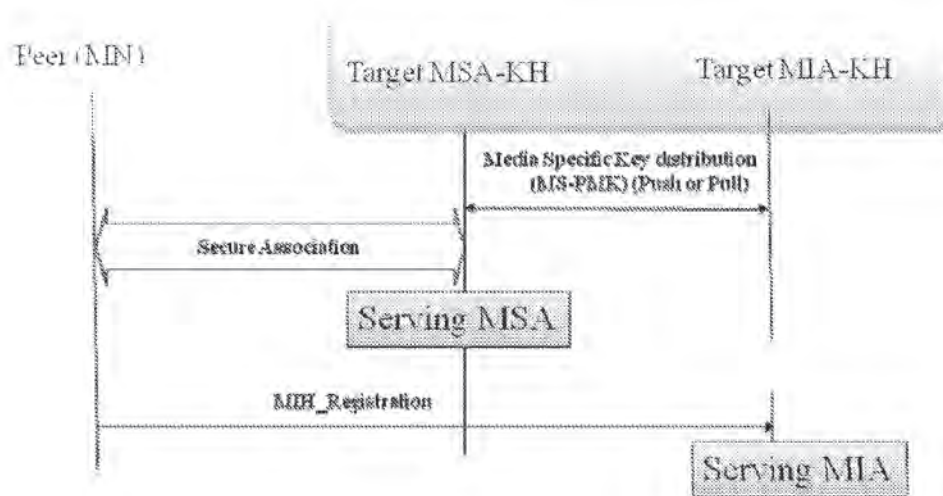


FIG. 17

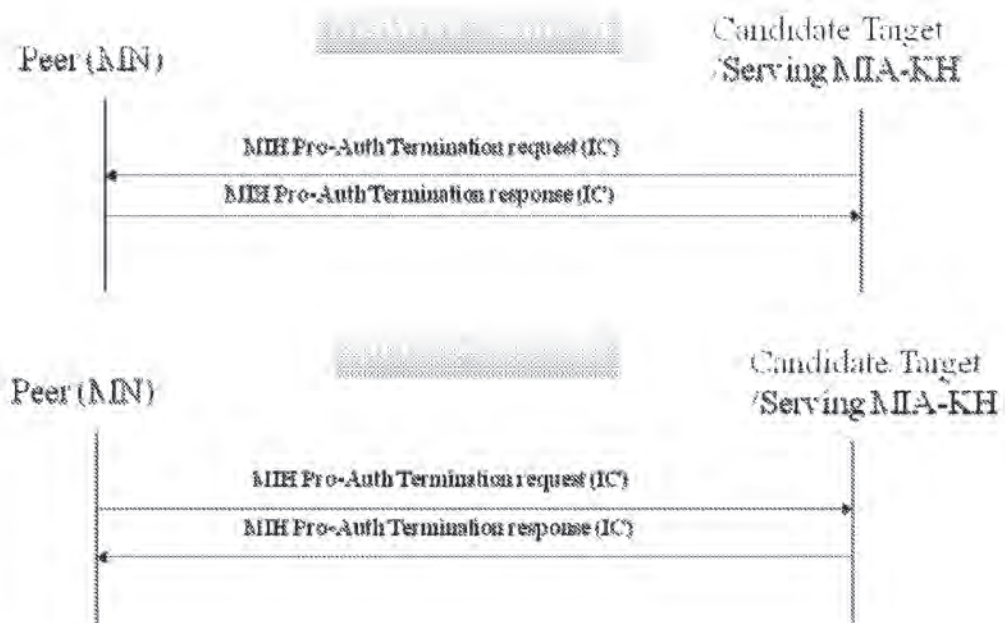


FIG. 18

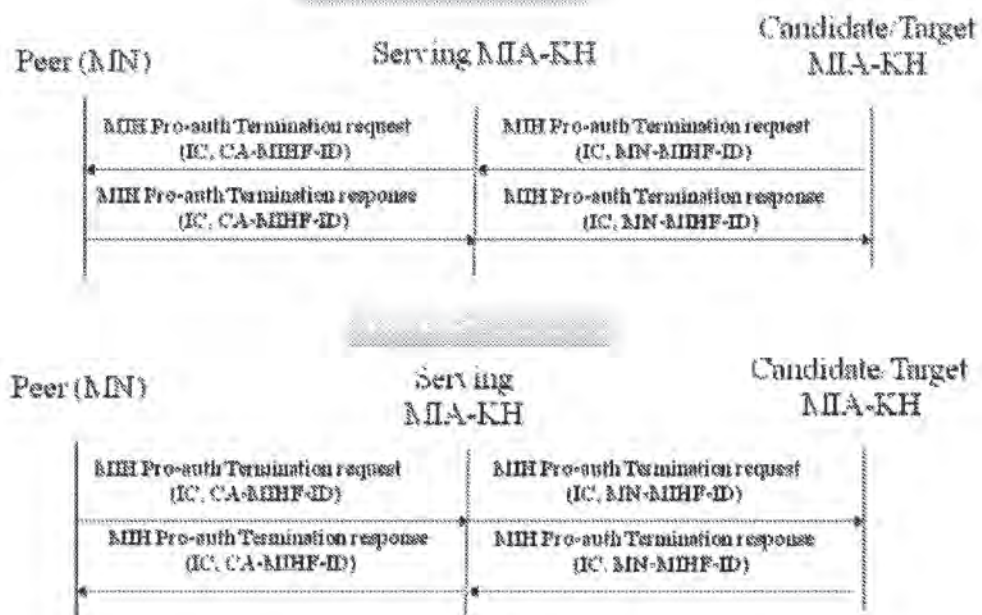


FIG. 19

PROACTIVE AUTHENTICATION**CROSS-REFERENCE TO RELATED APPLICATION**

[0001] This application claims the benefit of U.S. Provisional Applications No. 61/175,016, filed May 3, 2009 and 61/221,551, filed Jun. 29, 2009, the entire disclosures of which are incorporated herein by reference.

BACKGROUND

[0002] 1. Field

[0003] This invention relates to the security signaling between heterogeneous networks and more particularly, to a concept of authentication and authorization to perform a secure handover between a mobile node (MN) and a target network.

[0004] 2. Description of Related Art

[0005] Networks and Internet Protocol;

[0006] There are many types of computer networks, with the Internet having the most notoriety. The Internet is a world-wide network of computer networks. Today, the Internet is a public and self-sustaining network that is available to many millions of users. The Internet uses a set of communication protocols called TCP/IP (i.e., Transmission Control Protocol/Internet Protocol) to connect hosts. The Internet has a communications infrastructure known as the Internet backbone. Access to the Internet backbone is largely controlled by Internet Service Providers (ISPs) that resell access to corporations and individuals.

[0007] With respect to IP (Internet Protocol), this is a protocol by which data can be sent from one device (e.g., a phone, a PDA [Personal Digital Assistant], a computer, etc.) to another device on a network. There are a variety of versions of IP today, including, e.g., IPv4, IPv6, etc. Each host device on the network has at least one IP address that is its own unique identifier. IP is a connectionless protocol. The connection between end points during a communication is not continuous. When a user sends or receives data or messages, the data or messages are divided into components known as packets. Every packet is treated as an independent unit of data.

[0008] In order to standardize the transmission between points over the Internet or the like networks, an OSI (Open Systems Interconnection) model was established. The OSI model separates the communications processes between two points in a network into seven stacked layers, with each layer adding its own set of functions. Each device handles a message so that there is a downward flow through each layer at a sending end point and an upward flow through the layers at a receiving end point. The programming and/or hardware that provides the seven layers of function is typically a combination of device operating systems, application software, TCP/IP and/or other transport and network protocols, and other software and hardware.

[0009] Typically, the top four layers are used when a message passes from or to a user and the bottom three layers are used when a message passes through a device (e.g., an IP host device). An IP host is any device on the network that is capable of transmitting and receiving IP packets, such as a server, a router or a workstation. Messages destined for some other host are not passed up to the upper layers but are forwarded to the other host. The layers of the OSI model are listed below. Layer 7 (i.e., the application layer) is a layer at which, e.g., communication partners are identified, quality of

service is identified, user authentication and privacy are considered, constraints on data syntax are identified, etc. Layer 6 (i.e., the presentation layer) is a layer that, e.g., converts incoming and outgoing data from one presentation format to another, etc. Layer 5 (i.e., the session layer) is a layer that, e.g., sets up, coordinates, and terminates conversations, exchanges and dialogs between the applications, etc. Layer-4 (i.e., the transport layer) is a layer that, e.g., manages end-to-end control and error-checking, etc. Layer-3 (i.e., the network layer) is a layer that, e.g., handles routing and forwarding, etc. Layer-2 (i.e., the data-link layer) is a layer that, e.g., provides synchronization for the physical level, does bit-stuffing and furnishes transmission protocol knowledge and management, etc. The Institute of Electrical and Electronics Engineers (IEEE) sub-divides the data-link layer into two further sub-layers, the MAC (Media Access Control) layer that controls the data transfer to and from the physical layer and the LLC (Logical Link Control) layer that interfaces with the network layer and interprets commands and performs error recovery. Layer 1 (i.e., the physical layer) is a layer that, e.g., conveys the bit stream through the network at the physical level. The IEEE sub-divides the physical layer into the PLCP (Physical Layer Convergence Procedure) sub-layer and the PMD (Physical Medium Dependent) sub-layer.

[0010] Wireless Networks:

[0011] Wireless networks can incorporate a variety of types of mobile devices, such as, e.g., cellular and wireless telephones, PCs (personal computers), laptop computers, wearable computers, cordless phones, pagers, headsets, printers, PDAs, etc. For example, mobile devices may include digital systems to secure fast wireless transmissions of voice and/or data. Typical mobile devices include some or all of the following components: a transceiver (i.e., a transmitter and a receiver, including, e.g., a single chip transceiver with an integrated transmitter, receiver and, if desired, other functions); an antenna; a processor; one or more audio transducers (for example, a speaker or a microphone as in devices for audio communications); electromagnetic data storage (such as, e.g., ROM, RAM, digital data storage, etc., such as in devices where data processing is provided); memory; flash memory; a full chip set or integrated circuit; interfaces (such as, e.g., USB, CODEC, UART, PCM, etc.); and/or the like.

[0012] Wireless LANs (WLANs) in which a mobile user can connect to a local area network (LAN) through a wireless connection may be employed for wireless communications. Wireless communications can include, e.g., communications that propagate via electromagnetic waves, such as light, infrared, radio, microwave. There are a variety of WLAN standards that currently exist, such as, e.g., Bluetooth, IEEE 802.11, and HomeRF.

[0013] By way of example, Bluetooth products may be used to provide links between mobile computers, mobile phones, portable handheld devices, personal digital assistants (PDAs), and other mobile devices and connectivity to the Internet. Bluetooth is a computing and telecommunications industry specification that details how mobile devices can easily interconnect with each other and with non-mobile devices using a short-range wireless connection. Bluetooth creates a digital wireless protocol to address end-user problems arising from the proliferation of various mobile devices that need to keep data synchronized and consistent from one device to another, thereby allowing equipment from different vendors to work seamlessly together. Bluetooth devices may be named according to a common naming concept. For

example, a Bluetooth device may possess a Bluetooth Device Name (BDN) or a name associated with a unique Bluetooth Device Address (BDA). Bluetooth devices may also participate in an Internet Protocol (IP) network. If a Bluetooth device functions on an IP network, it may be provided with an IP address and an IP (network) name. Thus, a Bluetooth Device configured to participate on an IP network may contain, e.g., a BDN, a BDA, an IP address and an IP name. The term "IP name" refers to a name corresponding to an IP address of an interface.

[0014] An IEEE standard, IEEE 802.11, specifies technologies for wireless LANs and devices. Using 802.11, wireless networking may be accomplished with each single base station supporting several devices. In some examples, devices may come pre-equipped with wireless hardware or a user may install a separate piece of hardware, such as a card, that may include an antenna. By way of example, devices used in 802.11 typically include three notable elements, whether or not the device is an access point (AP), a mobile station (STA), a bridge, a PCMCIA card or another device: a radio transceiver; an antenna; and a MAC (Media Access Control) layer that controls packet flow between points in a network.

[0015] In addition, Multiple Interface Devices (MIDs) may be utilized in some wireless networks. MIDs may contain two independent network interfaces, such as a Bluetooth interface and an 802.11 interface, thus allowing the MID to participate on two separate networks as well as to interface with Bluetooth devices. The MID may have an IP address and a common IP (network) name associated with the IP address.

[0016] Wireless network devices may include, but are not limited to Bluetooth devices, Multiple Interface Devices (MIDs), 802.11x devices (IEEE 802.11 devices including, e.g., 802.11a, 802.11b and 802.11g devices), HomeRF (Home Radio Frequency) devices, Wi-Fi (Wireless Fidelity) devices, GPRS (General Packet Radio Service) devices, 3G cellular devices, 2.5G cellular devices, GSM (Global System for Mobile Communications) devices, EDGE (Enhanced Data for GSM Evolution) devices, TDMA type (Time Division Multiple Access) devices, or CDMA type (Code Division Multiple Access) devices, including CDMA2000. Each network device may contain addresses of varying types including but not limited to an IP address, a Bluetooth Device Address, a Bluetooth Common Name, a Bluetooth IP address, a Bluetooth IP Common Name, an 802.11 IP Address, an 802.11 IP common Name, or an IEEE MAC address.

[0017] Wireless networks can also involve methods and protocols found in, e.g., Mobile IP (Internet Protocol) systems, in PCS systems, and in other mobile network systems. With respect to Mobile IP, this involves a standard communications protocol created by the Internet Engineering Task Force (IETF). With Mobile IP, mobile device users can move across networks while maintaining their IP Address assigned once. See Request for Comments (RFC) 3344. NB: RFCs are formal documents of the Internet Engineering Task Force (IETF). Mobile IP enhances Internet Protocol (IP) and adds means to forward Internet traffic to mobile devices when connecting outside their home network. Mobile IP assigns each mobile node a home address on its home network and a care-of-address (CoA) that identifies the current location of the device within a network and its subnets. When a device is moved to a different network, it receives a new care-of address. A mobility agent on the home network can associate each home address with its care-of address. The mobile node

can send the home agent a binding update each time it changes its care-of address using, e.g., Internet Control Message Protocol (ICMP).

[0018] In basic IP routing (e.g., outside mobile IP), routing mechanisms rely on the assumptions that each network node always has a constant attachment point to, e.g., the Internet and that each node's IP address identifies the network link it is attached to. In this document, the terminology "node" includes a connection point, which can include, e.g., a redistribution point or an end point for data transmissions, and which can recognize, process and/or forward communications to other nodes. For example, Internet routers can look at, e.g., an IP address prefix or the like identifying a device's network. Then, at a network level, routers can look at, e.g., a set of bits identifying a particular subnet. Then, at a subnet level, routers can look at, e.g., a set of bits identifying a particular device. With typical mobile IP communications, if a user disconnects a mobile device from, e.g., the Internet and tries to reconnect it at a new subnet, then the device has to be reconfigured with a new IP address, a proper netmask and a default router. Otherwise, routing protocols would not be able to deliver the packets properly.

[0019] Media Independent Handover Services:

[0020] In I.E.E.E. P802.21/D.01.09, September 2006, entitled Draft IEEE Standard for Local and Metropolitan Area Networks: Media Independent Handover Services, among other things, the document specifies 802 media access-independent mechanisms that optimize handovers between 802 systems and cellular systems. The I.E.E.E. 802.21 standard defines extensible media access independent mechanisms that enable the optimization of handovers between heterogeneous 802 systems and may facilitate handovers between 802 systems and cellular systems.

[0021] The scope of the IEEE 802.21 (Media Independent Handover) standard is to develop a specification that provides link layer intelligence and other related network information to upper layers to optimize handovers between heterogeneous media. This includes links specified by 3GPP, 3GPP2 and both wired and wireless media in the IEEE 802 family of standards. Note, in this document, unless otherwise noted, "media" refers to method/mode of accessing a telecommunication system (e.g. cable, radio, satellite, etc.), as opposed to sensory aspects of communication (e.g. audio, video, etc.). See 1.1 of I.E.E.E. P802.21/D.01.09, September 2006, entitled Draft IEEE Standard for Local and Metropolitan Area Networks: Media Independent Handover Services, the entire contents of which document is incorporated herein into and as part of this patent application. In addition, the provisional application from which priority is claimed herein also incorporated Draft 05 (Referred to herein as D05) of said standard and I.E.E.E. standard 802.21 published on Jan. 21, 2009, the entire contents of which is again incorporated herein by reference—i.e., see, e.g., I.E.E.E. P802.21/D05.00, April, 2007, Draft Standard for Local and Metropolitan Area Networks: Media Independent Handover Services. Sponsored by the LAN MAN Standards Committee of the I.E.E.E. Computer Society and I.E.E.E. standard for Local and Metropolitan Area Networks—Part 21: Media Independent Handover Services published on Jan. 21, 2009. Sponsored by the LAN MAN Standards Committee of the I.E.E.E. Computer Society respectively.

[0022] General Architecture:

[0023] Introduction:

[0024] The IEEE 802.21 standard is intended to facilitate a variety of handover methods. Such methods are generally classified as 'hard' or 'soft', depending on whether the handover procedure is "break before make" or "make before break" with respect to the data transport facilities that support the exchange of data packets between the mobile node and the network.

[0025] In general, handover involves cooperative use of both mobile node and network infrastructure in order to satisfy network operator and end user needs. Handover control, handover policies and other algorithms involved in handover decision making are generally handled by communication system elements which do not fall within the scope of the IEEE 802.21 standard. However, it is beneficial to describe certain aspects of the overall handover procedure so that the role and purpose of MIH Event Service, MIH Command Service, MIH Information Service and MIHF in overall handover process is clear.

[0026] General Design Principles:

[0027] The IEEE 802.21 standard is based on the following general design principles.

[0028] a) MIH Function is logical entity that helps and facilitates handover decision-making. Upper layers make handover decisions and link selection based on inputs and context from MIHF. Facilitating the recognition that a handover should take place is one of the key goals of MIHF. Discovery of information on how to make effective handover decisions is also a key component.

[0029] b) MIHF provides abstracted services to higher layers. From that perspective MIHF offers a unified interface to the upper layers. The service primitives exposed by this unified interface are based on the technology specific protocol entities of the different access networks. The MIHF communicates with the lower layers of the mobility-management protocol stack through technology-specific interfaces.

[0030] The specification of the MIHF interfaces with the lower layers generally does not fall within the scope of this standard. Such interfaces may already be specified as service access points (SAPs) within the standards that pertain to the respective access technologies, such as IEEE 802.1, IEEE 802.3, IEEE 802.11, IEEE 802.16, 3GPP and 3GPP2. This standard may contain recommendations to amend the existing access technology specific standards when modifications of the lower-layer interfaces may enable or enhance MIHF functionality.

[0031] c) Handover signaling (as part of handover execution and subsequent updates) may not be part of the standard. Different access networks support horizontal handover mechanisms (mobile initiated, network initiated, etc.). Handover initiation trigger may be useful in heterogeneous handovers when not done as per the homogeneous scheme.

[0032] d) MIHF may do further processing on MAC/PHY triggers and other related local events. Definition of this processing is outside the scope of the standard. The standard shall provide support for remote events as well. Events are advisory in nature. The decision whether to cause a handover or not based on these events is outside the scope of the standard.

[0033] e) The standard shall specify mechanisms to support MN-initiated, MN-controlled, network-initiated and network-controlled handovers.

[0034] f) The standard may support transparent inter-working with legacy equipment. Thus IEEE 802.21 compatible equipment should be able to co-exist with legacy non IEEE 802.21 compliant equipment.

[0035] Media Independent Handover Reference Framework:

[0036] The following sections describe aspects with regards to communication between different MIHF entities in the client device (MN) and the network.

[0037] MIHF functions communicate with each other for various purposes. The client device (mobile node) exchanges MIH information with its MIH Point of Service. The MIHF in any Network Entity becomes an MIH PoS when it communicates directly with a MN based MIHF. An MIH Network Entity may not have a direct connection to the MN and therefore does not constitute an MIH PoS for that particular MN. The same MIH Network Entity may still act as MIH PoS for a different MN. MIHF communication may not take place on all L2 interfaces of an MIH capable MN. As an example, on an MIH capable MN with three L2 interfaces namely 802.11, 802.16, and 802.3, the 802.3 interface may be used only for system administration and maintenance operations, while the 802.11 and 802.16 interfaces may engage in the provision of MIHF services. The MN may use L2 transport for exchanging MIH information with an MIH PoS that resides in the same Network Entity as its Network PoA. The MN may use L3 transport for exchanging MIH information with an MIH PoS that may not reside in the same Network Entity as its Network PoA. The framework supports use of either L2 or L3 mechanisms toward communication among MIH network entities.

[0038] FIG. 3 shows an MIH communication model from the 802.21 standard. The model shows MIHFs in different distinctive roles and the communication relationships amongst them. The communication relationship shown in FIG. 3 applies only to MIHFs. It is noteworthy that each of the communication relationships in the communication model does not imply a particular transport mechanism. Rather, a communication relationship only intends to show that MIHF related information passing is possible between the two distinctive MIHFs. Moreover, 1) MIHF on the MN,

2) MIH PoS on the Network Entity that includes the serving PoA of the MN, 3) MIH PoS on the Network Entity that includes a candidate PoA for the MN (a candidate PoA is a PoA that the MN is aware of but not currently attached to; it becomes the target PoA if a handover eventually occurs), 4) MIH PoS on a Network Entity that does not include a PoA for the MN, 5) MIH non-PoS on a Network Entity that does not include a PoA for the MN.

[0039] The communication model also identifies the following communication reference points between different instances of MIHFs.

[0040] 1) Communication reference point R1: Reference Point R1 refers to MIHF procedures between the MIHF on the MN and the MIH PoS on the Network Entity of its serving PoA. R1 may encompass communication interfaces over both L2 and L3 and above. MIHF content passed over R1 may be related to MIIS, MIES, or MICS.

[0041] 2) Communication reference point R2: Reference Point R2 refers to MIHF procedures between the MIHF on the MN and the MIH PoS on the Network Entity of a candidate PoA. R2 may encompass communication interfaces over both L2 and L3 and above. MIHF content passed over R2 may be related to MIIS, MIES, or MICS.

[0042] 3) Communication reference point R3: Reference Point R3 refers to MIHF procedures between the MIHF on the MN and the MIH PoS on a non-PoA Network Entity. R3 may encompass communication interfaces over L3 and above and possibly L2 transport protocol like Ethernet bridging, MPLS, etc. MIHF content passed over R3 may be related to MIIS, MIES, or MICS.

[0043] 4) Communication reference point R4: Reference Point R4 refers to MIHF procedures between an MIH PoS in a Network Entity and an MIH non-PoS instance in another Network Entity. R4 may encompass communication interfaces over L3 and above. MIHF content passed over R4 may be related to MIIS, MIES, or MICS.

[0044] 5) Communication reference point R5: Reference Point R5 refers to MIHF procedures between two MIH PoS instances in distinct Network Entities. R5 may encompass communication interfaces over L3 and above. MIHF content passed over R5 may be related to MIIS, MIES, or MICS.

[0045] Illustration of the MIH Communication Model:

[0046] A network model including MIH services is shown in FIG. 4 for greater illustration of the MIH Communication Reference Points. Moving from right to left, the model includes an MIH-capable mobile node (MN, far right) that supports multiple wired and wireless access technology options. The model assumes that the provisioning service provider either operates multiple access technologies or allows its user to roam into other networks when SLA in support of inter-working has been established. The MN has an MIHF implemented, which enables it to send specific MIH queries. The MN may have the information service partially implemented internally.

[0047] The model illustrates access networks that are connected in some loose, serial way to a core network (Operator 1-3 Core). Also depicted is an access network that is more tightly interworked or coupled (Access Network-3). The Operator 1-3 Core each might represent a service provider, corporate intranet provider or just another part of the visited or home access, or even core network. In this model the provisioning provider is operating Access Network-3 coupled via R1 to a core (labeled Visited/Home Core Network). The terms Visited and Home are used to indicate the provisioning service provider or enterprise. Any of the illustrated networks could be both a visited or home network depending on the relation of the operator to the provisioner of the MN. Network providers offer MIH services in their access networks (Access Network-1 to 4) to facilitate handover into their networks. Each access technology either advertises its MIH capability or responds to MIH service discovery. Each service provider for the access network allows access to one or more MIH Points of Service (PoS, compare with Communication Model) These PoS may provide some or all of the MIH services as determined during MIH capabilities discovery. The location or node of an MIH PoS is not fixed by the standard. The PoS location may vary based on operator deployment scenario and the technology-specific MIH architecture.

[0048] An MIH PoS may reside next to or be co-located with the point of attachment (PoA) in the access network (Access Network 1,2,4 are typical). Alternatively the PoS may reside deeper inside the access or core networks (Access Network 3 is typical). As shown in FIG. 3, the MIH entity in the MN communicates with MIH network entities either by R1, R2 or R3 over any access network. When the PoA in the serving access network has a co-located MIH function, then

an R1 reference connection terminates at the PoA which is also the PoS (MN to Access Network 1,2,4 of the model could all be R1) In that case an R3 reference connection would be terminated at any non-PoA (also illustrated by MN to Access Networks 1,2,4). MIH events may originate at both sides of an active R1 link. The MN is typically the first node to react to these events.

[0049] The interaction of visited and home network could be either for control and management purposes or for data transport purposes. It is also possible that due to roaming or SLA agreements, the home network may allow the MN to access the public Internet directly through a visited network. As illustrated, two MIH network entities may communicate with each other via R4 or R5 reference connections. The MIH capable PoA may also communicate with other MIH network entities via R3 and R4 reference points. The MIH capable MN could have a MIH communication with other PoA in the candidate access networks via R2 reference point to obtain information services about the candidate network.

[0050] With regard to the MIH Information Service (MIIS) the providers offer access to their information server located in a MIH PoS node (upper far left). The operator provides the MIIS to mobile nodes so they can obtain pertinent information including but not limited to new roaming lists, costs, provider identification information, provider services, priorities and any other information that would enable to select and utilize services. As illustrated, it is possible for the mobile node to be pre-provisioned with MIIS data by its provider.

[0051] Also possible is for the mobile node to obtain MIH information services from any access network of its provider. MIIS could also be available from another overlapping or nearby network, using that network's MIIS point of service. A provisioner's network (depicted here as coupled with Access Network 3) may utilize R3 and R4 interfaces to access other MIH entities like the provisioner's or visited network's MIH information server.

[0052] With regard to the MIH Command Service (MICS), any of the Information Database may be also used as command service PoS. The MN MIHF typically communicates with this server using a layer three transport.

[0053] MIHF Services:

[0054] The MIHF provides asynchronous and synchronous services through well defined SAPs for link layers and MIH users. In the case of a system with multiple network interfaces of arbitrary type, the upper layers may use the Event service, Command service and Information service provided by MIH to manage, determine, and control the state of the underlying interfaces.

[0055] These services provided by MIH help the upper layers in maintaining service continuity, service adaptation to varying quality of service, battery life conservation, and network discovery and link selection. In a system containing heterogeneous network interfaces of 802 types and cellular 3GPP, 3GPP2 types, the Media Independent Handover Function may help the upper layers to implement effective procedures to couple services across heterogeneous network interfaces. Upper layers may utilize services provided by the MIHF across different entities to query resources required for a handover operation between heterogeneous networks.

[0056] MIH services in mobile devices facilitate seamless handover between heterogeneous networks. An MIH user such as a mobility management protocol (example Mobile IP) could be supported for handover and seamless session continuity. This shall not preclude other protocols in addition to

Mobile IP and even other upper layers from making use of MIH services to optimize handovers.

[0057] Mobile nodes employing MIH services would receive indications from link layers for asynchronous operations like Event service. The interaction with Command service and Information service will be through synchronous query and response type of mechanisms.

The MIHF would also provide the functionality for the exchange of information between the network and host entities of the same media type. Note, if mechanism for such information exchange already exists with a given type of media (such as with some cellular media types), the MIHF will make use of the existing mechanism whenever possible.

[0058] MIH Protocol:

[0059] The IEEE 802.21 standard supports the Media Independent Event service, Media Independent Command service and Media Independent Information service. The MIH Protocol defines the format of the messages (i.e. MIHF packet with header and payload) that are exchanged between remote MIHF entities and the transport mechanisms that support the delivery of the messages. The selection of the transport mechanism is dependent on the access technology that connects the MN to the network and the location of the MIH PoS.

[0060] The packet payload for these services may be carried over L2 management frames, L2 data frames or other higher layer protocols. Wireless networks such as 802.11 and 802.16 have a management plane and support management frames which could be suitably enhanced for carrying the above payloads. However, the wired Ethernet network does not have management plane and may carry the above payloads only in data frames.

[0061] The IEEE 802.21 standard defines the packet format and payloads in media independent manner in standard TLV format. Thereafter, these packets may be encapsulated in a L2 MIH Protocol using MIHF ethertype when the payload needs to be sent over normal data frames as in case of Ethernet. In other cases the TLV based messages and payload may be directly encapsulated in media specific management frames. Alternatively, MIH protocol messages may be encapsulated using a lower layer (L2) or a higher layer (L3 and above) transport.

[0062] The IEEE 802.21 standard defines the format of MIH Protocol data unit (PDU) header and payload. Standard TLV format provides media-independent representation for the PDU payload contents. The MIHF PDUs are encapsulated in data frames with MIHF ethertype over 802 links. For 802.11 and 802.16 links extensions of media-specific management frames are recommended for carrying MIH messages. No assumptions are made in this standard regarding the transport of MIH messages over 3GPP and 3GPP2 access links at L2.

[0063] Media Independent Information Service:

[0064] Introduction:

[0065] Media Independent Information Service (MIIS) provides a framework by which an MIHF both in the mobile node and in the network may discover and obtain network information within a geographical area to facilitate handovers. The objective is to acquire a global view of all the heterogeneous networks relevant to the MN in the area to facilitate seamless handovers when roaming across these networks.

[0066] Media Independent Information Service includes support for various Information Elements (IEs). Information

Elements provide information that is essential for a network selector to make intelligent handover decision.

[0067] Depending on the type of mobility, support for different types of information elements may be necessary for performing handovers. For example in case of horizontal handovers across different PoAs of the same access network, information available from lower link layers of access network may be sufficient. In such cases information elements like intra-technology neighbor reports and other link layer information required during handovers is directly available from the access network. In such cases the availability of higher layer services offered by the network may not change appreciably across different network point of attachment.

[0068] On the other hand, during vertical handovers there is a need to select appropriate PoA in the new network based on both optimum link layer connectivity as well as availability of appropriate higher layer services to permit service and session continuity for active user applications.

[0069] Media Independent Information Service (MIIS) provides the capability for obtaining the necessary information for handovers. This includes information about lower layers such as neighbor maps and other link layer parameters as well as information about available higher layer services such as internet connectivity, availability of VPN services, etc. The set of different higher layer services provided by the MIIS may constantly evolve. At the same time the list of access networks that are supported by MIIS may also evolve. As such, there is a need for flexibility and extensibility in the way the MIIS provides support for different information elements. Towards this end the MIIS defines a schema. The schema helps a client of MIIS to discover the capabilities of MIIS and also discover the entire set of different access networks and IEs supported by a particular implementation. Schema representation also allows the mobile node to query the information in a more flexible and efficient manner. As part of defining this schema the MIIS may also identify a set of basic information elements that may define the core functionality of different implementations of MIIS. Other information elements as they are added may become part of the extended set of MIIS capabilities.

[0070] MIIS provides information about different access networks such as 802 networks, 3GPP networks and 3GPP2 networks. The MIIS also allows this collective information to be accessed from any single network.

[0071] Thus for example using an 802.11 access network, it may be possible to get information not only about all other 802 networks in a particular region but also that of 3GPP and 3GPP2 networks as well. Similarly using a 3GPP2 interface, it may be possible to get access to information about all 802 and 3GPP networks in a given region. This capability allows the mobile node to use its currently active access network and scan for other available access networks in a geographical region. Thus, a mobile node is freed from the burden of powering up each of its individual radios and establishing network connectivity for the purpose of accessing heterogeneous network information. MIIS enables this functionality across all available access networks by providing uniform way to retrieve heterogeneous network information in any geographical area.

[0072] Information Service Elements:

[0073] The main goal behind the Information service is to allow mobile node and network entities to discover information that may influence the selection of appropriate networks during handovers. This information is intended to be prima-

rily used by a policy engine entity that may make effective handover decisions based on this information. This information service is expected to provide mostly static type of information, although network configuration changes must also be accounted for. Other dynamic information about different access networks such as current available resource levels, state parameters, dynamic statistics etc. should be obtained directly from the respective access networks. Some of the key motivations behind the Information Service are as follows:

[0074] 1) Provide information about the availability of access networks in a geographical area. Further, this information could be retrieved using any wireless network, for example, information about a nearby WiFi hotspot could be obtained using a GSM, CDMA, or any other cellular network, whether by means of request/response signaling, or information that is specifically or implicitly broadcast over those cellular networks. Alternatively, this information could be maintained in an internal database by the MN.

[0075] 2) Provide static link layer information parameters that could help the mobile devices in selecting the appropriate access network. For example knowledge of whether security and QoS are supported on a particular access network may influence the decision to select such an access network during handovers.

[0076] 3) The link layer information comprising of neighbor reports and information about capabilities of different PoAs could also aid in configuring the radios optimally (to the extent possible) for connecting to available/selected access networks. For example knowing about supported channels by different PoAs may help in configuring the channels optimally as opposed to scanning, beaconing etc. and then finding out this information. However, for most part, dynamic link layer parameters have to be obtained or selected based on direct interaction with the access networks and the Information Service may not be able to help much in that regard.

[0077] 4) Provide an indication of higher layer services supported by different access networks and other relevant information that may aid in making handover decisions. Such information may not be available (or could not be made available) directly from MAC/PHY layers of specific access networks, but could be provided as part of the Information service. For example, in certain cases classification of different networks into categories such as public, enterprise, home, others, etc. may influence handover decision. Other information here may be more vendor/network specific in nature and could be specified in that form.

[0078] The information service elements are classified into three groups:

[0079] 1) General Access Network Information: These information elements give a general overview of the different networks providing coverage within an area such as list of available networks and their associated operators, roaming agreements between different operators, cost of connecting to the network and network security and quality of service capabilities.

[0080] 2) Information about Points of Attachment: These information elements provide information about different PoAs for each of the available access networks. These IEs include PoA addressing information, PoA location, data rates supported, the type of PHY and MAC layers and any channel parameters to optimize link layer connectivity. This may also include higher layer services and individual capabilities of different PoAs.

[0081] 3) Other Information may be vendor/network specific, and could be specified appropriately.

[0082] Media Independent Handover Protocol:

[0083] Introduction:

[0084] The MIHF provides asynchronous and synchronous services through well defined SAPs for lower layers and upper layers. The services provided include the Event Service (ES), Command Service (CS), and Information Service (IS). Detailed description about MIH services are found in the 802.21 draft document. MIH SAPs include the MIH upper layer SAP, which is used by the users of MIH to gain access to various MIHF services, and MIH lower layer SAPs, which are used by MIHF to gain access and control of a variety of media dependent lower layer resources.

[0085] The MIH protocol defines frame formats for exchanging messages between peer MIHF entities. These messages are based on the primitives which are part of Media Independent Event service, Media Independent Command service and Media Independent Information service. IEEE 802.21 supports Media Independent Handover Function in mobile node, and network. The MIH Protocol allows peer MIHF entities to interact with each other.

[0086] In order for mobile node's MIHF entity to commence MIH protocol procedures, MIHF entity of mobile node may discover its peer remote MIHF entities. Peer remote MIHF entity is the correspondent MIHF entity with which MIHF of mobile node exchanges MIH protocol messages. Because peer remote MIHF entities reside in anywhere of the network, MIHF entity of mobile node may discover MIHF entity in the network before initiating MIH protocol procedure. This is done through the MIH Function Discovery procedure.

[0087] MIH Function Discovery can be done either at Layer 2 or Layer 3. However, this document only specifies how MIH Function Discovery is performed at Layer 2, when both MIH Functions are located within the same broadcast domain. MIH Function Discovery may be performed either through the MIH protocol (i.e. using L2 encapsulation such as LLC) or through media specific Layer 2 broadcast messages (i.e. 802.11 beacons, 802.16 DCD). MIH Function Discovery at Layer 3 is outside of scope of 802.21.

[0088] Once the peer MIHF has been discovered, the MN may discover the capabilities of the peer MIHF. This is done through the MIH Capability Discovery procedure. MIH Capability Discovery may be performed either through the MIH protocol or through media specific Layer 2 broadcast messages (i.e. 802.11 beacons, 802.16 DCD).

[0089] When the peer MIHF resides within the same broadcast domain as the MN, MIH Function Discovery can be performed using only MIH Capability Discovery.

[0090] Illustrative Architecture:

[0091] FIG. 1 depicts some illustrative architectural components that can be employed in some illustrative and non-limiting implementations including wireless access points to which client devices communicate. In this regard, FIG. 1 shows an illustrative wireline network 20 connected to a wireless local area network (WLAN) generally designated 21. The WLAN 21 includes an access point (AP) 22 and a number of user stations 23, 24. For example, the wireline network 20 can include the Internet or a corporate data processing network. For example, the access point 22 can be a wireless router, and the user stations 23, 24 can be, e.g., portable computers, personal desk-top computers, PDAs, portable voice-over-IP telephones and/or other devices. The

access point 22 has a network interface 25 linked to the wireline network 21, and a wireless transceiver in communication with the user stations 23, 24. For example, the wireless transceiver 26 can include an antenna 27 for radio or microwave frequency communication with the user stations 23, 25. The access point 22 also has a processor 28, a program memory 29, and a random access memory 31. The user station 23 has a wireless transceiver 35 including an antenna 36 for communication with the access point station 22. In a similar fashion, the user station 24 has a wireless transceiver 38 and an antenna 39 for communication to the access point 22. By way of example, in some embodiments an authenticator could be employed within such an access point (AP) and/or a supplicant or peer could be employed within a mobile node or user station.

[0092] FIG. 2 shows an illustrative computer or control unit that can be used to implement computerized process steps, to be carried out by devices, such as, e.g., an access point, a user station, a source node or destination node in some embodiments. In some embodiments, the computer or control unit includes a central processing unit (CPU) 322, which can communicate with a set of input/output (I/O) device(s) 324 over a bus 326. The I/O devices 324 can include, for example, a keyboard, monitor, and/or other devices. The CPU 322 can communicate with a computer readable medium (e.g., conventional volatile or non-volatile data storage devices) 328 (hereafter "memory 328") over the bus 326. The interaction between a CPU 322, I/O devices 324, a bus 326, and a memory 328 can be like that known in the art. Memory 328 can include, e.g., data 330. The memory 328 can also store software 338. The software 338 can include a number of modules 340 for implementing the steps of processes. Conventional programming techniques may be used to implement these modules. Memory 328 can also store the above and/or other data file(s). In some embodiments, the various methods described herein may be implemented via a computer program product for use with a computer system. This implementation may, for example, include a series of computer instructions fixed on a computer readable medium (e.g., a diskette, a CD-ROM, ROM or the like) or transmittable to a computer system via an interface device, such as a modem or the like. A communication medium may be substantially tangible (e.g., communication lines) and/or substantially intangible (e.g., wireless media using microwave, light, infrared, etc.). The computer instructions can be written in various programming languages and/or can be stored in memory device(s), such as semiconductor devices (e.g., chips or circuits), magnetic devices, optical devices and/or other memory devices. In the various embodiments, the transmission may use any appropriate communications technology.

GENERAL BACKGROUND REFERENCES

[0093] To facilitate reference, the following general background applications are made of reference (such applications also incorporate herein by reference the entire disclosures of all of the following patent applications for background reference):

- [0094] 1) U.S. Provisional Application No. 60/891,349, filed Feb. 23, 2007, to K. Taniuchi, et al.;
- [0095] 2) U.S. Provisional Application No. 60/825,567, filed Sep. 13, 2006 to Y. A. Cheng, et al.;
- [0096] 3) U.S. application Ser. No. 11/567,134, filed Dec. 5, 2006, to Y. Oba, et al.;

- [0097] 4) U.S. application Ser. No. 11/563,000, filed Nov. 23, 2006, to Y. Oba, et al.;
- [0098] 5) U.S. application Ser. No. 11/558,922, filed Nov. 12, 2006, to Y. Oba, et al.;
- [0099] 6) U.S. application Ser. No. 11/460,616, filed Jul. 27, 2006, to Y. Oba, et al.;
- [0100] 7) U.S. application Ser. No. 11/279,856, filed Apr. 14, 2006, to A. Dutta, et al., entitled Framework Of Media-Independent Pre-Authentication Improvements;
- [0101] 8) Including Considerations For Failed Switching And Switchback;
- [0102] 9) U.S. application Ser. No. 11/308,175, filed Mar. 9, 2006, to Y. Oba, et al., Framework of Media Independent Pre-Authentication Support for PANA;
- [0103] 10) U.S. application Ser. No. 11/307,362, filed February 2006, to A. Dutta, et al., entitled A Framework of Media-Independent Pre-authentication;
- [0104] 11) U.S. application Ser. No. 12/119,048, filed May 12, 2008, to Y. Oba, et al.

[0105] As noted above, the media independent solution for handovers does not address the security signaling requirement between heterogeneous networks. To perform a secure handover by mobile nodes (MN) attached between heterogeneous media supported networks, the authentication, authorization and confidentiality between a mobile node and the candidate network (target network) is required. However, the addition of security-related signaling can add significant delays to handover efforts which can affect time sensitive applications such as Voice over Internet Protocol (VoIP) for delivery of voice communications over IP networks such as the Internet or other packet-switched networks. In many cases, service continuity can no longer be met because of the impact on quality of service (QoS), which ultimately impacts the user experience, too great.

SUMMARY

[0106] In some exemplary embodiments, the problem of adverse impact on quality of service and the user experience due to the addition of security-related signaling over the media independent handover protocols is solved by proactive authentication techniques and MIH protocol level security mechanisms, among other things, by optimizing the security-related signaling during handovers.

[0107] In one aspect of this application, a system for proactively authenticating includes a server having media independent access functions including media independent authentication functions that authenticates other entities attached via an interface to an end of a link specific to a media. A plurality of heterogeneous networks each having media specific access functions including authentication functions corresponding to the other entities attached via the interface to the end of the link specific to the media and mobile devices connected to the plurality of heterogeneous networks, and the server having predefined media independent handover protocols and media independent handover identities based on the media independent functions related to handover, in which the server authenticates candidate access networks prior to the handover of the mobile devices from serving access networks to the candidate access networks each of which belonging to the plurality of heterogeneous access networks having the link specific to the media.

[0108] In another aspect of this application, an article of manufacture includes a machine-accessible medium having instructions encoded thereon for enabling a processor to

execute an operation for proactively authenticating mobile devices attached between heterogeneous media via media independent authenticator, comprising the steps of: assigning media specific authentication procedures to mobile devices attached to a serving access network within a plurality of heterogeneous access networks; providing information regarding candidate access networks by way of the media independent authenticator to mobile devices during a handover preparation stage; performing a proactive authentication using a interface; and executing, upon successfully performing the proactive authentication by way of the media independent authenticator, the handover to a candidate access network having the media specific authentication procedure.

[0109] In yet another aspect of this application, an apparatus having computer implemented media independent access functions includes an independent authenticator, the independent authenticator to authenticate mobile devices attached between heterogeneous media; an interface to connect the independent authenticator to a plurality of heterogeneous access networks each of which supports a specific authenticator that corresponds to a specific serving media; and pre-defined media independent handover protocols and media independent handover identities based on the media independent functions on the computer, wherein the independent authenticator authenticates candidate access networks prior to the handover of the mobile devices from serving access networks to the candidate access networks each of which belonging to the plurality of heterogeneous access networks having the specific serving media, wherein media specific keys are pulled or pushed from the specific authenticator of a serving access network and a candidate access network via a specific interface provided between the independent authenticator and the specific authenticator, in which the independent authenticator to hold master authentication keys to derive media independent pair-wise master authentication keys for handover, wherein the candidate access network and the mobile devices generate a point of attachment link address list and a mobile node link address list in a final request and response, respectively so as to securely bind the media specific keys with link layer identities, wherein each of the plurality of heterogeneous access networks is connected to a dedicated independent authenticator via the interface; and a separate interface to form a bilateral communication connection between the dedicated independent authenticator of each of the plurality of heterogeneous access networks, wherein each of the plurality of heterogeneous access networks with the dedicated independent authenticator having two separate point of services, wherein the independent authenticator being a single point of service for managing bilateral communication among each of the plurality of heterogeneous access networks via the interface, wherein the mobile devices perform authentication with the independent authenticator of a corresponding candidate access network by going through a serving access network, to which the mobile devices are attached, prior to handover of the mobile devices to the corresponding candidate access network, wherein a trigger for the handover can be generated by the candidate access network or by the mobile devices, wherein the mobile devices directly perform authentication with the independent authenticator of a corresponding candidate access network prior to handover of the mobile devices to the corresponding candidate access network, wherein a trigger for the handover can be generated by the candidate access network followed by the response from the mobile devices or by the mobile devices

followed by the response from the candidate network, wherein the serving media comprises cable, fiber, radio frequency and satellite.

[0110] The above and/or other aspects, features and/or advantages of various embodiments will be further appreciated in view of the following description in conjunction with the accompanying figures. Various embodiments can include and/or exclude different aspects, features and/or advantages where applicable. In addition, various embodiments can combine one or more aspect or feature of other embodiments where applicable. The descriptions of aspects, features and/or advantages of particular embodiments should not be construed as limiting other embodiments or the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0111] The above and other features and advantages of embodiments of the present invention will become more apparent by describing in detail exemplary embodiments thereof with reference to the attached drawings in which:

[0112] FIG. 1 depicts some illustrative architectural components that can be employed in some illustrative and non-limiting implementations including wireless access points to which client devices communicate;

[0113] FIG. 2 shows an illustrative computer or control unit that can be used to implement computerized process steps, to be carried out by devices, such as, e.g., an access point, a user station, a source node or destination node in some embodiments;

[0114] FIG. 3 is an illustrative media independent handover function communication model as disclosed in I.E.E.E. 802. 21 standard on which authentication related security signaling can be implemented in some embodiments;

[0115] FIG. 4 is an example of network model with media independent handover services communication model on which authentication related security signaling can be implemented in some embodiments;

[0116] FIG. 5 illustrates the relationship between different functional entities and their involvement during direct proactive authentication signaling;

[0117] FIG. 6 illustrates the relationship between different functional entities and their involvement during indirect proactive authentication signaling;

[0118] FIG. 7 illustrates an illustrative logical architecture for proactive authentication where two access networks are managed by one media independent authenticator and key holder;

[0119] FIG. 8 illustrates an illustrative logical architecture for proactive authentication where two access networks are managed by two separate media independent authenticator and key holders;

[0120] FIG. 9 shows the message flows between MN and Media independent authenticator for network initiated direct proactive authentication (EAP);

[0121] FIG. 10 shows the message flows between MN and Media independent authenticator for mobile initiated direct proactive authentication (EAP);

[0122] FIG. 11 shows the message flows between MN and Media independent authenticator for network initiated indirect proactive authentication (EAP);

[0123] FIG. 12 shows the message flows between MN and Media independent authenticator for mobile initiated indirect proactive authentication (EAP);

[0124] FIG. 13 shows the message flows between MN and Media independent authenticator for network initiated direct proactive authentication (ERP);

[0125] FIG. 14 shows the message flows between MN and Media independent authenticator for mobile initiated direct proactive authentication (ERP);

[0126] FIG. 15 shows the message flows between MN and Media independent authenticator for network initiated indirect proactive authentication (ERP);

[0127] FIG. 16 shows the message flows between MN and Media independent authenticator for mobile initiated indirect proactive authentication (ERP);

[0128] FIG. 17 shows the call flows between MN, target MSA-KH and target MIA-KH;

[0129] FIG. 18 shows the call flows for both network initiated and mobile initiated direct termination procedures; and

[0130] FIG. 19 shows the call flows for both network initiated and mobile initiated indirect termination procedures.

DETAILED DESCRIPTION

[0131] While the present invention may be embodied in many different forms, a number of illustrative embodiments are described herein with the understanding that the present disclosure is to be considered as providing examples of the principles of the invention and that such examples are not intended to limit the invention to preferred embodiments described herein and/or illustrated herein.

Terminologies:

- [0132] EAP: Extensible Authentication Protocol
- [0133] ERP: EAP Re-Authentication Protocol
- [0134] SA: Serving Authenticator
- [0135] CA: Candidate Authenticator

DEFINITIONS

[0136] Authentication Process: The cryptographic operations and supporting data frames that perform the authentication.

[0137] Media Specific Authenticator and Key Holder (MSA-KH): Media specific authenticator and key holder is an entity that facilitates authentication of other entities attached to the other end of a link specific to a media.

[0138] Media Independent Authenticator and Key Holder (MIA-KH): Media Independent authenticator and key holder is an entity that interacts with MSA-KH and facilitates proactive authentication of other entities attached to the other end of a link of a MSA-KH.

[0139] Proactive Authentication: An authentication process that is performed between MIA-KH and other entities attached to the other end of a link of a MSA-KH. This process occurs when the other entities intend to perform a handover to another link.

[0140] Serving MIA-KH: The MIA KH that is currently serving to a mobile node which is attached to an access network

[0141] Candidate MIA-KH: The MIA-KH that is serving to an access network which is in the mobile node's list of potential candidate access networks.

[0142] MIH Security Association (SA): An MIH SA is the security association between the peer MIH entities

REFERENCES

[0143] Each and every of the following references are incorporated herein by reference in this disclosure in their entireties for background reference:

- [0144] [RFC4748] H. Levkowitz, Ed. and et al, Extensible Authentication Protocol (EAP), RFC 3748;
- [0145] [RFC5296] V. Narayan and L. Dondeti, "EAP Extensions for EAP Re-authentication Protocol (ERP)" RFC 5296;
- [0146] [RFC4306] C. Kaufman, Ed, "Internet Key Exchange (IKEv2) Protocol," RFC 4306;
- [0147] [RFC5246] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246;
- [0148] [RFC4347] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security", RFC 4347;
- [0149] [RFC5295] J. Saloway, et al, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)" RFC 5295; and
- [0150] [IEEE802.21] IEEE P802.21 Std-2008, IEEE Standard for Local and Metropolitan Area Networks—Part 21: Media Independent Handover Services.

[0151] Proactive Authentication

[0152] Proactive authentication is a process by which an entity can perform a priori network access authentication with a media independent authenticator and key holder (MIA-KH) that is serving a candidate network. The entity performs such authentication in anticipation of handover to the neighboring networks. Proactive authentication can be performed in two ways: i) Direct Proactive Authentication (FIG. 5) whereby the authentication signaling is transparent to the serving MIA-KH and ii) Indirect Proactive Authentication (FIG. 6) whereby the serving MIA-KH is aware of the authentication signaling. In each case either EAP (Extensible Authentication Protocol) [RFC4798] or ERP (EAP Re-Authentication Protocol) [RFC5296] can be used as authentication protocol.

[0153] FIG. 5 and FIG. 6 illustrate the relationship between different functional entities and their involvement during proactive authentication signaling. For direct proactive authentication, mobile node directly communicates with the candidate MIA-KH (FIG. 5) and for indirect proactive authentication, mobile node first communicates with the serving MIA-KH. Serving MIA-KH then communicates with the candidate MIA KH on behalf of mobile node.

[0154] Proactive Authentication Architecture

[0155] The exemplary embodiments in FIG. 7 and FIG. 8 depict two example logical architectures for proactive authentication. The Media Independent Authenticator and Key Holder (MIA-KH) is the entity that facilitates the authentication prior to handover to candidate networks. In this architecture, the authentication functionalities are added within Media Independent Handover Function (MIHF) and the new entity is called as enhanced POS (e.g., PoS+).

[0156] The Media Specific Authenticator and key holder (MSA-KH) is responsible for authenticating devices for access to a specific access network and the proposed architecture assumes no change of such existing mechanisms. The difference between FIG. 7 and FIG. 8 is that in FIG. 7, two access networks are managed by one MIA-KH and hence there exists one PoS while in FIG. 8, each access network has

their own MIA-KH and hence two separate PoSes are required. FIG. 8 also has one additional interface called RP5 per MIH communication model [IEEE Std 802.21.2008].

[0157] This disclosure supports both direct and indirect proactive authentication including network-initiated and mobile-initiated procedures. The sequence of operation involves:

- [0158] MN attaches to the access network with access specific authentication procedures;
- [0159] During handover preparation stage, MN discovers the candidate authenticators;
- [0160] Depending upon the reachability of the media independent authenticator, MN performs either direct or indirect proactive authentication using RP1 interface;
- [0161] Once media independent authentication is successfully performed, the media specific keys are either pushed to or pulled from MSA-KH. Interface MIA-KH-MSA-KH is used to perform this operation;
- [0162] MN executes the handover by performing the media specific secure association (e.g., 4-way handshake for 802.11) and attaches to one of the candidate networks known as target network; and
- [0163] After connection establishment, MN reregisters with the PoS.

Following assumptions are made throughout the rest of this section:

Assumptions

- [0164] Authenticator is a MIH PoS;
- [0165] MIH protocol is used for carrying EAP and ERP;
- [0166] MIHF-ID of MN is used as the media-independent identity of the MN;
- [0167] MIHF-ID of authenticator is used as the media-independent identity of the authenticator;
- [0168] Media Independent Authenticator holds MSK (Master Session Key) or rMSK (Re-authentication MSK) generated by EAP;
- [0169] MSK or rMSK is used for deriving media-independent pair-wise master key (MI-PMK); and
- [0170] When MN hands over to the target MSA-KH and it has a media-specific PMK (MS-PMK) derived from an MI-PMK for the target MSA-KH, it runs media-specific secure association using the MS-PMK.
- [0171] Proactive Authentication Using EAP
- [0172] This section describes the procedures using EAP as proactive authentication protocol.
- [0173] Direct Proactive Authentication
- [0174] In this scenario, MN directly performs the authentication with the media independent candidate authenticator. The assumption here is that MN either knows the candidate authenticator or discovers through MIH Information Service. Candidate authenticator must be reachable directly from the MN via an IP link.
- [0175] Call Flows
- [0176] FIG. 9 describes the message flows between MN and Media independent authenticator for network initiated direct proactive authentication. It covers both example A and example B architectures. For example B architecture, Serving MIA-KH and candidate MIA-KH authenticators are two separate entities and they use interface RP5 to communicate each other. Two new MIH message types: i) MIH_Pro_Auth Request (EAP) message and ii) MIH_Pro_Auth Response message are proposed for carrying EAP messages over MIH. The first MIH_Pro_Auth request message is initiated by the

network that carries the EAP_Identity_Request message which is followed by a MIH_Pro_Auth Response message from the MN. The PoA-Link-Addr-List and MN-Link-Addr-List are necessary in the final request/response message in order for securely binding the keys with the link layer identities.

[0177] FIG. 10 describes the message flows between MN and Media independent authenticator for mobile initiated proactive authentication. The important difference from the previous one is that the trigger comes from the MN that generates the MIH_Pro_Auth Request and sends it to the candidate authenticator directly. The remaining call flows are similar to network initiated direct proactive authentication as described in FIG. 9.

[0178] Indirect Proactive Authentication

[0179] In this scenario, MN cannot perform the authentication directly with the media independent candidate authenticator. The serving authenticator takes part in forwarding the messages either to the MN (in case of network initiated authentication) or candidate MIA-KH (in case of mobile initiated authentication). The assumption here is that MN either knows the candidate authenticator or discovers through MIH Information Service but MN cannot reach to the candidate authenticator directly via an IP link.

[0180] Call Flows

[0181] FIG. 11 describes the message flows between MN and Media independent authenticators for network initiated indirect proactive authentication. As described earlier, it covers both example A and example B architectures and for example B architecture, Serving MIA-KH and candidate MIA-KH entities are two separate entities and they use interface RP5 to communicate each other. The first MIH_Pro_Auth request message is initiated by the candidate MIA-KH and sent it to serving MIA-KH which is then forwarded to the MN.

[0182] MN generates MIH_Pro_Auth Response message and subsequent EAP messages are carried over request and response messages. The PoA-Link-Addr-List and MN-Link-Addr-List are used for securely binding the key with the link layer identities.

[0183] FIG. 12 depicts the mobile initiated indirect proactive authentication in which trigger comes from the MN that generates the MIH_Pro_Auth Request message and sends it to the serving MIA-KH. Candidate MIA-KH receives this message from serving MIA-KH and sends the MIH_Pro_Auth Response message to the serving MIA-KH which is then forwarded to the MN. The remaining call flows are similar to network initiated indirect proactive authentication as described in FIG. 11.

[0184] Proactive Authentication Using ERP

[0185] This section describes the procedures using ERP as proactive authentication protocol.

[0186] Direct Proactive Authentication

[0187] In this scenario, MN directly performs the authentication with the media independent candidate authenticator. The assumption here is that MN either knows the candidate authenticator or discovers through MIH Information Service. Candidate authenticator must be reachable directly from the MN via an IP link.

[0188] Call Flows

[0189] FIG. 13 illustrates the message flows between MN and Media independent authenticator for network initiated direct proactive authentication. It covers both example A and example B architectures and for example B architecture,

Serving MIA-KH and candidate MIA-KH authenticators are two separate entities and they use interface RP5 to communicate each other. A new MIH message type called MIH_Pro_Auth Indication is proposed for initiating ERP messages exchanges over MIH. This triggers the MN to generate the MIH_Pro_Auth request (ERP) message. The PoA-Link-Addr-List and MN-Link-Addr-list are necessary in the request/response message in order for securely binding the keys with the link layer identities.

[0190] FIG. 14 illustrates the message flows between MN and Media independent authenticator for mobile initiated proactive authentication. The main difference from the previous one is that the trigger comes from the MN that generates the MIH_Pro_Auth Request and sends it to the candidate authenticator directly. Finally candidate MIA-KH sends the MIH_Pro_Auth Response with the authentication success or failure.

[0191] Indirect Proactive Authentication

[0192] In this scenario, MN cannot perform the authentication directly with the media independent candidate authenticator. The serving authenticator takes part in forwarding the messages either to the MN (in case of network initiated authentication) or candidate MIA-KH (in case of mobile initiated authentication). The assumption here is that MN either knows the candidate authenticator or discovers through MIH Information Service but MN cannot reach to the candidate authenticator directly via an IP link.

[0193] Call Flows

[0194] FIG. 15 illustrates the message flows between MN and Media independent authenticators for network initiated indirect proactive authentication. The first MIH_Pro_Auth request message is initiated by the candidate MIA-KH and sent it to serving MIA-KH which is then forwarded to the MN. MN generates MIH_Pro_Auth Response message and subsequent EAP messages are carried over request and response messages. The PoA-Link-Addr-List and MN-Link-Addr-list are used for securely binding the key with the link layer identities.

[0195] FIG. 16 illustrates the mobile initiated indirect proactive authentication in which trigger comes from the MN that generates the MIH_Pro_Auth Request message with ERP and sends it to the serving MIA-KH. Candidate MIA-KH receives this message from serving MIA-KH and sends the MIH_Pro_Auth Response message with ERP to the serving MIA-KH which is then forwarded to the MN. The PoA-Link-Addr-List and MN-Link-Addr-list are necessary in the request/response message for securely binding the keys with the link layer identities.

[0196] Attachment to Target Authenticator

[0197] After the authentication is performed and mobile node decides to execute the handover, it chooses one of the candidate networks and switch to that access network. This candidate network becomes the target network and the authenticator that serves this access network is called the target media specific authenticator and key holder (MSA-KH). Mobile node then performs the media specific secure association (SA) assuming that the target MSA has obtained the right set of keys from the target media independent authenticator and key holder (MIA-KH) for the mobile node.

[0198] Call Flows

[0199] FIG. 17 depicts the call flows between MN, target MSA-KH and target MIA-KH. Once the proactive authentication is successfully performed, MIA-KH generates per mobile node media specific keys that can either be pushed to

MSA-KH or pulled by the MSA-KH. Once the keys are available at the MSA-KH, mobile node can perform the media specific security association as soon as it switches to the network without needing to perform a full authentication. Once the secure association is successful and an IP connection is established, MN registers with the MIA-KH in order for the MIA-KH to correctly register the mobile node as its serving node.

[0200] Proactive Authentication Termination

[0201] The purpose of the proactive authentication termination is to ensure that mobile node and candidate/target/ serving authenticator terminates the session and corresponding state machines are synchronized. At this point MI-PMK and MS-PMK are either cached or deleted.

[0202] Direct Proactive Authentication Termination

[0203] Direct proactive authentication termination allows both network and mobile node to directly terminate the authentication states.

[0204] Call Flows

[0205] FIG. 18 illustrates the call flows for both network initiated and mobile initiated termination procedures. The purpose of including the integrity check is to verify the authenticity of the termination request and response.

[0206] Indirect Proactive Authentication Termination

[0207] Indirect proactive authentication termination allows both network and mobile node to terminate the authentication states via the serving MIA-KH.

[0208] Call Flows

[0209] FIG. 19 illustrates the call flows for both network initiated and mobile initiated termination procedures. The purpose of including the integrity check is to verify the authenticity of the termination request and response. In both cases, serving MIA-KH forwards the termination request either to the MN or to the candidate MIA-KH.

[0210] Primitives

[0211] This section outlines the primitives and corresponding parameters that are required to enable the proactive authentication.

[0212] Event Primitives

[0213] Table 1 describes the list of link events.

TABLE 1

List of Event Primitives		
Link Events	Link event type	Description
Link_Pro_Auth_Key_Install	Local	This event indicates that a key distribution request for a MS-PMK has been generated by a media-specific access function.
[0214] Link_Pro_Auth_Key_Install.Indication		
[0215] Function: this notification is delivered when a layer 2 connection establishment is attempted on the specified link interface Semantics of service primitive.		
Link_Pro_Auth_key_Install.indication (LinkIdentifier)		

[0216] Parameters:

Name	Type	Description
Link Identifier	LINK_TUPLE_ID	Identifier of the link associated with the event.

[0217] When Generated

[0218] This notification is generated when a layer 2 connection is established for the specified link interface or after the completion of a successful proactive authentication.

[0219] Effect on Receipt

[0220] The MIHF shall pass this link notification to the MIH User(s) that has subscribed for this notification. The MIH User(s) either push or pull the media specific keys.

[0221] MIH Events

[0222] Table 2 describes the list of MIH events

TABLE 2

List of MIH Events		
MIH Events	MIH event type	Description
MIH_Pro_Auth_Result	Local	Success or failure of proactive authentication (EAP or ERP)

[0223] MIH_Pro_Auth_Result.Indication

[0224] Function: the MIH_Pro_Auth Result is sent to local MIHF users to notify them of a local event, or is the result of the receipt of an MIH_Pro_Auth Request message to indicate to the remote MIHF users, who have subscribed to this remote event.

Semantics of Service Primitive

[0225]

```
MIH_Pro_Auth_Result.indication (
    SourceIdentifier
    MobileNodeIdentifier
    CandidateIdentifier(Optional)
    MobileLinkIdentifiers
    PoALinkIdentifiers
    Status
)
```

[0226] Parameters:

Name	Type	Description
Source Identifier	MIHF_ID	This identifies the invoker of this primitive which can be either the local MIHF or a remote MIHF.
Mobile Node Identifier	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Mobile Link Identifiers	LIST(LINK_ID)	Link identifiers of MN.

-continued

Name	Type	Description
PoA Link Identifiers	LIST(LINK_ID)	Link identifiers of PoA.
Status	STATUS	Status of (Success or failure) of proactive authentication

[0227] When Generated

[0228] This primitive is generated by the local or remote MIHF when an MIH_Pro_Auth request message is received.

[0229] Effect on Receipt

[0230] The MIHF shall pass this link notification to the MIH User(s) that has subscribed for this notification.

[0231] Command Primitives

[0232] Table 3 describes the MIH commands.

TABLE 3

List of MIH Commands		
MIH Commands	MIH Command type	Description
MIH_Pro_Auth_Start	Remote	Starting Proactive authentication
MIH_Pro_Auth_Termination	Remote	Terminating proactive authentication
MIH_Pro_Auth_key_Install	Local	Installing proactive authentication keys

MIH_Pro_Auth_Start.Request

[0233] Function: The primitive is invoked by an MIH User to indicate to a local MIHF or a peer MIH User about its intent of proactive authentication.

Semantics of Service Primitive

[0234]

```
MIH_Pro_Auth_Start.request (
    DestinationIdentifier
    MobileNodeIdentifier(Optional)
    CandidateIdentifier(Optional)
)
```

[0235] Parameters:

Name	Type	Description
Destination Identifier	MIHF_ID	This identifies a remote MIHF that will be the destination of this request.
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.

[0236] When Generated

[0237] This primitive is invoked by an MIH User to communicate with a local MIHF or remote MIH User about its intent of proactive authentication.

[0238] Effect on Receipt

[0239] Upon receipt of this primitive, the local MIHF generates and sends an MIH_Pro_Auth request message to the remote MIHF identified by the Destination Identifier. The remote MIHF forwards the request as an indication to the MIH User.

MIH_Pro_Auth_Start.Indication

[0240] Function: this primitive is used by an MIHF to indicate to an MIH User that an MIH_Pro_Auth request message was received from a remote MIHF.

Semantics of Service Primitive

[0241]

```
MIH_Pro_Auth_Start.indication (
    SourceIdentifier,
    MobileNodeIdentifier
    CandidateIdentifier
)
```

Parameters:

[0242]

Name	Type	Description
Source Identifier	MIHF_ID	This identifies the invoker of this primitive which can be either the local MIHF or a remote MIHF.
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.

[0243] When Generated

[0244] This primitive is generated by an MIHF on receiving an MIH_Pro_Auth request message from a peer MIHF.

[0245] Effect on Receipt

[0246] An MIH User receiving this indication shall invoke an MIH_Pro_Auth_Start.response primitive towards the remote MIHF indicated by the Source Identifier in the request message.

MIH_Pro_Auth_Start.Response

[0247] Function: This primitive is used by the MIHF on an MN to respond to an MIH_Pro_Auth request message from a remote MIHF in the network.

Semantics of Service Primitive

[0248]

```
MIH_Pro_Auth_Start.response (
    DestinationIdentifier,
    MobileNodeIdentifier (Optional),
    CandidateIdentifier (optional),
    Status,
)
```

Parameters:

[0249]

Name	Type	Description
Destination Identifier	MIHF_ID	This identifies a remote MIHF that will be the destination of this request.
Mobile Node Identifier	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Status	STATUS	Status of (Success or failure) of proactive authentication

[0250] When Generated

[0251] The remote MIH User invokes this primitive in response to an MIH_Pro_Auth_Start.indication from its MIHF.

[0252] Effect on Receipt

[0253] The MIHF sends an MIH_Pro_Auth response message to the peer MIHF as indicated in the Destination Identifier.

MIH_Pro_Auth_Start.Confirm

[0254] Function: This primitive is used by the MIHF to confirm that an MIH_Pro_Auth response message was received from a peer MIHF.

Semantics of Service Primitive

[0255]

```
MIH_Pro_Auth_Start.confirm (
    SourceIdentifier,
    MobileNodeIdentifier(Optional),
    CandidateIdentifier(Optional),
    Status,
)
```

Parameters:

[0256]

Name	Type	Description
Source Identifier	MIHF_ID	This identifies the invoker of this primitive which can be either the local MIHF or a remote MIHF.

-continued

Name	Type	Description
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Status	STATUS	Status of (Success or failure) of proactive authentication

[0257] When Generated**[0258]** This primitive is generated by the MIHF on receiving an MIH_Pro_Auth response message from a peer MIHF.**[0259]** Effect on Receipt**[0260]** On receiving the primitive the entity that originally initiated the proactive authentication request decides to carry out the proactive authentication or abort it based on the primitive. However, if Status does not indicate "Success or Failure", the recipient ignores any other returned values and, instead, performs appropriate error handling.

MIH_Pro_Auth_Termination.Request

[0261] Function: The primitive is invoked by an MIH User to indicate to a peer MIH User about the termination of proactive authentication.

Semantics of Service Primitive

[0262]

```

MIH_Pro_Auth_Termination.request (
    DestinationIdentifier,
    MobileNodeIdentifier(Optional)
    CandidateIdentifier(Optional)
    KeyCache
)

```

Parameters:

[0263]

Name	Type	Description
Destination Identifier	MIHF_ID	This identifies a remote MIHF that will be the destination of this request.
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Key Cache	Boolean	This indicates if a key caching is requested or not: True: key caching required; False: key caching not required;

[0264] When Generated**[0265]** This primitive is invoked by an MIH User to communicate with a remote MIH User about the termination of proactive authentication.**[0266]** Effect on Receipt**[0267]** Upon receipt of this primitive, the local MIHF generates and sends an MIH_Pro_Auth_Termination request message to the remote MIHF identified by the Destination Identifier. The remote MIHF forwards the request as an indication to the MIH User.

MIH_Pro_Auth_Termination.Indication

[0268] Function: This primitive is used by an MIHF to indicate to an MIH User that an MIH_Pro_Auth_Termination request message was received from a remote MIHF.

Semantics of Service Primitive

[0269]

```

MIH_Pro_Auth_Termination.indication (
    SourceIdentifier,
    MobileNodeIdentifier(Optional)
    CandidateIdentifier(Optional)
    KeyCache
)

```

Parameters:

[0270]

Name	Type	Description
Source Identifier	MIHF_ID	This identifies the invoker of this primitive which can be either the local MIHF or a remote MIHF.
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Key Cache	Boolean	This indicates if a key caching is requested or not: True: key caching required; False: key caching not required;

[0271] When Generated**[0272]** This primitive is generated by an MIHF on receiving an MIH_Pro_Auth_Termination request message from a peer MIHF.

Effect on Receipt

[0273] An MIH User receiving this indication shall generate a MIH_Pro_Auth_Termination.response towards the remote MIHF indicated by the Source Identifier in the request message.

MIH_Pro_Auth_Termination.Response

[0274] Function: this primitive is used by the MIHF on a MN to respond to an MIH_Pro_Auth request message from a remote MIHF in the network.

Semantics of Service Primitive

[0275]

MIH_Pro_Auth_Termination.response	(
DestinationIdentifier,	
MobileNodeIdentifier(Optional),	
CandidateIdentifier(Optional),	
Status,	
)	

Parameters:

[0276]

Name	Type	Description
Destination Identifier	MIHF_ID	This identifies a remote MIHF that will be the destination of this request.
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Status	STATUS	Status of (Success or failure) of proactive authentication

[0277] When Generated

[0278] The remote MIH User invokes this primitive in response to an MIH_Pro_Auth_Termination.indication from its MIHF.

[0279] Effect on Receipt

[0280] The MIHF sends an MIH_Pro_Auth response message to the peer MIHF as indicated in the Destination Identifier.

MIH_Pro_Auth_Termination.Confirm

[0281] Function: This primitive is used by the MIHF to confirm that an

[0282] MIH_Pro_Termination response message was received from a peer MIHF.

Semantics of Service Primitive

[0283]

MIH_Pro_Auth_Termination.confirm	(
SourceIdentifier,	
MobileNodeIdentifier(Optional),	
CandidateIdentifier(Optional),	
Status,	
)	

Parameters:

[0284]

Name	Type	Description
Source Identifier	MIHF_ID	This identifies the invoker of this primitive which can be either the local MIHF or a remote MIHF.
Mobile Node Identifier (Optional)	MIHF_ID	This identifies a mobile node MIHF that will be the destination of this request.
Candidate Identifier (Optional)	MIHF_ID	This identifies a remote MIHF that will be the future destination.
Status	STATUS	Status of (Success or failure) of proactive authentication

[0285] When Generated

[0286] This primitive is generated by the MIHF on receiving an MIH_Pro_Auth response message from a peer MIHF.

[0287] Effect on Receipt

[0288] On receiving the primitive the entity that originally initiated the proactive authentication request decides to terminate the proactive authentication or abort it based on the primitive. However, if Status does not indicate "Success or Failure", the recipient ignores any other returned values and, instead, performs appropriate error handling.

Link_Pro_Auth_Key_Install.Request

[0289] Function: The primitive is invoked by an MIH User to indicate a local MIHF to install the media specific keys to media specific authenticator.

Semantics of Service Primitive

[0290]

Link_Pro_Auth_Key_Install.request	(
DestinationLinkIdentifierwithKeys	
)	

Parameters:

[0291]

Name	Type	Description
Destination Link Identifier with keys	LIST(SEQUENCE (LINK_TUPLE_ID, MS-PMK))	Identifier of the link(s) for which keys need to be installed and the media specific keys.

[0292] When Generated

[0293] This primitive is invoked by an MIHF to indicate a local layers to install the keys to media specific authenticator.

[0294] Effect on Receipt

[0295] Upon receipt of this primitive, the lower layer generates a Link_Pro_Auth_Key_Install.confirm primitive to MIHF.

Link_Pro_Auth_Key_Install.Confirm

[0296] Function: This primitive is used by the MIHF to confirm that a Link_Pro_Auth_Key_Install.request was received.

Semantics of Service Primitive

[0297]

```

Link_Pro_Auth_Key_Install.confirm (
    Status
)

```

Parameters:

[0298]

Name	Type	Description
Status	STATUS	Status of key install request received

[0299] When Generated

[0300] This primitive is generated by the lower layer on receiving an Link_Pro_Auth_Key_Install.request from MIHF.

[0301] Effect on Receipt

[0302] On receiving the primitive the MIHF decides to keep the proactive authentication states or abort it based on the status.

[0303] Key Generation Algorithm

[0304] KDF: Key derivation function specified in RFC 5246

[0305] The default KDF (i.e., IKEv2 PRF+ with based on HMAC-SHA-256) is used unless explicitly negotiated between peer MIHFs

[0306] MI-PMK=KDF(MK, "MI-PMK")RAND_P|RAND_A|length)

[0307] Length of MI-PMK is 64 octets

[0308] MK (Master Key): MSK or rMSK

[0309] RAND_P: A random number generated by peer

[0310] RAND_A: A random number generated by authenticator

[0311] MS-PMK=KDF(MI-PMK, "MS-PMK")|MN_LINK_ID|POA_LINK_ID|length)

[0312] Length of MS-PMK depends on each media. In the case of 802.11, Length=32.

[0313] MN_LINK_ID: Link identifier of MN, encoded as LINK_ID data type

[0314] POA_LINK_ID: Link identifier of media-specific authenticator, encoded as LINK_ID data type

[0315] PRF+: PRF+key expansion specified in IKEv2 [RFC4306]

[0316] PRF+(K,S)=T1|T2|T3|T4|...

[0317] Where:

[0318] '|' means concatenation

[0319] T1=PRF (K, S|0x01)

[0320] T2=PRF (K, T1 (S|0x02)

[0321] T3=PRF (K, T2 (S|0x03)

[0322] T4=PRF (K, T3 (S|0x04)

[0323] ...

[0324] continuing as needed to compute the required length of key material. The default PRF is taken as

HMAC-SHA-256 [SHA256]. Since PRF+ is only defined for 255 iterations it may produce up to 8160 octets of key material.

Key Distribution Mechanism

[0325] In some embodiments, a key distribution mechanism can be established by those in the art. For example, this can require a standard or can be left as deployment specific or implementation choice.

[0326] While illustrative embodiments of the invention are set forth and described herein, the present invention is not limited to the various preferred embodiments described herein, but includes any and all embodiments having equivalent elements, modifications, omissions, combinations (e.g., of aspects across various embodiments), adaptations and/or alterations as would be appreciated by those in the art based on the present disclosure. The limitations in the claims (e.g., including that to be later added) are to be interpreted broadly based on the language employed in the claims and not limited to examples described in the present specification or during the prosecution of the application, which examples are to be construed as non-exclusive. For example, in the present disclosure, the term "preferably" is nonexclusive and means "preferably, but not limited to." In this disclosure and during the prosecution of this application, means-plus-function or step-plus-function limitations will only be employed where for a specific claim limitation all of the following conditions are present in that limitation: a) "means for" or "step for" is expressly recited; b) a corresponding function is expressly recited; and c) structure, material or acts that support that structure are not recited. In this disclosure and during the prosecution of this application, the terminology "present invention" or "invention" may be used as a reference to one or more aspect within the present disclosure. The language present invention or invention should not be improperly interpreted as an identification of criticality, should not be improperly interpreted as applying across all aspects or embodiments (i.e., it should be understood that the present invention has a number of aspects and embodiments), and should not be improperly interpreted as limiting the scope of the application or claims. In this disclosure and during the prosecution of this application, the terminology "embodiment" can be used to describe any aspect, feature, process or step, any combination thereof, and/or any portion thereof, etc. In some examples, various embodiments may include overlapping features. In this disclosure, the following abbreviated terminology may be employed: "e.g." which means "for example,"

What is claimed is:

1. An apparatus having computer implemented media independent access functions, comprising:

an independent authenticator, said independent authenticator to authenticate mobile devices attached between heterogeneous media;

an interface to connect said independent authenticator to a plurality of heterogeneous access networks each of which supports a specific authenticator that corresponds to a specific serving media; and

predefined media independent handover protocols and media independent handover identities based on the media independent functions on the computer,

wherein said independent authenticator authenticates candidate access networks prior to the handover of the mobile devices from serving access networks to the can-

- candidate access networks each of which belonging to the plurality of heterogeneous access networks having the specific serving media.
2. The apparatus as set forth in claim 1, wherein: media specific keys are pulled or pushed from the specific authenticator of a serving access network and a candidate access network via a specific interface provided between the independent authenticator and the specific authenticator, in which said independent authenticator to hold master authentication keys to derive media independent pair-wise master authentication keys for handover.
 3. The apparatus as set forth in claim 2, wherein: the candidate access network and the mobile devices generate a point of attachment link address list and a mobile node link address list in a final request and response, respectively so as to securely bind the media specific keys with link layer identities.
 4. The apparatus as set forth in claim 1, wherein: each of the plurality of heterogeneous access networks is connected to a dedicated independent authenticator via the interface; and a separate interface to form a bilateral communication connection between the dedicated independent authenticator of each of the plurality of heterogeneous access networks, wherein each of the plurality of heterogeneous access networks with the dedicated independent authenticator having two separate point of services.
 5. The apparatus as set forth in claim 1, wherein: said independent authenticator being a single point of service for managing bilateral communication among each of the plurality of heterogeneous access networks via the interface.
 6. The apparatus as set forth in claim 1, wherein: the mobile devices perform authentication with the independent authenticator of a corresponding candidate access network by going through a serving access network, to which the mobile devices are attached, prior to handover of the mobile devices to the corresponding candidate access network.
 7. The apparatus as set forth in claim 4, wherein: a trigger for the handover can be generated by the candidate access network or by the mobile devices.
 8. The apparatus as set forth in claim 1, wherein: the mobile devices directly perform authentication with the independent authenticator of a corresponding candidate access network prior to handover of the mobile devices to the corresponding candidate access network.
 9. The apparatus as set forth in claim 8, wherein: a trigger for the handover can be generated by the candidate access network followed by the response from the mobile devices or by the mobile devices followed by the response from the candidate network.
 10. The apparatus as set forth in claim 1, wherein: the serving media comprises cable, fiber, radio frequency and satellite.
 11. An article of manufacture, comprising: a machine-accessible medium having instructions encoded thereon for enabling a processor to execute an operation for proactively authenticating mobile devices attached between heterogeneous media via media independent authenticator, comprising the steps of:
 - assigning media specific authentication procedures to mobile devices attached to a serving access network within a plurality of heterogeneous access networks; providing information regarding candidate access networks by way of the media independent authenticator to mobile devices during a handover preparation stage; performing a proactive authentication using a interface; and executing, upon successfully performing said proactive authentication by way of the media independent authenticator, the handover to a candidate access network having the media specific authentication procedure.
 12. An article of manufacture as set forth in claim 11, wherein:
 - pulling or pushing media specific keys from the serving access network and the candidate access network via a specific interface provided between the media independent authenticator and media specific authenticator, in which said media independent authenticator holding master authentication keys for deriving media independent pair-wise master authentication keys for handover.
 13. An article of manufacture as set forth in claim 12, wherein:
 - the mobile devices directly performing the proactive authentication with the media independent authenticator of the candidate access network prior to handover of the mobile devices to the candidate access network.
 14. An article of manufacture as set forth in claim 13, wherein:
 - generating a trigger for the handover by the candidate access network is followed by the response from the mobile devices.
 15. An article of manufacture as set forth in claim 14, wherein:
 - the candidate access network and the mobile devices generate a point of attachment link address list and a mobile node link address list in a final request and response respectively so as to securely bind the media specific keys with link layer identities.
 16. An article of manufacture as set forth in claim 13, wherein:
 - generating a trigger for the handover by the mobile devices is followed by the response from the candidate access network.
 17. An article of manufacture as set forth in claim 16, wherein:
 - the candidate access network and the mobile devices generate a point of attachment link address list and a mobile node link address list in a final request and response respectively so as to securely bind the media specific keys with link layer identities.
 18. An article of manufacture as set forth in claim 11, wherein:
 - the mobile devices performing the proactive authentication with the media independent authenticator of the candidate access network by going through the serving access network, to which the mobile devices are attached, prior to handover of the mobile devices to the candidate access network.
 19. An article of manufacture as set forth in claim 18, wherein:
 - pulling or pushing media specific keys from the serving access network and the candidate access network via a

specific interface provided between the media independent authenticator and media specific authenticator.

generating a trigger for the handover by the candidate access network is followed by the response from the mobile devices, the candidate access network and the mobile devices generate a point of attachment link address list and a mobile node link address list in a final request and response respectively so as to securely bind the media specific keys with link layer identities; and

generating a trigger for the handover by the mobile devices is followed by the response from the candidate access network, the candidate access network and the mobile devices generate a point of attachment link address list and a mobile node link address list in a final request and response respectively so as to securely bind the media specific keys with link layer identities.

20. A system for proactively authenticating, comprising:

a server having media independent access functions including media independent authentication functions

that authenticates other entities attached via an interface to an end of a link specific to a media;

a plurality of heterogeneous networks each having media specific access functions including authentication functions corresponding to the other entities attached via the interface to the end of the link specific to the media;

mobile devices connected to said plurality of heterogeneous networks, and

said server having predefined media independent handover protocols and media independent handover identities based on the media independent functions related to handover,

wherein said server authenticates candidate access networks prior to the handover of the mobile devices from serving access networks to the candidate access networks each of which belonging to the plurality of heterogeneous access networks having the link specific to the media.

* * * * *

EXHIBIT E

```

leader 02756nam a22006015i 4500
001 99340000000024081
005 20230412075208.0
006 m o d |
007 cr nn 008mamaa
008 111117s2011 gw | o |||| 0|eng d
020 ##$a3-642-25283-4
024 7#$a10.1007/978-3-642-25283-9 $2doi
035 ##$a(CKB)3400000000024081
035 ##$a(SSID)ssj0000611067
035 ##$a(PQKBManifestationID)11397708
035 ##$a(PQKBTitleCode)TC0000611067
035 ##$a(PQKBWorkID)10644948
035 ##$a(PQKB)11100980
035 ##$a(DE-He213)978-3-642-25283-9
035 ##$a(MiAaPQ)EBC6303431
035 ##$a(MiAaPQ)EBC5592713
035 ##$a(Au-PeEL)EBL5592713
035 ##$a(OCOLC)765365879
035 ##$a(PPN)157509265
040 ##$aMiAaPQ $beng $erda $epn $cMiAaPQ $dMiAaPQ
041 ##$aeng
050 #4$aHF5734.7 $b.C446 2011
072 #7$aUKN $2bicssc
072 #7$aCOM075000 $2bisacsh
082 0#$a004 $223
245 10$aTrusted Systems : $bSecond International Conference, INTRUST 2010, Beijing, China, December 13-15, 2010, Revised Selected Papers / $cedit
250 ##$a1st ed. 2011.
264 #1$aBerlin, Heidelberg : $bSpringer Berlin Heidelberg : $bImprint: Springer, $c2011.
300 ##$a1 online resource (XIV, 362 p. 73 illus.)
336 ##$atext $btxt
337 ##$acomputer $bc
338 ##$aonline resource $bcr
490 1#$aSecurity and Cryptology ; $v6802
546 ##$aEnglish
500 ##$aIncludes index.
505 0#$aIntro -- Title -- Preface -- Table of Contents -- Implementation Technology -- Seamless Integration of Trusted Computing into Standard C
505 8#$aPlatform Instantiation and Property Verification -- Initial Memory Contents -- Component Location Configuration -- Application Logic -- P
505 8#$aIntroduction -- Related Works -- Trusted Subjects Configuration -- Notation for Trusted Subject -- Information Flow Graph -- MLS Policy I
505 8#$aComparisons of VLR Methods -- Comparisons between Rekey and VLR Methods -- Using Rekey and VLR Approaches Together -- Conclusions -- Refe
505 8#$aHost Computer Smart Card Middlewares and Smart Card Filesystem -- Smart Card Provisioning Server -- Credential Life-Cycle -- Implementati
505 8#$aInformation Leakage Attacks on Robots.
520 ##$aThis book constitutes the thoroughly refereed post-conference proceedings of the International Conference on Trusted Systems, INTRUST 201
650 #0$aComputer networks.
650 #0$aCryptography.
650 #0$aData encryption (Computer science).
650 #0$aElectronic data processing--Management.
650 #0$aAlgorithms.
650 #0$aComputers and civilization.
650 #0$aData protection.
650 14$aComputer Communication Networks.
650 24$aCryptography.
650 24$aIT Operations.
650 24$aAlgorithms.
650 24$aComputers and Society.
650 24$aData and Information Security.
776 ##$z3-642-25282-6
760 0#$tLecture Notes in Computer Science $w(CKB)954928560451 $w(DLC)sf92020465 $x1611-3349
700 1#$aChen, Liqun. $eeditor. $4edt $4http://id.loc.gov/vocabulary/relators/edt
700 1#$aYung, Moti. $eeditor. $4edt $4http://id.loc.gov/vocabulary/relators/edt
830 #0$aSecurity and Cryptology ; $v6802
906 ##$aBOOK

```

This service is provided by [GW Libraries](#) in partnership with the [Washington Research Library Consortium](#)

EXHIBIT F

Uni-directional Trusted Path: Transaction Confirmation on Just One Device

Atanas Filyanov*, Jonathan M. McCune†, Ahmad-Reza Sadeghi‡, Marcel Winandy*

* *Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany*

{*atanas.filyanov, marcel.winandy*}@rub.de

† *CyLab, Carnegie Mellon University, USA*

jonmccune@cmu.edu

‡ *Center for Advanced Security Research Darmstadt / Technical University Darmstadt, Germany*

ahmad.sadeghi@cased.de

Abstract—Commodity computer systems today do not include a full trusted path capability. Consequently, malware can control the user’s input and output in order to reveal sensitive information to malicious parties or to generate manipulated transaction requests to service providers. Recent hardware offers compelling features for remote attestation and isolated code execution, however, these mechanisms are not widely used in deployed systems to date. We show how to leverage these mechanisms to establish a “one-way” trusted path allowing service providers to gain assurance that users’ transactions were indeed submitted by a human operating the computer, instead of by malware such as transaction generators. We design, implement, and evaluate our solution, and argue that it is practical and offers immediate value in e-commerce, as a replacement for captchas, and in other Internet scenarios.

Keywords—security; transaction confirmation; trusted path; trusted computing;

I. INTRODUCTION

For decades a significant challenge in computer security has been realizing a mechanism for establishing a full *trusted path*, i.e., a mechanism that (i) isolates the input and output channels of different applications to preserve the integrity and confidentiality of data exchanged with the user, (ii) provides a technical means that assures the user of a computer system that she is truly interacting with the intended software, and (iii) assures running applications that user inputs truly originate from the actions of a human (as opposed to being synthesized or injected by other software).

Software vulnerabilities in the increasingly complex software stack on commodity computing devices offer a large attack surface that attackers can exploit to inject malicious Trojan horse software, such as keyloggers [1] or transaction generators [2]. These software-based attacks can eavesdrop on user input or modify transactions in a malicious way, e.g., waiting until the user has legitimately authenticated to the web server, and then issuing illegitimate transactions using this channel (e.g., by faking or scripting user input).

Some trusted path proposals suggest the use of an external *verifier* device. However, the increasing use of mobile devices for sensitive online transactions and e-commerce

applications make these solutions inconvenient, especially given today’s trend in device convergence (e.g., smartphones including cameras, web access, GPS, etc.). Using one’s mobile phone to verify one’s laptop may be reasonable in some scenarios, but there are no verifiers available if the transaction itself is taking place on the mobile device while on-the-go. Thus, the challenge remains to somehow provide a trustworthy mechanism on *just one device* that can defend against transaction generators and malicious scripting that imitates actions intended to be performed only by humans.

An additional disincentive for existing solutions is that end users are rarely exposed to the liability for fraud. In the US and much of Europe, credit card companies, banks, and online merchants absorb this liability. Thus, it is these institutions that have the incentive to make online transactions more resilient to attacks.

We design, implement, and evaluate a secure transaction confirmation architecture that can provide assurance to a remote server that the user of a client system has indeed confirmed a proposed action. With this property, a service provider can ensure that it only commits transactions on the user’s behalf if the user has actually confirmed the transaction. We are able to realize such a system by providing the properties of a trusted path *in one direction only*.

Restricting ourselves to a uni-directional trusted path enables us to realize a significantly more practical implementation, remaining compatible with legacy operating systems and applications. Our goal is to provide remotely-verifiable evidence to service providers of events taking place on the physical user-centric I/O devices. That is, *we wish to prove to a remote server that an actual user typed something or saw something displayed on-screen*. This makes it possible to achieve trusted transaction confirmation on *just one device*. Online service providers using our system are able to detect and reject transaction generators. In contrast to a full trusted path, we do not provide local, user-verifiable evidence of the genuineness of output, i.e., malicious code may still fake the confirmation process from the perspective of the user.

When implemented properly, the client-side user experi-

ence need not deviate from what users expect today. This same technology also enables a more usable captcha [3] mechanism, since our framework enables client platforms to generate evidence of events originating on user-controlled peripherals. We consider application scenarios where human users can perform actions (e.g., enroll for services or issue transactions) on Internet services that require a verification of the user's intent. E-commerce web sites, online banking, e-government services, and even e-voting are all examples.

Based on a uni-directional trusted path, we architect a trusted confirmation and optionally authentication (§III-D) agent that executes in CPU-provided isolation. The trusted computing base (TCB) of the trusted agent is relatively small compared to prior works (a few thousand lines of code), and remains compatible with users' existing operating system and application environments. We evaluate the performance and security of our implementation, and provide an estimation of the efforts needed to port our solution to other hardware architectures, such as smartphones.

II. PROBLEM DEFINITION

A. Background: Trusted Path

Several approaches for realizing a full trusted path have been proposed over the years, though none enjoy widespread adoption on commodity systems. We briefly review some representative designs, and their drawbacks.

One approach is to use a full trusted OS and window manager, where a dedicated area of the screen is reserved for the exclusive use of a trusted software component that shows the identity and status of the current application [4]. While this concept has been implemented in some research systems [5], [6], the goal of widespread adoption has remained elusive.

A closely related variant leverages the notion of a secure attention sequence, e.g., "Press Control-Alt-Delete to log on." The assumption here is that the OS kernel remains uncompromised, and will always be the first software layer to process keyboard input. Thus, any spoofed login dialog box will be immediately overwritten by the legitimate box. However, users must be taught to always press the necessary key sequence. Another approach for indicating a trusted state of a computer system is based on its ability to display a "secret picture" [7] (or any other human-recognizable secret). The system is architected such that the image data can only be decrypted if the system is running approved software, and the user must diligently remember to look for her designated image whenever she performs a security-sensitive operation. A final approach is to use some form of dedicated additional hardware as an axiomatically trustworthy indicator, in the limit something as simple as a dual-color LED [8]. This design is compelling as it still enables full screen applications, which must otherwise be disallowed given their ability to spoof other security indicators.

Unfortunately, none of these designs enjoy widespread deployment for online transactions. While additional hardware

costs are sometimes a factor, we believe a significant barrier to adoption on commodity systems is a lack of interest from users. Vendors and financial institutions absorb the majority of the risk in fraudulent online transactions today. A confirmation system can be constructed based on a uni-directional trusted path that does not strictly require any additional work from the user, beyond reading a transaction summary before finalizing a transaction (an action that is already a standard part of online purchases).

B. Adversary Model

The model we consider involves four parties: (1) the user, (2) the user's computing device, (3) the service provider, and (4) an attacker. The attacker has complete control over the network between the service provider and the user's device. Thus, the attacker may try to impersonate the service provider to trick the user. Moreover, the attacker is able to install malware on the user's computing platform or modify any existing software there. The goal of the attacker is to issue transactions to the service provider illegitimately on behalf of the user.

C. Assumptions

While the adversary has control over the user's software environment, we make the following exception. We assume the device has some form of secure execution environment that is protected against software-based attacks, and that the hardware is correctly implemented and protects the integrity of the secure environment. Note that AMD SVM [9] and Intel TXT [10] are examples of widely deployed hardware capabilities that can provide the necessary properties. Moreover, we assume that the attacker cannot gain physical access to the user's platform. We assume that the service provider is honest, and that the service provider's servers are secure. We do not consider denial-of-service attacks.

D. Security Objectives

Our main security requirements are the following:

1) *Mutual authenticity of the user and service provider:* The user must somehow authenticate herself to the service provider, and the user's device must somehow authenticate the service provider on behalf of the user.

2) *Integrity of the transaction:* Authenticated users can engage in transactions with the service provider. Transaction requests must be integrity-protected both during final confirmation on the user's computing device and during transmission to the service provider.

3) *Intention of the human user:* The service provider must gain assurance that a requested transaction is indeed submitted by a human user, and not by malware. For this, the service provider must be able to reliably verify that a certain action (e.g., receive keypress events from a keyboard) has been performed in a secure execution environment.

4) *Binding between user and machine*: The service provider must be able to verify the binding between a confirmation and a particular transaction request from a certain machine.

The last requirement prevents proxy attacks, where confirmations are re-directed to other machines to get other users perform the required action. However, where privacy is a concern, the *real* identity of the user's hardware device should be appropriately anonymized, e.g., via a trusted third party or cryptographic protocols [11].

These requirements are typical for e-commerce scenarios. We prioritize application scenarios' integrity requirements over their secrecy requirements (excluding authentication). For example, in e-commerce or online banking applications, a Trojan horse plug-in that resides in the web browser may eavesdrop on account balances or what items have been recently ordered. However, the malware is prevented from generating or modifying transactions.

III. ARCHITECTURE AND DESIGN

In this section we describe the architecture and design of our uni-directional trusted path (UTP) solution. Our primary design goal is to provide an integrity-protected confirmation facility even when client users possess only a single device without full trusted path capabilities.

A. High-Level Design of UTP

The main property of UTP is to enable a remote server to gain a high level of assurance that a certain action submitted by a client system was initiated at the behest of its physically-present human user, and not by malware such as a transaction generator. This means UTP realizes properties (i) and (iii) of a trusted path (c.f. §I), but not necessarily property (ii). Hence, users might not be able to verify that they have been interacting with the intended application or web site. The underlying motivation for such an architecture is the simple fact that most users do not actually pay attention to security indicators [12], [13], [14].

If anything goes wrong, e.g., malware tries to modify a requested transaction or generate a new one, the remote server in our design will notice this and discard such malicious transactions. This is the essence of the value provided by UTP. The trusted path goes (verifiably) from the user to the server, and enables the server to significantly reduce its exposure to transaction generators and other large-scale, automated malicious client-side behavior.

UTP must enable a remote server to gain assurance that the client platform is running a software configuration that can take total control of the user-centric I/O devices, including the keyboard, display, and optionally the mouse. The client system must have the capability (via hardware or software mechanisms) to construct a trustworthy *channel* between these platform components, and to generate some form of remotely-verifiable evidence that it is indeed in

control of these devices. We explicitly do not require local, user-verifiable evidence of the presence of these channels, as this implies the full trusted path capability.¹

One viable approach for providing such a capability is an isolated execution environment with remote attestation capabilities, such as Flicker [15]. Flicker enables security-sensitive code to execute in hardware-enforced isolation from all other code and devices on the system, and to generate attestations enabling remote verification that the execution environment is established as intended. The security-sensitive code can be constructed to take control of the user-centric I/O devices (i.e., keyboard and display), and the remote verifier can ascertain, given the code and assumed correct hardware, that user-centric I/O is working as intended.

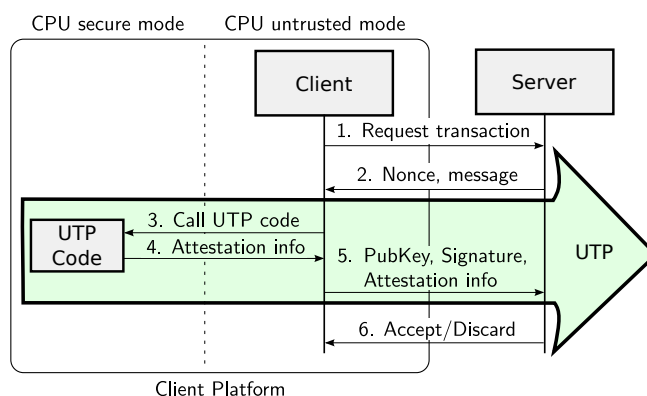


Figure 1. General architecture of UTP.

Figure 1 shows the high-level design of UTP. When the client requests an action from the server that requires a confirmation of the user's intent, the server establishes the uni-directional trusted path by sending a message to the client (and a random nonce for the purpose of preventing replay attacks). The (untrusted) client program invokes the execution of the UTP code in the CPU's secure execution mode. This mode ensures that the UTP code executes isolated from other software and successfully takes control of the user-centric I/O devices. The UTP program displays the message provided by the server (e.g., a transaction summary) to the user. Once the user has viewed the message and acted as required (e.g., confirmed her intention to submit the transaction), UTP assembles the necessary data to generate an attestation that these events transpired while in the secure execution mode. This information is cryptographically signed by a keypair that is accessible only while the isolated

¹Note that our UTP architecture can readily support the "secret picture" form of full trusted path [7]. However, we are highly skeptical of users' motivation to diligently remember to check for their picture, and to actually stop their primary task if it is missing [12], [13], [14]. Our emphasis in this paper is on the utility of a mechanism for which users need not dedicate any additional time or attention, as compared to existing systems.

execution environment is active, in order to demonstrate its authenticity and integrity. The data and its signature are then sent to the server. The server can verify the signature with the provided and (e.g., from a trusted third party) certified public key of the client platform, and subsequently verifies the attestation information in order to get assurance about the execution of UTP. Note that UTP is more than just TCG-style attestation [16] (see also the Appendix): If the verification succeeds, the server knows that a uni-directional trusted path to the human user has been established and that the transaction can be processed.

Advantages: The realization of UTP as a small trusted program using a secure execution mode of the CPU has some advantages over solutions based on a secure OS:

- *Smaller trusted code:* The overall trusted computing base (TCB) is even smaller than a secure user interface system on top of a secure operating system kernel (e.g., [5], [6]). A lot of functionality can be “out-sourced” to the untrusted environment, including the initialization of certain hardware resources. Moreover, in contrast to a security kernel or hypervisor, our design requires no persistent code when no sensitive transactions are in progress.
- *Higher adoption rate:* The chances of adoption are much higher since our system can be readily deployed with off-the-shelf operating systems that are used today. Although modern virtualization technology offers a compelling way to run a secure hypervisor beneath commodity operating systems (e.g., [17], [18], [19], [20]), monopolization of available hardware virtualization support for security architectures has been discouraged.

B. Transaction Confirmation with UTP

We describe the interactions and user experience of transaction confirmation with UTP. The basic procedure begins after the user has prepared some kind of transaction (e.g., by populating a shopping cart at a web-based retailer) and is ready to commit (e.g., make a purchase). At this point, a well-behaved client system will invoke UTP to allow the user to confirm that transaction, though a malicious client system may attempt to fool the user.

Once invoked, the UTP code takes control of the keyboard and display and shows a summary of the transaction that is about to be committed. During this time the normal (untrusted) OS is suspended, and the UTP code executes in the secure execution mode of the CPU (c.f. step 3 in Figure 1). Displaying the transaction summary and requesting an active confirmation from the user is an essential step that reveals the human user’s intentions. Otherwise, the user may be tricked into confirming anything. Note that the practical details of how to summarize the transaction are likely best implemented on the server side, either per-web server or as a standard mechanism that is adopted across several web-based service providers. It may also be an option to implement client-side logic on a per-provider basis, if the

user has a finite set of service providers with which she engages in sensitive transactions. One can even imagine the content of the confirmation page being rendered somewhere in the cloud, with a static image being transmitted to the UTP environment on the client system.²

Next, the user is prompted to perform the actual confirmation. If no attack is taking place, then the user is indeed looking at a legitimate summary of the proposed transaction, and the UTP code will be the only code that can receive the user’s forthcoming input. This confirmation is the user’s opportunity to realize that an attacker has tampered with her session. The server wants to be sure that the user actually verified the summary. From a technical perspective, the actual confirmation act may be as simple as pressing “enter” to generate a physical keyboard interrupt, but we also wish to avoid habituation by users. Interesting designs may include manually entering the final invoice amount; entering a short, random confirmation code; or typing the word YES to continue. The precise design is outside the scope of this paper, but it should be selected in response to an evidence-based user study to determine which is least intrusive while providing the desired security properties. We discuss these issues again in our security analysis (§V-A).

Once the user has viewed the transaction summary and confirmed her intention to commit the transaction, UTP assembles the necessary data to generate the attestation information and terminates its execution. The CPU exits the isolated environment and resumes the normal OS. The client (e.g., the web browser) sends the attestation information via a secure channel to the remote server. The server accepts the pending transaction if it is able to verify the attestation. This indicates that the server is convinced that the legitimate and expected software ran in an isolated environment on a device associated with the user’s account, and that the user successfully confirmed her desire to commit the transaction.

Our design leverages an isolated execution environment. Some implementations of such an environment necessarily halt the execution of other code running on the system. For example, the user’s music will stop playing when the confirmation dialog is displayed. An important question to resolve is whether this disruption is good or bad. A “pro” is that this disruption helps to get the user’s attention and focus them on the sensitive task. A “con” is that the users may find this requirement annoying, and vendors may object on the grounds that it will reduce the rate of impulse purchases.

C. CAPTCHA with UTP

CAPTCHA is defined as a Completely Automated Public Turing test to tell Computers and Humans Apart. Captchas are commonly manifested online as distorted images, and users are asked to type in the characters contained in the

²This type of pre-rendered design has been reviewed favorably, e.g., for electronic voting systems [21].

images. Unfortunately, captchas are often difficult even for humans to solve reliably [3]. Moreover, without a secure execution environment, captchas can readily be inlined into other types of free content. For example, it has been shown that there is no shortage of users willing to solve captchas for otherwise unfettered access to adult content online [22].

Our UTP solution is capable of supporting captchas that are simultaneously more usable and harder for attackers to inline. In essence, a UTP-based captcha amounts to asking the user to verify some transaction detail, e.g., to enter the final amount of a pending purchase.

D. Mutual Authentication Overview

As we have shown, UTP can be used to give a remote server assurance that a human user has actively entered a confirmation to a requested action. One of our motivations is to protect against malicious transactions resulting from malicious transaction generators. In scenarios where the transaction is associated with a user account (e.g., online purchases), malware could steal the user's account credentials and send them to the attacker. Given that the user cannot ascertain whether her system is truly in a trustworthy state, we cannot depend upon the user to employ appropriate discretion when deciding whether to enter sensitive data such as passwords or account credentials. Thus, we include in our architecture that the server is able to authenticate the client user without solely depending on user-memorized secrets (e.g., passwords). However, authentication is orthogonal to our work, and we therefore refer to existing approaches that can be easily incorporated into our design.

One option for mutual authentication is to use public key-based credentials. For the purposes of discussion, we consider strong device authentication based on hardware-protected keys, e.g., asymmetric non-migratable keys residing in a Trusted Platform Module (TPM) [16]. Viable alternatives may include password-based authentication, where the password is actually a high-entropy secret managed by and accessible only to trusted code. The PAKE protocol for password-assisted key exchange may also be used [23], [24], [25]. Challenge-response protocols, which verify knowledge of a shared secret between client and server, are another option. Finally, wallet-like authentication agents [26], [27], [28], [29], [30] automatically manage mutual authentication between the user's computer and a remote server. The authentication agent (wallet) is generally executed in an environment that isolates it from the rest of the software stack. Hence, an authentication agent could be executed in the secure execution mode of the CPU in the same way as the UTP code (see also §VI).

E. Enrollment and Setup

When UTP is used in the context of confirming transactions demanding (mutual) authentication, the credentials for the authentication (e.g., password, shared secret, or public

key-based credentials) must be established in a setup phase. The primary obstacle is enrolling the public component of a hardware-protected keypair from the client system with the service provider for subsequent use in client device authentication and attestation. This challenge has two parts. First, the hardware-protected key generation and storage infrastructure employed must be capable of certifying that a particular keypair was generated internally and will remain protected. Second, this certified key must be somehow bound to the identity of the device's user, in order to support an authentication procedure that remains secure even if the user enters her password into a malicious application.

Certifying Hardware-Protected Keys: Although technically well-understood, certifying public key-based credentials is challenging because a global public key infrastructure that can scale to include all client devices does not exist. Thus, in practice, it is likely that some form of per-service-provider bootstrapping mechanism will be required. It may be reasonable for organizations to partner such that the bootstrapping mechanism can be out-sourced to a third-party provider (with parallels to single sign-on solutions, e.g., SAML [31] or OpenID [32]). In cases where privacy is required (e.g., unlinkability of online purchases to the real identity of hardware devices), cryptographic protocols like Direct Anonymous Attestation (DAA) [33] may be used, where a signature proves the membership to a certain group, but does not reveal the identity of that particular member.

Binding Keypairs to Users: To enable the service provider to associate a certified keypair with a user, users must provide certain information about themselves (e.g., name, e-mail address, etc.). The initial enrollment can be performed on a trust-on-first-use basis in many use cases.³

To bind a certified keypair to a particular user, we can use our protected input mechanism to demonstrate to the service provider that a human entered the relevant information. For instance, the UTP-based captcha approach described above can be used. This solution demonstrates that the platform from which the human's information originates is able to perform operations using the private key corresponding to the certified public key.

IV. IMPLEMENTATION

We have implemented an end-to-end system for transaction confirmation using a uni-directional trusted path. Our setup consists of a modified open-source online merchant software package on the server side, and a browser plugin and confirmation agent on the client side. For the client side, we have implemented the UTP confirmation agent based on the Flicker [15] framework, leveraging hardware support for dynamic root of trust using Intel TXT CPU

³Of course, one person could pose as another if the impostor knows sufficient information about the individual being impersonated and if the target individual has never been enrolled before. Note that this risk already exists today, and solutions that address it are outside the scope of this paper.

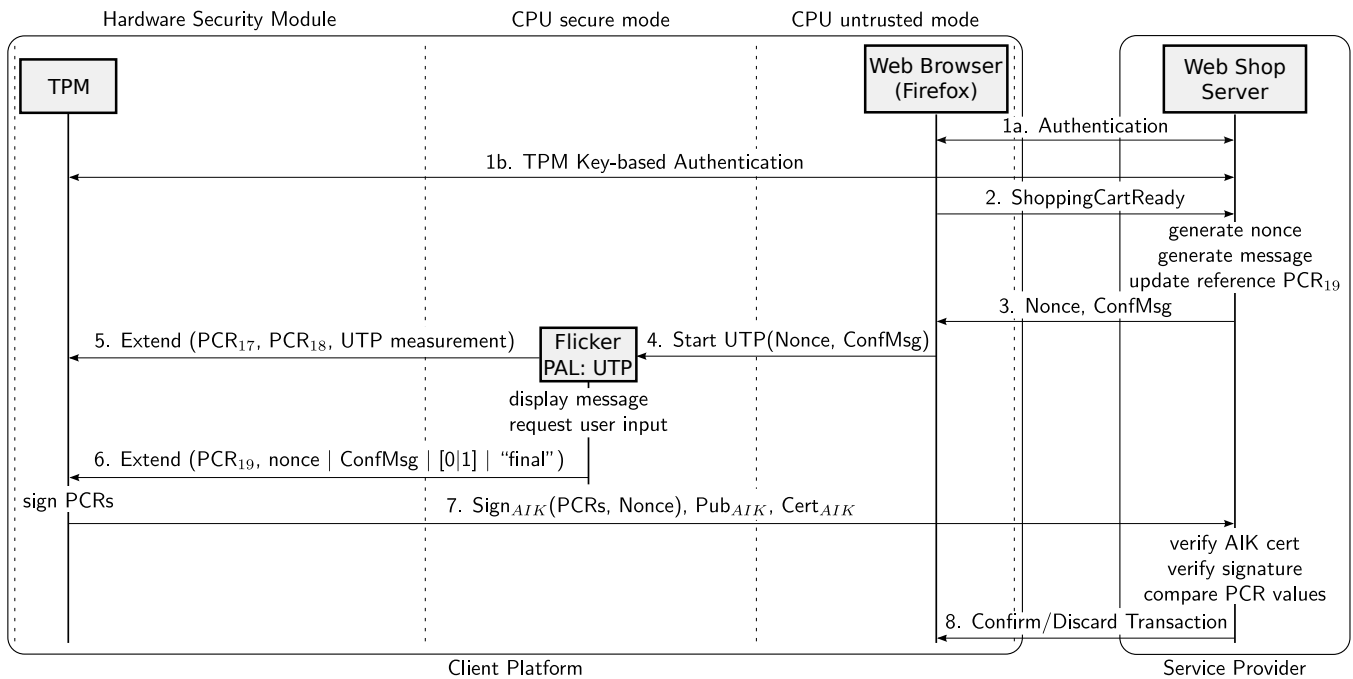


Figure 2. Prototype implementation of the UTP transaction confirmation.

extensions [10] and a v1.2 TPM [16]. Figure 2 illustrates our implementation, which we describe below. Appendix A gives additional background.

A. Authentication and Enrollment

User and Device Authentication: We implement two-factor client authentication by authenticating the client device as well as the client user. Thus, part of the user’s enrolling with our web shop includes generating and registering a TPM-based Attestation Identity Keypair (AIK) to represent the user’s client system to that service provider.⁴ Hardware support for protecting cryptographic keys, such as the TPM chip [16], has reached a level of market penetration where it is a legitimate option on which to build real-world systems. Hence, device authentication, when implemented leveraging the TPM, provides excellent protection against a compromised OS being able to masquerade as a different physical system.

Enrollment Phase: We use an AIK to represent the identity of the client’s TPM, and thus platform. In our implementation, we use a privacy CA to certify our AIK as having been generated in a real TPM. Specifically for our prototype we use privacyca.com, which provides an interface to obtain a valid AIK certificate that can be used by the TPM, given that the TPM manufacturer included an Endorsement Key Credential (Figure 3). Instead of including

⁴A unique AIK can easily be used per-merchant, alleviating any risk of privacy invasion by correlating transactions between different merchants.

a trusted third party as privacy CA, one might achieve unlinkability, as mentioned before, by using DAA [33], [34].

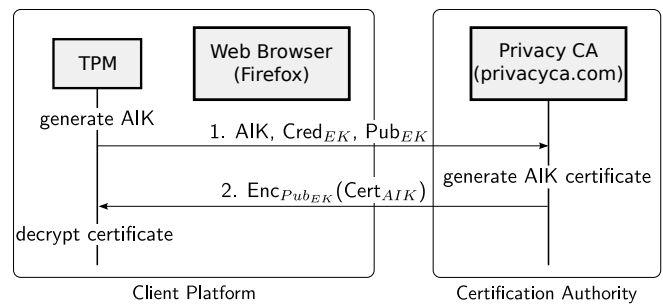


Figure 3. Device enrollment using a Privacy CA.

B. UTP Sessions

Web server: We made minor changes to the open-source FreeWebshop.org [35] to support UTP-based transaction confirmation. The server has a typical shopping cart interface where users can add items that they wish to buy, before proceeding to a check-out process that collects payment and shipping information. We modified the summary page of the ordering process in FreeWebshop.org to appear without automatically finalizing the order, and to include one additional button, which invokes UTP-based verification. When the user clicks this button, a server-side script that we have implemented to manage the UTP-based verification process is invoked. The script assembles the final verification

message to be confirmed by the user and sends it to the client. This message consists of a simplified invoice (short name for each product in the cart, quantities, itemized cost, and the total cost) and a nonce to ensure the freshness of the expected attestation covering the UTP-based verification on the client (Figure 4). The script is responsible for the exchange of the actual attestation messages and all data is tunneled through the existing https connection.

Our server-side script consists of 132 lines of code. Further, our verification program consists of 356 LoC, which includes nonce generation and verification of the client TPM’s Quote operation and AIK certificate.

As soon as a client confirms a specific transaction and engages in the verification protocol, the server-side script invokes a local verification program, that we have developed, which generates a nonce for that particular user. The nonce is used to ensure the freshness of the forthcoming attestation of the UTP confirmation session on the client. The verification program also computes the expected attestation result with the nonce and the hash of the generated confirmation message. Those two hashes are later verified to be included in the hash chain that comprises the value in the TPM Quote, which is sent during attestation from the client.

After the confirmation session on the client, an attestation is sent back to the server, again through the https connection. When the server-side script receives the data it parses it and invokes the verification program to process the TPM Quote (AIK-signed PCR aggregates), and makes the decision as to whether this attestation represents a legitimate confirmation of the pending transaction. Depending on the result of the verification, the script sends a message to the client browser (that malware may suppress), stating whether or not the transaction is confirmed, and either commits or aborts the transaction in the web shop’s backend.

```
Confirmation Agent

To confirm the purchase of the following 3 items:
1. Widget          50 $
2. Doodad          10 $
3. Thingamajig     50 $
-----
TOTAL              110 $

Please type this in exactly: 3e

>: 3e

You typed in: 3e
Transaction will be confirmed.
```

Figure 4. Sample UTP confirmation display.

Client PC: We have developed an extension for the Firefox web browser, which runs on the untrusted OS on the client and is capable of sending and receiving data through the existing https connection. We have also developed a local

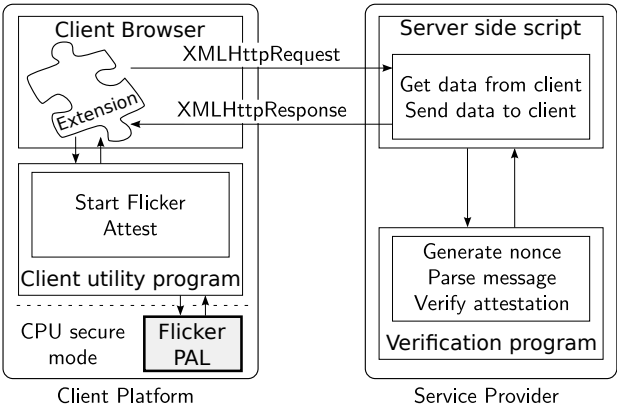


Figure 5. Implemented modules in the prototype.

utility program that is capable of invoking a UTP-specific, security-sensitive PAL (Piece of Application Logic) to run in a Flicker-based secure execution mode that is realized using Intel TXT [15]. Additionally, the program performs the attestation after the UTP Flicker session has completed.

In the common usage scenario, the user opens the web site of the online web shop in Firefox and selects items to buy. When the user is ready to submit her order, she presses the “Submit” button on the final summary page of the web shop. The extension then detects that this button has been clicked (because it has a special ID) and connects to the web server to request a nonce and a confirmation message. When the extension receives the requested files, it calls the client utility program, which invokes text mode in the graphics card⁵ and formats the received data as inputs for a Flicker session. Note that leveraging the untrusted code to invoke text mode simplifies the work that must be done inside the Flicker-protected session. When invoked, the Flicker framework executes using TXT-based hardware protections, with the UTP module as its PAL. Figure 5 shows the additional modules we have implemented.

Table I shows the lines of code (LoC) for the implemented modules on the client side, as well as the code size of the existing Flicker framework that we build upon and that belongs to the TCB of our UTP agent. The overall size of the TCB (including drivers for user I/O) is only a few thousand LoC.

Confirmation Agent (client-side): When invoked, the UTP module displays the confirmation message. It prompts the user to enter a randomly chosen character sequence (to avoid habituation by simply hitting ENTER). Figure 4 shows our sample confirmation dialog.

When the correct character sequence has been entered, UTP extends PCR 19 with the nonce and the confirmation

⁵We have used text mode for simplicity, since a basic VGA driver is straightforward to implement inside Flicker. A fully graphical confirmation page could also be implemented, but it would require additional support inside Flicker for the graphics mode(s) of interest.

	Module	Language	LoC
untrusted	Client utility program	C	321
	Firefox extension	JavaScript	95
	AIK generation program	C	540
	Sum		956
trusted	Keyboard & display drivers ^a	C	254
	Flicker PAL	C	260
	Flicker TCB	C/Assembly	741
	UTP helper functions	C	1080
	Sum		2335

Table I
CODE SIZES ON CLIENT SIDE.

^aThe given LoC include the PS/2 and VGA drivers. Support for USB keyboards would increase the code size by 1500-2000 LoC.

text string that was provided by the server. If an incorrect sequence is entered (or if the user chooses not to confirm the transaction), then a failure message is extended into the PCR. Finally, a special value denoting that the secure session is ending is extended into PCRs 18 and 19. Though a well-behaved OS will prevent application software from accessing the TPM at localities 2 and 3, a malicious OS can access those addresses. On Intel hardware, PCR 17 can only be extended by Intel's SINIT Authenticated Code Module (we use Q35_SINIT_17.BIN for our prototype).

After the UTP module returns, the utility program requests a TPM Quote that includes PCRs 17 and 18 (containing measurements of Flicker and the UTP code), and PCR 19, which contains the hashed nonce and confirmation text string. The nonce will also be used by the TPM for signature generation during the attestation operation.

Confirmation Verification (server-side): The verification script receives the data and verifies it according to the following criteria:

- The signature is valid and from a registered TPM (i.e., one that has been registered during the enrollment phase). We currently authenticate the client device based on a simple whitelist of allowed devices, realized as a list of known public AIKs. We note that more sophisticated authentication methods are readily applicable (e.g., [23], [24], [25]).
- The PCR values for Flicker and the UTP code are correct in PCRs 17 and 18, i.e., they are on the known-good list.
- The PCR value of the nonce and confirmation text string in PCR 19 is correct, i.e., corresponding to those sent by the verifier for this transaction confirmation.

When all checks are successfully performed, the transaction is accepted by the web server, and the user will see a corresponding message in the (well-behaved) web browser.

Detailed PCR Contents: Here we describe in detail the exact data that is extended into PCRs 17–19 in our prototype.

This helps to illustrate precisely what code and operations comprise the software Trusted Computing Base (TCB) for the UTP verification session.

- PCR 17: (these values represent the SINIT module for SinitMleData.Version 6 [10]):

```
Extend(SHA1(SinitMleData.SinitHash |
            SinitMleData.EdxSenterFlags))
Extend(SHA1(SinitMleData.BiosAcm.ID |
            SinitMleData.MsegValid |
            SinitMleData.StmHash |
            SinitMleData.PolicyControl |
            SinitMleData.LcpPolicyHash |
            (OsSinitData.Capabilities, 0)))
```

- PCR 18: (this is precisely the hash of the Flicker-based PAL that we have written for UTP):

```
Extend(SinitMleData.MleHash)
```

- PCR 19: (these are the input and output parameters of the UTP Flicker session, where 1 means confirmed, and 0 means not confirmed):

```
Extend(1|0); Extend(nonce); Extend(SHA1(ConfMsg))
```

C. Deployment

The design of our UTP confirmation agent is generic and not service provider-specific. Since the confirmation message is provided by the server for each transaction, the service providers can adapt the content or look and feel of these messages without the need to redesign the UTP agent. The end user is not required to have any understanding of the code that runs in secure mode, nor is there any need to do per-system or per-TPM generation of expected hashes.

To use our UTP agent, the user only needs to download the agent code and the browser extension and install it on an OS that supports the Flicker framework (currently Linux). The same agent code and browser extension can be reused across many platforms. There may need to be a few variants (e.g., Windows vs Linux, Intel vs AMD), but we do not foresee a significant scalability problem.

To support a wide range of hardware for user input, our prototype includes drivers for VGA display and PS/2 keyboards, and optionally USB keyboards. On PC-class platforms there is a basic set of VGA and PS/2 I/O available (even on systems with USB keyboards). These can be used as a widely available fall-back in the event that more sophisticated drivers for a particular platform are not available. Our current prototype is built with the assumption that the OS is the lowest layer of system software. However, Flicker and UTP can coexist with virtualization software. A hypervisor such as Xen [36] could be readily modified to enable Flicker and thus our UTP agent.

D. Performance Evaluation

We have implemented and tested our prototype on a Dell Optiplex 755, with a 3.0 GHz Intel Core2 Duo E6850 and ST Microelectronics v1.2 TPM. We run Ubuntu 9.04 (i386) and Linux kernel 2.6.30-6.

Figure 6 shows the timeline of a typical transaction confirmation. While the system waits for user confirmation in secure mode, the rest of the OS is halted, and thus the user's current work is suspended. However, all of the steps before entering secure mode and after resuming the system happen in background and are transparent to the user. There is a delay time of approximately one second before the actual confirmation summary is displayed. This time includes the time to switch the graphics card to text mode, enter secure mode, and initialize the video device in the Flicker PAL. However, such a short delay is quite common across web sites when waiting for a summary of a transaction.

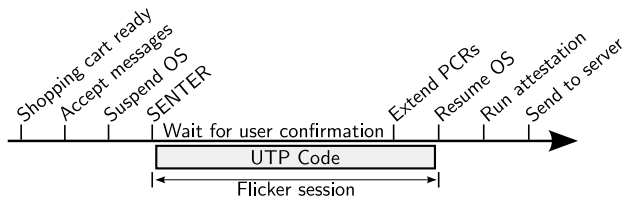


Figure 6. Timeline of a typical UTP session.

V. SECURITY ANALYSIS

The security of our approach relies mainly on the following three properties: (1) the server accepts only transactions that are properly confirmed; (2) the UTP agent is executed in an isolated mode where it is protected against software-based tampering, and where it solely controls the user-centric I/O devices (keyboard, screen, mouse); and (3) the remote server can get assurance as to whether the confirmation agent was actually executed in the secure mode and that the user indeed confirmed the transaction via the keyboard. Also required is that the user actually read the transaction summary. We discuss different attack scenarios in the following.

A. Transactions Must Be Confirmed

The essence of our approach is that a specially designed confirmation agent is executed in an isolated environment on the client system to enable the user to confirm a proposed transaction. Given the strength of our adversary (superuser privileges on the legacy OS), we ultimately depend on the user during a final confirmation step. Thus, the usability of the confirmation procedure is of utmost importance.

We also briefly comment on the ability of attackers to attach malicious input devices or compromise other peripheral devices and reprogram them to masquerade as input devices. While heuristics such as refusing to verify a transaction on a system with more than one USB keyboard attached may offer some level of protection, fundamentally this is a significant challenge. However, solutions where the user must transcribe a few digits from the display again raises the bar for attackers, as they would also need a malicious video

interface card. This is a significantly more complex task than compromising [37] or impersonating a USB keyboard.

In our implementation, the server sends the confirmation message to be displayed by the UTP confirmation agent. The agent displays the message, and requests that the user enter a few (randomly chosen) characters as confirmation. The adversary (e.g., malware on the user's computer) may modify a user-generated, or construct a new, transaction request and send it to the server. He has this power because he can control the execution of the web browser and OS as long as the isolated execution environment is inactive.

Though the details of the malicious transaction are displayed on-screen, the user could potentially confirm the modified transaction through an oversight. For example, a study of transaction authentication via mobile phones as used in some online banking applications has shown that users do not always check the confirmation message carefully enough [38]. Thus, the particular details of the displayed message, and the actions required of the user that will be interpreted as confirmation, are of great interest.

Here, we focus on the technical realization of the one-way trusted path to realize secure transaction confirmation. The definition of confirmation messages is up to the service provider. The transaction confirmation may be improved by letting the user enter a transaction-specific detail as confirmation instead of a few random characters. Researchers have shown that users' intentions are more accurately captured when the user actively re-selects the destination of a transaction from a list of different destinations [29].

B. Isolated Execution

Our confirmation agent executes with hardware-enforced isolation, taking advantage of Intel TXT and the Flicker architecture to protect itself from all other code and devices on the system. Our confirmation agent is specifically constructed to take control of the user-centric I/O devices, i.e., the keyboard and graphics display. The adversary has two options for tampering with the confirmation agent: (1) manipulate or replace the confirmation agent code when it is going to be executed, or (2) execute an entirely different application that fakes the confirmation agent display on screen (thereby fooling the user).

In case (1) the malicious agent will execute, and the user may be tricked into confirming a different transaction than the one that the malicious code will attempt to submit to the web server. However, activation of the secure mode of Intel TXT will result in hashing the agent's code into PCR 18. We show in §V-C that this will be detected by the web server, which will invalidate the transaction.

In case (2) neither the confirmation agent nor the isolated execution mode are invoked. Instead, an application in the untrusted mode simulates the confirmation dialog. The user is not able to detect a faked dialog since we do not have a full trusted path with the ability to authenticate the currently

running application to the user.⁶ However, even if the user confirms the faked confirmation dialog, the malicious code is unable to cause PCR 18 to reset and be extended with the hash of the confirmation agent expected by the server. Again, §V-C shows how the web server detects this.

C. Remote Attestation

As our adversary has superuser privileges on the untrusted OS, he may be able to invoke an isolated execution environment with code and inputs of his choosing. Or, he may fake the entire process from the perspective of the user. In either case, our defense rests upon the contents of the PCRs in the client system's TPM. Specifically, PCRs 17–19 contain (respectively) hashes of the chipset-specific signed code from the chipset vendor (e.g., Intel), the hash of the confirmation agent code itself, and hashes summarizing the nonce, confirmation string that was displayed to the user, and the user's response.

PCR 17 can only be extended by the CPU or Intel-signed SINIT Authenticated Code Module. Thus, the remote server can learn precisely what code was invoked in the isolated environment and is able to detect unintended code execution when it verifies the values during the attestation process. This gives the web server the ability to detect and deny transactions that originate on hardware with known chipset or CPU vulnerabilities. Perhaps more importantly, it also enables the web server to detect that no such operation took place, i.e., that the isolated environment was *not* invoked.

PCR 18 is extended as part of the dynamic root of trust operation that invokes the isolated execution environment (e.g., GETSEC[SENDER]). It will contain the hash of precisely the code that received control following the reset of the dynamic PCRs (including 17–19) and platform reinitialization by the Authenticated Code Module. On a well-behaved system, this will be the hash of the UTP confirmation agent code. Again, the web server can verify the presence of known-good code. The presence of other code, or the absence of any measurement, signify an attack.

PCR 19 is used to summarize the anti-replay nonce provided by the web server prior to confirmation, the confirmation message itself, and the user's input in response to the confirmation message. The user's input must arrive via the keyboard driver in the isolated execution environment. Thus, solely software-based attacks are prevented. An adversary's best course of action is to physically compromise the keyboard, which is a significantly more difficult attack to perpetrate without physical access to the system.

D. Denial of Service

Instead of manipulating a transaction confirmation, the adversary can block a pending confirmation by preventing the UTP code from running on the client. The server will

not receive any confirmation and abort the transaction after a timeout. This is a denial-of-service (DoS) attack and outside the scope of this paper.

VI. RELATED WORK

E-EMV [39] is a software-based credit card application to secure transaction confirmation, similar to our UTP agent. They also use an enrollment phase where an AIK of the user's TPM is certified by a Privacy CA or in a DAA group. However, they do not combine transaction details with an explicit user interaction. Users can still be tricked into authorizing malicious transactions. Our approach includes a transaction summary and user-entered text into the attestation. Moreover, their approach needs a persistently running security kernel and a full trusted path.

Phoolproof Phishing [40] seeks to strengthen user authentication to web sites by leveraging users' smartphones as a store for public key-based authentication credentials that serve as an additional factor for authentication. Dedicated hardware solutions for transaction confirmation have also been proposed. The IBM ZTIC [41] is a small USB device that is assumed to share cryptographic keys with the user's financial institution. It includes a screen to display transaction details to the user, and buttons for accepting or aborting the transaction. Our design is similar to both of these in that we also use public key-based authentication credentials stored in the system's TPM. However, our design has the advantage that the user is not required to carry around or manipulate multiple devices during a transaction.

Several wallet-based approaches [26], [27], [28], [29], [30] have been proposed to secure user authentication or protect login credentials. They either require a trustworthy external device as an out-of-band channel for authentication [28], [42], [43], or require a secure kernel [26], [27] that isolates the underlying operating system and its complex software stack from the wallet. Many existing authentication agent proposals are prototyped using off-the-shelf virtual machine monitors, which are not without their share of vulnerabilities, e.g., [44], [45], [46]. In contrast, TruWalletM [30] uses M-Shield [47], a hardware-provided secure execution mode on a mobile phone, to protect login credentials of a user. The wallet is only invoked during login and executed in isolation in the secure mode. Although some use cases of UTP require authentication, this is orthogonal to our work. It should be a moderate effort to port their authentication agent code to Flicker and, hence, integrate them for the authentication step in our design.

SpyBlock [27] is a browser extension that requests authentication and confirmation of transactions from the user in a separate confirmation agent. However, the confirmation agent relies on a trusted window (full trusted path) and a secure hypervisor platform. In contrast, in our design we use the secure execution mode of the CPU to enable UTP

⁶This is where a "secret picture" (or similar) mechanism can be a great aid if users can be trained to exhibit sufficient diligence.

to execute exclusively during confirmation. Hence, the confirmation agent in our design depends on less trusted coded and is less disruptive to existing software environments.

The Not-a-Bot system employs an attester that certifies network traffic as originating in a system within some threshold time period after legitimate user input [48]. While an interesting design, the presented prototype includes a hypervisor and additional operating systems and thus includes a large TCB. Further, the binding between traffic and user input is temporal only – malware can readily wait and send all of its traffic only while the user types.

Grawrock describes the concept of a *Verification Model* possible on platforms that include support for dynamic root of trust [49]. The system that we describe in this paper can be considered a concrete realization of these abstract concepts, complete with evaluation and several use cases.

Dynamic root of trust on x86 systems is not the only mechanism available today for isolated execution. For example, ARM TrustZone [50] and Texas Instruments' M-Shield [47] offer similar functionalities, especially when paired with a Mobile Trusted Monitor. Nokia's OnBoard Credentials project explores some of the capabilities of these platforms [51]. The Cell Broadband Engine [52], used in Sony's PlayStation 3, offers eight processing cores that can be operated in a secure execution mode, isolated from each other and from the main core that runs the OS. We believe that our proposed solution should be portable to those platform architectures with manageable effort.

VII. CONCLUSION

We show how the combination of an on-demand isolated execution environment and temporal control of user-centric I/O devices enables construction of a mechanism for a one-way trusted path, and show that it is practical and deployable on commodity systems today. The uni-directional trusted path constructed by our system extends from the client system to the remote server. Although the immediate feedback available to the client user remains susceptible to manipulation by malware, our system enables service providers to gain additional assurance that client transactions are initiated at the behest of a human user on the computer, and not via malware such as transaction generators. Over the long term, users will be less likely to become the victims of a scam. Hence, service providers have high incentives for deployment, as they can simply offer UTP as a download for their users.

VIII. ACKNOWLEDGMENTS

We wish to thank Ahren Studer, Jim Newsome, and the anonymous reviewers for their valuable comments, which greatly improved the paper.

REFERENCES

- [1] CNET, "Pop-up program reads keystrokes, steals passwords," 2004, http://news.cnet.com/2100-7349_3-5251981.html.
- [2] —, "New Trojans plunder bank accounts," 2006, http://news.cnet.com/2100-7349_3-6041173.html.
- [3] W3C Working Group, "Inaccessibility of CAPTCHA," Note 23, Nov. 2005.
- [4] J. Epstein, J. McHugh, H. Orman, R. Pascale, A. Marmor-Squires, B. Danner, C. R. Martin, M. Branstad, G. Benson, and D. Rothnie, "A high assurance window system prototype," *Journal of Computer Security*, vol. 2, no. 2, 1993.
- [5] N. Feske and C. Helmuth, "A Nitpicker's guide to a minimal-complexity secure GUI," in *Proc. Annual Computer Security Applications Conference (ACSAC)*, 2005.
- [6] J. S. Shapiro, J. Vanderburgh, and E. Northup, "Design of the EROS trusted window system," in *Proc. USENIX Security*, 2004.
- [7] TrustMark, "TrustMark banking and financial solutions," <http://www.trustmark.com/passmark/index.html>, Aug. 2010.
- [8] B. Lampson, "Usable security: How to get it," *Communications of the ACM*, vol. 52, no. 11, 2009.
- [9] AMD, "AMD64 virtualization codenamed "Pacifica" technology — secure virtual machine architecture reference manual," AMD, Tech. Rep. Publication Number 33047, Revision 3.01, May 2005.
- [10] Intel Corporation, "Intel trusted execution technology MLE developer's guide," Intel Corporation, Tech. Rep. Document Number: 315168-006, Dec. 2009.
- [11] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2001.
- [12] L. F. Cranor, "What do they indicate? evaluating security and privacy indicators," *Interactions*, vol. 13, no. 3, 2006.
- [13] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The emperor's new security indicators," in *Proc. IEEE Symposium on Security and Privacy*, 2007.
- [14] M. Wu, R. C. Miller, and S. L. Garfinkel, "Do security toolbars actually prevent phishing attacks?" in *Proc. Conference on Human Factors in Computing Systems (CHI)*, 2006.
- [15] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: An execution infrastructure for TCB minimization," in *Proc. ACM SIGOPS European Conference on Computer Systems (EuroSys)*, 2008.
- [16] Trusted Computing Group, "TPM main specification, version 1.2, revision 103," Jul. 2007.
- [17] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: a virtual machine-based platform for trusted computing," in *Proc. ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
- [18] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. van Doorn, J. L. Griffin, and S. Berger, "sHype: Secure hypervisor approach to trusted virtualized systems," IBM Research Division, Tech. Rep. RC23511, Feb. 2005.
- [19] European Multilaterally Secure Computing Base Project, "Turaya," <http://www.emscb.org/content/pages/turaya.htm>.
- [20] OpenTC Project Consortium, "Open Trusted Computing," <http://www.opentc.net>.
- [21] K.-P. Yee, D. Wagner, M. Hearst, and S. M. Bellovin, "Prerendered user interfaces for higher-assurance electronic voting," in *Proc. USENIX Electronic Voting Technology Workshop*, 2006.
- [22] C. Doctorow, "Solving and creating captchas with free porn," Boing Boing, Jan. 2004.
- [23] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proc. IEEE Symposium on Security and Privacy*, 1992.

- [24] D. P. Jablon, "Strong password-only authenticated key exchange," *Computer Communication Review*, vol. 26, 1996.
- [25] T. Wu, "The secure remote password protocol," in *Proc. Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 1998.
- [26] S. Gajek, A.-R. Sadeghi, C. Stübke, and M. Winandy, "Compartmented security for browsers – or how to thwart a phisher with trusted computing," in *Proc. Conference on Availability, Reliability and Security (ARES)*, 2007.
- [27] C. Jackson, D. Boneh, and J. Mitchell, "Spyware resistant web authentication using virtual machines," <http://crypto.stanford.edu/spyblock/>, 2006.
- [28] R. C. Jammalamadaka, T. W. van der Horst, S. Mehrotra, K. E. Seamons, and N. Venkasubramanian, "Delegate: A proxy based architecture for secure website access from an untrusted machine," in *Proc. Annual Computer Security Applications Conference (ACSAC)*, 2006.
- [29] M. Wu, R. C. Miller, and G. Little, "Web Wallet: Preventing Phishing Attacks by Revealing User Intentions," in *Proc. Symposium on Usable Privacy and Security (SOUPS)*, 2006.
- [30] S. Bugiel, A. Dmitrienko, K. Kostianen, A.-R. Sadeghi, and M. Winandy, "TruWalletM: Secure web authentication on mobile platforms," in *Proc. Trusted Systems, Second International Conference (INTRUST)*, 2010.
- [31] OASIS, "Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0," OASIS Standard saml-core-2.0-os, Mar. 2005.
- [32] OpenID Foundation, "OpenID authentication 2.0 - final specification," Dec. 2007.
- [33] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2004.
- [34] E. Cesena, H. Löhr, G. Ramunno, A.-R. Sadeghi, and D. Vernizzi, "Anonymous authentication with TLS and DAA," in *Proc. Conference on Trust and Trustworthy Computing (TRUST)*, 2010.
- [35] FreeWebShop.org, <http://www.freewebshop.org>.
- [36] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer, "Xen and the art of virtualization," in *Proc. ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
- [37] K. Chen, "Reversing and exploiting an Apple firmware update," in *Black Hat*, 2009.
- [38] M. AlZomai, B. AlFayyadh, A. Jøsang, and A. McCullagh, "An experimental investigation of the usability of transaction authorization in online bank security systems," in *Proc. Australasian Information Security Conference (ACSC)*, 2008.
- [39] S. Balfe and K. G. Paterson, "e-EMV: Emulating EMV for internet payments with trusted computing technologies," in *Proc. ACM Workshop on Scalable Trusted Computing (STC)*, 2008.
- [40] B. Parno, C. Kuo, and A. Perrig, "Phoolproof phishing prevention," in *Proc. Financial Cryptography and Data Security Conference*, 2006.
- [41] IBM Zurich Research Lab, "Security on a stick," Press release, Oct. 2008.
- [42] A. Vapen, D. Byers, and N. Shahmehri, "2-clickAuth - optical challenge-response authentication," in *Proc. Conference on Availability, Reliability and Security (ARES)*, 2010.
- [43] T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch, "The Zurich Trusted Information Channel – an efficient defence against man-in-the-middle and malicious software attacks," in *Proc. Conference on Trust and Trustworthy Computing (TRUST)*, 2008.
- [44] "Elevated privileges," CVE-2007-4993, 2007.
- [45] "The CPU hardware emulation does not properly handle the Trap flag," CVE-2008-4915 (under review), 2008.
- [46] R. Wojtczuk, "Subverting the Xen hypervisor," Invisible Things Lab, 2008.
- [47] J. Azema and G. Fayad, "M-Shield mobile security technology: making wireless secure," Texas Instruments, Feb. 2008, http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf.
- [48] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy, "Not-a-Bot (NAB): Improving Service Availability in the Face of Botnet Attacks," in *Proc. Network Systems Design and Implementation (NSDI)*, Apr. 2009.
- [49] D. Grawrock, *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, 2008.
- [50] T. Alves and D. Felton, "TrustZone: Integrated hardware and software security," *Information Quarterly*, vol. 3, no. 4, 2004.
- [51] K. Kostianen, J.-E. Ekberg, N. Asokan, and A. Rantala, "On-board credentials with open provisioning," in *Proc. Symposium on Information, Computer, and Communications Security (ASIACCS)*, 2009.
- [52] K. Shimizu, "The Cell Broadband Engine processor security architecture," <http://www.ibm.com/developerworks/power/library/pa-cellsecurity/>, Apr 2006.

APPENDIX

The Trusted Platform Module is an inexpensive, passive device that can serve as a hardware root of trust [16]. TPMs include a bank of Platform Configuration Registers (PCRs) that can be *extended* with *measurements* of code or data. A measurement $m \leftarrow \text{SHA1}(\text{data})$ represents a cryptographic hash over an object of interest. An extend operation is defined as $\text{PCR}_{\text{new}} \leftarrow \text{SHA1}(\text{PCR}_{\text{old}} || m)$.

After measurements are accumulated into a hash chain inside one or more PCRs, *sealing* and *attestation* become possible. In a seal operation, a cryptographic key is access-controlled based on the values of one or more PCRs. With an appropriate software architecture, this can render data available only to certain software stacks. In an attestation, a digitally signed PCR aggregate called a *Quote* is produced. The asymmetric key used to sign the Quote is called an Attestation Identity Key (AIK), which can be linked to a particular TPM instance using the Endorsement Key (EK) certificate provided by reputable TPM vendors.

Newer x86 platforms (circa 2007) equipped with a v1.2 TPM and certain CPU and chipset enhancements can also create a *dynamic* root of trust for measurement (DRTM). A DRTM operation (1) reinitializes the CPU and memory controller into a known-good state and (2) resets *dynamic* PCRs inside the TPM into a distinguished state, enabling the possibility to bootstrap isolated execution with data sealing and attestation capabilities even in the face of a compromised OS or virtual machine monitor (VMM).

EXHIBIT G

[Browse](#) [My Settings](#) [Help](#)

FINNEGAN

Access provided by:
Finnegan Henderson Farabow

FINNEGAN

Access provided by:
Finnegan Henderson Farabow Garrett & Dunner[File Cabinet](#)[Contact Administrator](#)
[Sign Out](#)[Show Usage](#)

All



ADVANCED SEARCH

[Conferences](#) > [2011 IEEE/IFIP 41st Internati...](#) [?](#)

Uni-directional trusted path: Transaction confirmation on just one device

Publisher: IEEE

[Cite This](#)

PDF

Atanas Filyanov ; Jonathan M. McCuney ; Ahmad-Reza Sadeghiz ; Marcel Winandy **All Authors** ...16
Cites in
Papers1
Cites in
Patent321
Full
Text Views

Alerts

[Manage Content Alerts](#)[Add to Citation Alerts](#)

Abstract

Document Sections

- I. Introduction
- II. Problem Definition
- III. Architecture and Design
- IV. Implementation
- V. Security Analysis

[Show Full Outline](#)[Authors](#)[Figures](#)[References](#)[Citations](#)[Keywords](#)[Metrics](#)[More Like This](#)

Downl

PDF

Abstract:

Commodity computer systems today do not include a full trusted path capability. Consequently, malware can control the user's input and output in order to reveal sensitive... [View more](#)

▼ Metadata

Abstract:

Commodity computer systems today do not include a full trusted path capability. Consequently, malware can control the user's input and output in order to reveal sensitive information to malicious parties or to generate manipulated transaction requests to service providers. Recent hardware offers compelling features for remote attestation and isolated code execution, however, these mechanisms are not widely used in deployed systems to date. We show how to leverage these mechanisms to establish a "one-way" trusted path allowing service providers to gain assurance that users' transactions were indeed submitted by a human operating the computer, instead of by malware such as transaction generators. We design, implement, and evaluate our solution, and argue that it is practical and offers immediate value in e-commerce, as a replacement for captchas, and in other Internet scenarios.

Published in: 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)**Date of Conference:** 27-30 June 2011**DOI:** 10.1109/DSN.2011.5958202**Date Added to IEEE Xplore:** 18 July 2011**Publisher:** IEEE

☰ Contents

I. Introduction

For decades a significant challenge in computer security has been realizing a mechanism for establishing a full trusted path, i.e., a mechanism that (i) isolates the input and output channels of different applications to preserve the integrity and confidentiality of data exchanged with the user, (ii) provides a technical means that assures the user of a computer system that she is truly interacting with the intended software, and (iii) assures running applications that user inputs truly originate from the actions of a human (as opposed to being synthesized or injected by other software).

Authors	▼
Figures	▼
References	▼
Citations	▼
Keywords	▼
Metrics	▼
Footnotes	▼

More Like This

An Adaptive Server Side Software Authentication Framework Based on User's Activity Pattern
2011 Second International Conference on Emerging Applications of Information Technology
Published: 2011

HSC-IoT: A Hardware and Software Co-Verification Based Authentication Scheme for Internet of Things
2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)
Published: 2017

Show More



IEEE Personal Account

CHANGE
USERNAME/PASSWORD

Purchase Details

PAYMENT OPTIONS
VIEW PURCHASED
DOCUMENTS

Profile Information

COMMUNICATIONS
PREFERENCES
PROFESSION AND
EDUCATION
TECHNICAL INTERESTS

Need Help?

US & CANADA: +1 800
678 4333
WORLDWIDE: +1 732
981 0060
CONTACT & SUPPORT

Follow



[About IEEE Xplore](#) | [Contact Us](#) | [Help](#) | [Accessibility](#) | [Terms of Use](#) | [Nondiscrimination Policy](#) | [IEEE Ethics Reporting](#)  | [Sitemap](#) | [IEEE Privacy Policy](#)

A not-for-profit organization, IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.

© Copyright 2024 IEEE - All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies.

IEEE Account

- » [Change Username/Password](#)
- » [Update Address](#)

Purchase Details

- » [Payment Options](#)
- » [Order History](#)
- » [View Purchased Documents](#)

Profile Information

- » [Communications Preferences](#)
- » [Profession and Education](#)

» [Technical Interests](#)

Need Help?

» **US & Canada:** +1 800 678 4333

» **Worldwide:** +1 732 981 0060

» [Contact & Support](#)

[About IEEE Xplore](#) | [Contact Us](#) | [Help](#) | [Accessibility](#) | [Terms of Use](#) | [Nondiscrimination Policy](#) | [Sitemap](#) | [Privacy & Opting Out of Cookies](#)

A not-for-profit organization, IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.

© Copyright 2024 IEEE - All rights reserved. Use of this web site signifies your agreement to the terms and conditions.

EXHIBIT H

Achieving Attestation with Less Effort: An Indirect and Configurable Approach to Integrity Reporting

Jun Ho Huh
Information Trust Institute
University of Illinois at
Urbana-Champaign
Urbana, IL, USA
jhuh@illinois.edu

Hyoungshick Kim
Computer Laboratory
University of Cambridge
Cambridge, UK
hk331@cl.cam.ac.uk

John Lyle
Oxford University Computing
Laboratory
Oxford, UK
john.lyle@comlab.ox.ac.uk

Andrew Martin
Oxford University Computing
Laboratory
Oxford, UK
andrew.martin@comlab.ox.ac.uk

ABSTRACT

This paper proposes an *indirect attestation* paradigm for verifying the trustworthiness of end user platforms. This approach overcomes several criticisms of attestation by maintaining the user's freedom to choose their own software configurations and minimising the whitelist management overhead for the relying party. Each user platform defines its own acceptable software combination in terms of reference integrity measurements, and reports the local verification results to the relying party through a late-launched, trusted Platform Trust Service. The relying party simply checks this verification result and a *security meta-policy* that has been used to ensure the quality of the security checks performed locally. The Platform Trust Service is also responsible for reporting whether this meta-policy is satisfied. By configuring the meta-policy, the relying party selects an indirect attestation paradigm that best meets their high-level security requirements.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection; K.6.5 [Management of Computing and Information Systems]: Security and Protection

General Terms

Design, Security

Keywords

indirect attestation, whitelist management, security meta-policy

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

STC'11, October 17, 2011, Chicago, Illinois, USA.

Copyright 2011 ACM 978-1-4503-1001-7/11/10 ...\$10.00.

1. INTRODUCTION

Trusted Computing (TC) provides a means to reliably report the integrity of platform configurations to a remote party – this capability is referred to as ‘remote attestation’ by the Trusted Computing Group (TCG) [2]. Remote attestation has often been suggested as a mechanism suitable for authenticating user platforms: an obvious application is a web server that verifies the user's platform integrity and grants access to only those running with trustworthy configurations. Such an attestation system can be used together with protected key storage to build a strong two-factor authentication mechanism – the user's private signature key can be strongly protected through a Trusted Platform Module (TPM), which is normally a hardware chip embedded in the motherboard.

Despite the potential security benefits, there are some inherent practicality and usability issues associated with attestation which can be overlooked – these issues often impede further advances in attesting/verifying user platform configurations in systems that manage large number of users. For instance, one challenge of implementing attestation for an online system such as Internet banking is the management of application whitelists [8]. Pre-defining and validating which configurations/combinations are acceptable and keeping the whitelists up to date would be very costly tasks for the service providers. Agility is another potential problem: if the user configurations are patched, the service provider must accept those patched systems immediately, which may be hard to do in practice. With enforcement of these whitelists, there is also the danger of *locking* the user into a particular platform configuration – this would severely affect usability and privacy. Moreover, the process of maintaining and distributing Trusted Computing software (required for attestation) would have to be managed efficiently and be transparent to the end users. Perhaps it is these overwhelming challenges that discourage researchers and service providers from further exploring the idea of attesting user platform configurations. Without further investigation, attestation in large user-base systems may remain as a theoretical concept forever.

It is in this spirit that we propose a configurable approach for selecting an *indirect attestation* paradigm that is suitable

for the security needs of the service provider. Our approach would give more freedom and privacy to the users in terms of choosing their own acceptable platform configurations. The costs associated with maintaining the whitelists and performing attestation are spread to the end users, software vendors, and security companies. We also evaluate security and usability of the proposed indirect attestation system, and discuss the implications of using it.

The following section explains the Trusted Computing concepts. Section 3 explains the challenges of making attestation work, and Section 4 proposes our indirect attestation system. We then explain to what extent the whitelist management and platform locking (software freedom) issues can be resolved in Section 5. Our conclusions are at Section 7.

2. TRUSTED COMPUTING CONCEPTS

Trusted Computing provides a means to *measure* the software loaded during the platform's boot process. Measurements are taken by calculating a cryptographic hash of binaries before they are executed. Hashes are stored in Platform Configuration Registers (PCRs) in the TPM. They can only be modified through special TPM ordinals, and the PCRs are never directly written to; rather, measurements can only be *extended* by an entity. This is to ensure that no other entity can just modify or overwrite the measure value. As a result, every executable piece of code in this *authenticated boot* process will be measured and PCRs extended sequentially (*transitive trust*). Any malicious piece of code (e.g. rootkit) executed during the boot process will also be recorded and identified. A 'PCR event log' is created during the boot process and stores all of the measured values (and a description for each) externally to the TPM. These values can be extended in software to validate the contents of the event log. The resulting hash can be compared against the reported, signed PCR values to see if the event log is correct.

2.1 Remote attestation

Remote attestation involves the TPM reporting PCR value(s) that are digitally signed with TPM-generated 'Attestation Identity Keys' (AIKs), and allowing others to validate the signature and the PCR contents [6]. The relying party validates the signature by using the public part of the AIK and validates the AIK with the AIK credential. The PCR log entries are then compared against a list of 'known-good' *reference values* to check if the reported PCRs represent an acceptable configuration. This list is often referred to as an 'application whitelist'. Attestation can be used on a platform that supports authenticated boot to verify that only known pieces of software are running on it.

2.2 Late launch

CPUs from both Intel and AMD now allow an arbitrary piece of code to be 'late-launched' [6] on an untrusted platform. Starting from a dynamic root of trust (which can be initiated anytime after platform boot), this code is executed in a protected environment and measured without any interference from the software currently running. On an Intel platform, PCRs 17-20 are reset and used to store the measurements of the secure initialization (SINIT) module and the trustworthy code to be late-launched. The SINIT module is responsible for measuring and launching the trustworthy code. This enables attestation of the SINIT module and trustworthy code. Attestation of the two components

typically involves sending a TPM Quote of PCRs 17 and 18 for an Intel platform or PCR 17 for an AMD platform. A remote verifier checks that these PCRs were reset properly and the hashes of the SINIT module and trustworthy code. In Section 4.2 we discuss how a late-launched Platform Trust Service can be used to report local security checks reliably to the service provider.

3. WHY IS ATTESTATION DIFFICULT IN PRACTICE?

3.1 Whitelist management issues

The reality is that people have not been able to make remote attestation work properly: modern operating systems and software are too complex and change too much. Although Microsoft Windows is still the dominant operating system, other operating systems like Linux and Mac OS X are also widely used. Furthermore, there are hundreds and thousands of different proprietary security software available (e.g. anti-virus and firewall) and a service provider might want to verify that the user is running some combination of these. We argue that it is infeasible for the service provider (such as the bank) to know about all different acceptable user platform configurations (operating systems, versions, and security software) and maintain the whitelist efficiently.

To make matters worse, this service provider would have to additionally manage old versions of operating systems and applications, since not all users update their systems in a timely manner. On top of this, the service provider might also want to verify the integrity of important configuration files. It is not just about the size of whitelists, but also the huge effort that needs to go into discovering new acceptable configurations and combinations, and keeping them up to date.

This is the first problem: no service provider would pay the cost of managing such activities and maintaining huge whitelists for enhancing security.

3.2 Platform locking issues

At first glance, mandating certain security configurations might seem like a practical solution to the whitelist management problems: the service provider would select a number of configurations that they consider trustworthy and force the users to use them to access their service. However, mandating the use of proprietary security software can impose severe usability and compatibility penalties, and must be treated with caution. There is the danger of *locking* people into a particular platform, depriving them of the option of using an online service via their choice of browser, operating system, and security software. A good example of this is the South Korean Internet banking system [7]: in order to use Internet banking, the users are obliged to install some number of proprietary security software such as anti-virus, firewall, and keystroke encryption software that are provided by the bank. These are typically installed on the user's machine in the form of ActiveX plugins. As a result, the users can only use Microsoft Internet Explorer to do Internet banking and have no option but to use Windows as their main operating system.

Likewise, mandating certain security configurations to facilitate meaningful attestation will have similar implications.

To most users, this concept of being obliged to use certain platform configurations, on their own PCs, is likely to be unacceptable. Due to poor usability and compatibility, some users may even move away to other banks that provide more compatible and open Internet banking services. Hence, no bank – unless required by some state law – would take this kind of risk just to improve security.

This is the second problem: most users will not tolerate the inconvenience of being obliged to use a certain configuration, and the service providers would be reluctant to frustrate their users to the extent that they might decide to use another service.

Moreover, insisting on certain platform configurations means that attackers know exactly what to look for vulnerabilities in, potentially making them less secure than they would otherwise be. With these issues in mind, the next section proposes a compromise solution for attestation that allows the users to select their own acceptable security software/configuration to use.

4. INDIRECT ATTESTATION PARADIGM SELECTION

We propose an indirect attestation system which allows users to set their own *security policy* and attest their conformance to relying service providers. Service providers are also able to configure and check certain high-level *meta-policies* about the user's security policy, without finding out exactly what it contains. By configuring the meta-policies, service providers select an indirect attestation paradigm that best meets their security needs as well as their clients' usability expectations.

4.1 User security policies

User security policies take the form of *program execution rules* and govern the identity of software trusted to run on the user's platform. For example, one form of policy would be a whitelist of programs trusted by the user. Another policy would be to specify which *vendors* of software are trusted, and then allow the platform to run any program which has a Reference Integrity Measurement (RIM) signed by them. A combination of these would allow the user to create a policy approving only a subset of the software signed by a certain vendor.

More complicated policies are also possible. A whitelist could be specified only up to a certain stage in the boot process – for example, until after the operating system is loaded – and then any software may be used. Alternatively, all applications running with super-user privilege might need to be whitelisted after the kernel, but all normal software may be unrestricted. There are numerous possibilities which could be explored as future work.

Perhaps a more important issue is how and by whom the security policies are created. The system we propose allows for the user to have complete control over their security policy. However, some users may prefer delegating the management to a trusted authority, such as an anti-virus company or ISP. In this case we imagine the trusted authorities regularly providing a signed whitelist of trusted software to be used as the security policy. Another alternative might be the provision of security policy templates, which allow the user to select the specific programs they trust.

4.2 Basic Attestation System

An overview of the proposed attestation system is shown in Figure 1. The key principle behind this system is that the relying service provider does not check the vast majority of the user's platform integrity measurements or necessarily insist on a specific operating system and software. Instead, the service provider checks that a known 'Platform Trust Service' (PTS) [1] component has been *late-launched* (see Section 2.2). The PTS is given the responsibility for checking the platform's other PCR values against the user-selected RIMs contained in the security policy. The service provider checks that the PTS has been run, that it has reported a 'PCR values OK' type of response, and that it is using an acceptable security meta-policy (see Section 4.3 for more details). The following protocol shows how the user and service provider interact. The AIK certificate is labelled as $\{\text{cert}(\text{AIK})\}_{PCA, \text{nonce}_{sp}}$ is a nonce generated by the service provider (SP), PCR_{17-20} is the attestation of the value of PCRs 17 to 20, PCRLog_{17-20} is the integrity measurement log for PCRs 17-20, and *meta-policy* is the meta-policy description. The basic attestation protocol is taken from Sailer et al. [10] and we assume that all communication takes place using transport-layer security:

1. $User \rightarrow SP$: Login Request , $\{\text{cert}(\text{AIK})\}_{PCA}$
2. $SP \rightarrow User$: nonce_{sp}
3. $User \rightarrow SP$: $\{\text{PCR}_{17-20}, \text{nonce}_{sp}\}_{AIK}$,
 PCRLog_{17-20} , *meta-policy*

At this stage, the service provider does all the usual checks including verifying the signatures on the AIK certificate, nonce and PCR values. Then the integrity measurement log for PCRs 17-20 is checked to make sure that a trusted PTS was properly late-launched – this log is checked against a fairly static and small list of known-good measurements for the PTS and SINIT module (see Section 2.2) that are managed by the service provider. The integrity of the PTS indicates the correctness of the reported security check/result. To minimise the chances of being compromised at runtime, the PTS should be designed as a small and simple piece software that only manages the following functions. First, it will check the user platform's PCRs against the user's security policy. Then it will *extend* into the PCRs the hash of any assertions made in the meta-policy, such as 'policy updated in the last month' or 'policy signed by authority X'¹ and check that the security policy satisfies the meta policy. It will then extend the result of this check: either a 'PCR values OK' or a 'PCR invalid' string. When the service provider receives the verified PCR log, it needs to check that it just consists of a 'PCR values OK' result and an *acceptable* meta-policy. This check would ensure the integrity of the user platform configuration to the extent specified in the security policy as well as the conformance with the meta-policy. Following a successful platform authentication, the service provider can go on to request authentication of the user as per normal.

4.3 Platform Trust Service and security meta-policies

As mentioned in the previous section, the Platform Trust Service is late-launched in this scenario to check the platform's PCRs against a user-defined security policy. It can

¹The precise vocabulary of the security policies and meta-policies is left as future work.

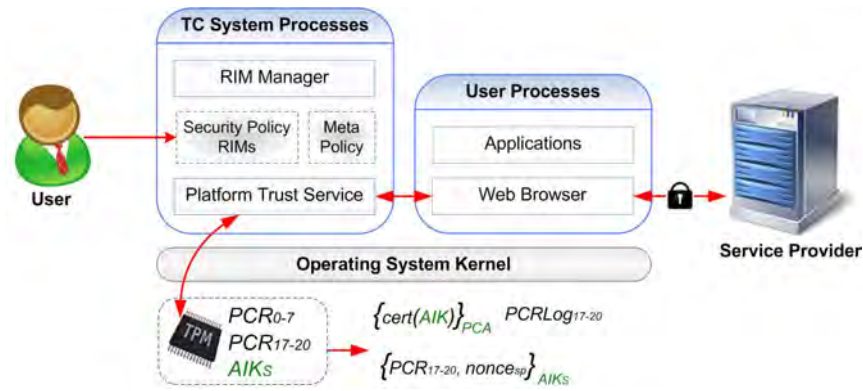


Figure 1: Indirect attestation overview. The late-launched Platform Trust Service is responsible for performing local configuration verification and reliably reporting the results to the service provider.

also check that the user's policy conforms with certain security meta-policies. These are designed to give the relying service provider more information about the user's platform without requiring them to use only software trusted by the service provider or compromising the user's privacy. We suggest the following meta-policies (listed in order of specificity) which provide useful security information.

P1. Non-trivial whitelisting

A whitelist is being checked by the PTS and this whitelist is non-trivially satisfied – i.e. the whitelist does not allow *all* software to be run and is of a reasonable size; this policy may also indicate that everything up to the OS kernel must be whitelisted.

P2. Whitelist signed by trusted authorities

A whitelist is being used by the platform and either this whitelist is signed by a trusted authority (the meta policy could state the authority) or each reference measurement in the whitelist is signed by a number of trusted authorities (again, stated in the meta policy). The service provider might be interested in knowing who the authority is, or whether it is one of the authorities on their list of trusted authorities.

P3. Recently updated security policy

The user has a security policy which has been updated after a given date. This could be implemented in a number of ways, but the easiest would be if the policy was signed by a trusted authority who also includes a timestamp. User-defined policies can also be timestamped (see 'policy provenance' below).

P4. Policy provenance

The meta-policy could also refer to the provenance of the user's security policy. This is required when the user does not use a whitelist signed by a trusted authority, but chooses his or her own selection of trusted software. How can the service provider know that the *user* created the whitelist, rather than a piece of malware running on the user's platform?

Clearly the solution is not to check the policy itself: the user must remain free to choose any security policy, even one

that some might consider to be insecure. Instead, we propose that the policy is demonstrated to have been updated only through trusted software, the *RIM Manager* (see Figure 1). This can be late-launched like the Platform Trust Service and provide an interface for users to change the list of parties who are trusted to sign RIMs.

The RIM Manager would finish by creating a TPM Quote containing the platform's PCR values and the hash of the created policy. If a timestamp is required, this could also be included by contacting a trusted external time server and using the TPM's tick counter.

P5. Trusted kernel and pre-kernel boot sequence

The meta-policy contains the identity of all the software which is used in the boot sequence up to and including the kernel. Following this, a whitelist is used but the content of it remains confidential. This works if the service provider believes that the operating system and boot sequence contains the full TCB of the platform. As a further elaboration, the meta-policy could state that all applications run with super-user privilege are also on a disclosed whitelist.

P6. Enforcing certain type of applications

This meta-policy states the *type* of security-critical applications that needs to be installed and loaded during the initial boot. For instance, a service provider might require that an anti-virus software and firewall are loaded on the user platform, without specifying the exact software. The integrity measurement log can be used to capture the type of software that has been loaded/measured – this information can be used by the PTS to check whether all of the required type of software has been loaded properly.

P7. Enforcing certain security configurations

Certain security configurations may also be stated as part of the meta-policy. For instance, a service provider might want to check that the automatic update feature is enabled on the user's OS and anti-virus software. This would be useful for specifying a more fine-grained set of security requirements. To satisfy this policy, the PTS would have to be equipped with a mechanism to scrutinize certain parts of the configuration files (or registries in Windows).

We imagine that some combination of these policies would

be used to build up and enforce different levels of security requirements. For instance, policies like P5 and P6 can be used together to ensure that a trusted kernel as well as anti-virus software and firewall are loaded on the user platform.

5. OBSERVATIONS

5.1 How the costs are spread

The usefulness of the proposed attestation system lies in the benefit each party gains from using it versus the associated costs. The user gains the ability to have a third party assess their system and report on whether their security policies are being implemented. This is possible, but difficult for users to do for themselves without a trusted external hardware token [5]. The costs to the user involve having a platform capable of late launch, and the overhead of managing a security policy. This may be negligible if the policy is provided by a trusted authority, but might be onerous otherwise. The service provider gains additional information about the user's platform state, and provides extra value (or a better service) for the user who may be willing to pay for the benefit. Moreover, the service provider can also check the security policy for conformance with meta-policies of varying levels of granularity. The cost to the service provider is minimal: they require knowledge of a small amount of trusted software, plus any trusted parties involved in the process. They may also need to provide support to the user if their platform fails the attestation verification.

5.2 What indirect attestation achieves

In Section 3 we identified whitelist management and platform locking as the two key problems of attestation. Based on the meta-policies suggested in Section 4.3, Table 1 evaluates to what extent these problems can be resolved using the proposed attestation system and what security properties can be ensured.

5.3 Security evaluation

As Table 1 shows, enforcing different level of meta-policies will have different security implications. A service provider might only enforce P1, indicating that everything up to the OS kernel level as well as all applications running with super-user privilege must be measured and verified. Conforming to this meta-policy, the attestation result will guarantee that the user's choice of platform configuration, up to the OS kernel level, is integrity protected. Moreover, it will show that all privileged applications are integrity protected. Such a result, however, does not give much indication as to how secure and up to date the platform configuration is. For instance, a user might decide to miss a few security-critical updates on the OS. As long as the integrity of this old version of OS is verified, the PTS will report a 'PCR values OK' response to the service provider. In consequence, the user is never locked into a particular platform configuration.

Meta-policies like P6 and P7 can be enforced in addition to P1 to provide more security assurance of the user platform. For instance, P6 may indicate that an anti-virus software and firewall must be loaded during boot time, and P7 might state that the automatic-update feature must be enabled on both the OS and anti-virus software. These extra policies would provide much more security assurance of the user platform than merely using P1, but may require the user to install new pieces of software and change settings. Alternatively,

enforcing P2 would allow the service provider to rely on the trusted third parties to correctly verify the user's security policy and essential security properties that must be met.

Some service providers, that operate in a more controlled user environment, may feel the need to enforce P5, publishing a list of trusted OS measurements that the user may choose from. The user would end up installing an OS version that has been listed by the service provider, but would still be able to use their choice of applications from the OS level up. Combining P5 with policies like P6 and P7 can give chain of trust as far as to the browser, ignoring all other software running. Such a combined meta-policy would ensure a highly secure environment on the user platform, but only with some level of platform locking as discussed in Table 1.

5.4 Integration with mobile reference architecture

Our system and the mobile reference architecture [3] share some similarities in distributing the signed RIMs through trusted authorities. What is missing in the mobile architecture is a trusted reporting component, the late-launched PTS in our model, that would report the local secure boot results to relying parties in a trustworthy manner. Hence, we believe the ideas for late launching the PTS – perhaps after going through some generalisation – could be applied to the mobile architecture, enabling high assurance mobile services without placing any heavy burden on the service providers.

6. RELATED WORK

Sadeghi and Christian Stübke [9] have proposed Property-Based Attestation (PBA), which provides a level of indication between integrity reporting and the evaluation of a platform's trustworthiness so that *properties* are attested rather than binary hashes. There are a range of implementation options, one of which is to have a trusted third party provide property certificates linking reference integrity measurements to properties [9]. PBA can be implemented in our system by having meta-policies signed by a trusted party which specify certain property to integrity measurement mappings, and then having the PTS locally check for this property. PBA, however, cannot support many of our meta-policies, including P1 and P4, and does not allow the user to specify their own security-policy (i.e. the whitelist). Furthermore, our approach integrates the PTS which can look at policies referring to fine-grained platform properties, as described in P6 and P7.

Local verification of trusted computing platforms has been proposed by Ali et al. [4]. Their local verification mechanism records system calls to identify with a high probability whether a runtime attack has occurred. However, this also requires users to agree in advance on their software configurations and so the problems that our approach attempts to solve are quite different. Seshadri et al. [11] have proposed SecVisor, which is a small hypervisor that is late-launched to protect the code integrity of kernels and ensure that only approved code can execute in kernel mode. The notion of relying on a late-launched component to perform a set of trusted operations locally runs commonly through SecVisor and our solution; but the set of operations that are managed by our PTS and SecVisor are, again, quite different and designed based on different security requirements.

Policy	Security properties	Whitelist management	Platform locking
P1	It may indicate that the user is only running software from a whitelist of a size below certain number.	The user manages one security policy that includes a software whitelist; the service provider does not manage any whitelist.	No platform locking issue.
P2	Ensures that the user's security policy that represents the user's platform configuration is verified and signed by one or more trusted authorities.	The user manages one software whitelist; the service provider does not manage any whitelist; each trusted authority manages some portion of the RIMs.	No platform locking issue.
P3	Ensures that the user's RIMs are up to date.	The user manages one software whitelist; the service provider does not manage any whitelist.	No platform locking issue.
P4	Ensures that the user-defined whitelist has not been tampered with.	The user manages one software whitelist; the service provider does not manage any whitelist.	No platform locking issue.
P5	Trusted kernel can be loaded.	The user manages one software whitelist excluding the OS kernel measurements; the service provider manages the trusted OS kernel measurements.	The user may be locked into certain OS trusted by the service provider.
P6	The use of certain types of security-critical applications (e.g. anti-virus software) can be ensured.	The user manages one software whitelist; the service provider does not manage any whitelist.	Certain types of applications can be mandated.
P7	Important security settings/configurations can be ensured, e.g. enabling automatic-update on the OS.	The user manages one software whitelist; the service provider does not manage any whitelist.	Certain software configurations can be mandated.

Table 1: Evaluation of whitelist management and platform locking problems

7. CONCLUSIONS

Attestation does not work well in large user-base systems due to the associated whitelist management costs and the danger of locking users into certain platform configurations. We proposed an indirect attestation system that spreads the whitelist management overheads – which would otherwise all fall on the service provider – among the users, software vendors and security companies (trusted third parties). Each user defines their own acceptable platform configuration in terms of the reference integrity measurements. Upon attestation, a late-launched Platform Trust Service checks whether the current platform configuration matches these reference values, and reports this local verification result to the service provider. Additionally, the Platform Trust Service checks that the user's local whitelist conforms with high-level meta-policies defined by the service provider. Configuring the meta-policies allows the service provider to select a suitable indirection attestation paradigm to be used. The service provider merely examines the reported verification result and conformance with its meta-policies in order to authenticate the user platform. Instead of obliging the user to configure their platform in one particular way and checking whether they have, our system verifies whether the integrity of the user's *choice of configuration* has been maintained.

The proposed attestation system explores one suitable compromise between the whitelist management overheads, usability and security. We hope that our work would provide some initiatives for further discussion and investigation of similar ideas for making attestation more manageable in large user-base systems.

8. REFERENCES

- [1] TCG Infrastructure Working Group Platform Trust Services Interface Specification (IF-PTS). Specification version 1.0, November 2006.
- [2] Trusted computing group background.

<https://www.trustedcomputinggroup.org/about/>, October 2006.

- [3] TCG Mobile Reference Architecture. Specification version 1.0, June 2007.
- [4] T. Ali, M. Nauman, and X. Zhang. On leveraging stochastic models for remote attestation. In *INTRUST 2010: Proceedings of the 2nd International Conference on Trusted Systems*, 2010.
- [5] E. Bangerter, M. Djakov, and A.-R. Sadeghi. A Demonstrative Ad Hoc Attestation System. In V. R. Tzong-Chen Wu, Chin-Laung Lei and D.-T. Lee, editors, *ISC '08: Proceedings of the 11th International Conference on Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 17–30, Taipei, Taiwan, September 2008. Springer.
- [6] D. Grawrock. *Dynamics of a Trusted Platform*. Intel Press, February 2009.
- [7] H. Kim, J. H. Huh, and R. Anderson. On the Security of Internet Banking in South Korea. Technical Report RR-10-01, OUCL, March 2010.
- [8] A. Lee-Thorp. Attestation in Trusted Computing: Challenges and Potential Solutions. Technical report, Royal Holloway, 2010.
- [9] A.-R. Sadeghi and C. Stübke. Property-based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms. In *NSPW '04: Proceedings of the 2004 Workshop on New Security Paradigms*, pages 67–77, New York, NY, USA, 2004. ACM.
- [10] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *USENIX Security Symposium*, volume 13, pages 223–238. USENIX Association, 2004.
- [11] A. Seshadri, M. Luk, N. Qu, and A. Perrig. SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 335–350, New York, NY, USA, 2007. ACM.

EXHIBIT I



Books

Trusted Systems : Second International Conference, INTRUST 2010, Beijing, China, December 13-15, 2010, Revised Selected Papers

AVAILABLE AS

Online

SUMMARY

This book constitutes the thoroughly refereed post-conference proceedings of the International Conference on Trusted Systems, INTRUST 2010, held in Beijing, China, in December 2010. The 23 revised ...

[Show More](#)

[View Online](#)

[♥ Save to Lists](#)

[” Citation](#)

Details

Publication Details

CREATOR

edited by Liqun Chen, Moti Yung

FORMAT

Books

LANGUAGE

English

CONTRIBUTORS

[Chen, Liqun.](#) editor

[Yung, Moti.](#) editor

PUBLICATION

1st ed. 2011

Berlin, Heidelberg : Springer Berlin Heidelberg : Imprint: Springer, 2011

SERIES

[Security and Cryptology ; 6802](#)

PHYSICAL DETAILS

1 online resource (XIV, 362 p. 73 illus.)

ISBNs

9783642252822, 9783642252839, 3642252834, 3642252826

OCLC

ocn765365879

Notes

Includes index.

English

Contents

Intro -- Title -- Preface -- Table of Contents -- Implementation Technology -- Seamless Integration of Trusted Computing into Standard Cryptographic Frameworks -- Introduction -- Related Work -- Our Contribution and Background -- Architecture Overview -- Comparison to Classic TCG Trusted Software Stacks -- Connection and Context Setup -- Using the doTSS Client API -- System Implementation -- Server Architecture -- Client Architecture -- Client/Server Communication Protocol -- doTSS on Embedded Systems -- Conclusion and Future Work -- References -- Design and Implementation of Document Access Control Model Based on Role and Security Policy -- Introduction -- Background and Motivation -- Previous Works -- Challenging Issues -- Our Contributions -- XACML-RBAC Framework -- Document Access Control Model -- Security Requirements of Model -- Security Policy of the Model -- Static Security Rules -- Dynamic Security Rules -- Information Flow Analysis -- Document Access Control System -- Implementation -- Evaluation -- Conclusion and Future Work -- References -- Towards High-Performance IPsec on Cavium OCTEON Platform -- Introduction -- Background -- Cavium OCTEON -- Run-to-Completion vs Pipeline -- IPsec -- Encrypting Algorithms -- IPsec Implementation -- OUTBOUND Design -- INBOUND Design -- Performance Evaluation -- Development Platform and Test Environments -- Experiment on Different Encryption Algorithms and Packet Length -- Experiment on Different Core Numbers -- Experiment on Pipeline and Run-to-Completion Mechanism -- Conclusion -- References -- Security Analysis -- An Abstract Model of a Trusted Platform -- Introduction -- Trusted Computing -- Trusted Platforms -- The Need for Reasoning about Platform Properties -- A CSP Primer -- Why CSP -- An Abstract Model -- Runtime System -- Resources -- Applications -- Platform Configurations -- Platform Instantiation and Property Verification -- Initial Memory Contents -- Component Location Configuration -- Application Logic -- Property Specification and Verification -- Case Study: Cooper's Trusted Grid Platform -- Property Specification -- Common Setup -- Instantiating a Malicious Platform -- Instantiating a Non-malicious Platform -- Property Verification, Results and Discussion -- Performance Results -- Observations -- Application Specification -- Extending Configurations -- State Space Reduction -- Performance and Scalability of the Model -- Related Work -- Conclusion and Future Work -- References -- Modeling TCG-Based Secure Systems with Colored Petri Nets -- Introduction -- Preliminaries -- Trusted Computing -- Colored Petri Nets --

Modeling TCG-Based Secure Systems with CPN -- Basic Model -- Process Management -- Synchronous I/O -- Data Management -- Late Launch -- Case Studies -- Memory Protection in TrustVisor -- Remote Attestation Based on Dynamic Root of Trust -- Related Work -- Secure Systems Based on Trusted Computing -- CPN Based Formal Studies -- Formal Verification on Trusted Computing -- Conclusion -- References -- Information Flow Graph: An Approach to Identifying Covert Storage Channels -- Introduction -- The Information Flow Graph Approach -- Graph Notation and Semantics -- Constructing An Information Flow Graph -- Pruning the Information Flow Graph -- Searching for Information Flow Paths -- Analyzing Operation Sequences -- Example Analyses Using Information Flow Graph -- A Brief Description of the File System Example -- Example Information Flow Graph and Scenario List for File System Example -- Comparison of IFG, CFT and SRM Analyses for the File System Example -- Comparison among SRM, CFT and IFG -- Conclusion and Future Work -- References -- Trusted Subjects Configuration Based on TE Model in MLS Systems

Introduction -- Related Works -- Trusted Subjects Configuration -- Notation for Trusted Subject -- Information Flow Graph -- MLS Policy Information Flow Graph -- Type Enforcement Policy and Entity Information Flow Graph -- Determine the Trusted Subjects -- Security Label Range of Trusted Subjects -- Risk Information Flow Graph -- Risk of Trusted Subjects -- Risk Security -- The Assurance Requirements of Trusted Subjects -- Conclusions -- References -- Cryptographic Aspects (I) -- Key Exchange with Anonymous Authentication Using DAA-SIGMA Protocol -- Introduction -- Our Contribution -- Related Work -- Organization of the Paper -- Background and Building Blocks -- Notations -- Review of DAA -- Review of SIGMA Key Exchange Protocol -- Security Model -- Review of Canetti-Krawczyk Key Exchange Model -- Security Model for Key Exchange with Anonymous Authentication -- Proposed DAA-SIGMA Protocol -- Security Proof of the DAA-SIGMA Protocol -- Proof of Property P1 -- Proof of Property P2 -- Discussions and Extensions -- Using Group Signatures Instead of DAA -- Use Certificates Instead of Identities -- Supporting User-Controlled-Traceability and DAA Revocation -- Mutual Anonymous Authentication Using DAA -- Conclusions and Future Work -- References -- Revocation of Direct Anonymous Attestation -- Introduction -- General Concept of DAA Revocation -- Two Approaches -- Security Properties -- Review Three Types of DAA Schemes -- The RSA-DAA Scheme -- The LRSW-DAA Schemes -- The SDH-DAA Schemes -- Review of the Existing Revocation Solutions -- Rekey-Based Revocation -- An Overview -- Rekey

in the RSA-DAA Scheme -- Rekey in the LRSW-DAA Schemes -- Rekey in the SDH-DAA Schemes -- Verifier-Local Revocation -- An Overview -- New VLR in the RSA-DAA Scheme -- New VLR in the SDH-DAA Schemes -- Comparisons and Discussions -- Performance Measurement

Comparisons of VLR Methods -- Comparisons between Rekey and VLR Methods -- Using Rekey and VLR Approaches Together -- Conclusions -- References -- Measuring Random Tests by Conditional Entropy and Optimal Execution Order -

- Introduction -- Preliminaries -- Principle of a Statistical Test -- NIST Random Test Package -- The Conditional Entropy of the Tests -- The Application of the Entropy -- The Result and Analysis -- The Order of the Tests' Execution -- The Way to Decide the Execution Order -- The Result and Analysis -- Conclusion -- References -- Cryptographic Aspects (II) -- Leakage Resilient Strong Key-Insulated Signatures in Public Channel -- Introduction -- Our Contribution -- Definitions -- Some Impossibility Results -- Our Proposal -- A Concrete Instantiation -- References -- Two-Head Dragon Protocol: Preventing Cloning of Signature Keys -- Introduction -- Previous Work -- Preliminaries -- A Simple Solution -- Target Application Area -- Two-Head Dragon Protocol -- Preliminaries -- Card Deployment -- Signature Creation and Verification -- Situation of an Adversary -- Security Considerations -- Remarks on Cloning Assumptions -- Composition Issues -- Final Remarks -- Further Applications -- Conclusion -- References -- Addressing Leakage of Re-encryption Key in Proxy Re-encryption Using Trusted Computing -- Introduction -- Preliminaries -- Model of Unidirectional Proxy Re-encryption -- Review of Chow et al.'s Scheme -- Our Method -- Overview of the Method -- Modifying Chow et al.'s Scheme -- Details of Our Method -- Security Analysis -- Discussions -- Conclusion -- References -- Mobile Trusted Systems -- Can Hand-Held Computers Still Be Better Smart Cards? -- Introduction -- System Requirements and Use Cases -- Background -- Standards in the Field -- Trusted Hardware -- Architecture

Host Computer Smart Card Middlewares and Smart Card Filesystem -- Smart Card Provisioning Server -- Credential Life-Cycle -- Implementation -- Speed Measurements -- Security Analysis -- Related Work -- Conclusions and Future Work -- References -- TruWalletM: Secure Web Authentication on Mobile Platforms -- Introduction -- Model and Requirement Analysis -- Architecture and Design Decisions -- Design Decisions -- Architecture -- Use Cases and Protocols -

- Establishing SSL/TLS Connection and Managing Two Logical SSL/TLS Subchannels -- TruWalletM Initialization -- Registration -- Login -- Password Change -- Implementation -- Security Considerations -- Related Work --

Conclusion and Future Work -- References -- A Game Theory-Based Surveillance Mechanism against Suspicious Insiders in MANETs -- Introduction -- Background and Motivation -- Related Works -- Challenging Issues -- Our Contributions -- Threat Analysis -- Reputation Mechanism -- Misbehavior Prediction -- Surveillance Game -- Multilevel Surveillance -- Reward and Punishment Scheme -- Surveillance Game Model -- Game Solutions -- Decision Making -- Case Study -- Conclusion -- References -- Hardware Security -- Hardware Trojans for Inducing or Amplifying Side-Channel Leakage of Cryptographic Software -- Introduction -- Preliminaries and Related Work -- Bug Attacks -- Micro-Architectural Side-Channels -- Fault Attacks -- Activation Mechanisms -- Method 1: Snooping the Data Bus -- Method 2: Snooping Operands of Instructions -- Effects of the Trojan -- Fault Induction -- Timing Variation -- Variation of Power Consumption -- Case Studies -- Attacking AES -- Attacking RSA -- Conclusions -- References -- An Emerging Threat: Eve Meets a Robot -- Introduction -- What is a Robot? -- Security of Robotics -- A Simple Active Attack and a Countermeasure -- Active Attacks on Robots -- Information Leakage Attacks on Robots

Subjects

[Computer networks.](#)

[Cryptography.](#)

[Data encryption \(Computer science\).](#)

[Electronic data processing—Management.](#)

[Algorithms.](#)

[Computers and civilization.](#)

[Data protection.](#)

Items Related By Call Number

[Shelf Browse](#)



Books

Interact and engage! : 50+ activities for virtual...

LaBorie, Kassy, author

Online

HF5734.7

Sound Recordings

Running virtual meetings : test your technology, ...

Online

HF5734.7



Additional Information

Information from the Web

Library Staff Details

MARC Bibliographic Record

LEADER 02545cam a2200517Ka 4500

001 9988737113602122

005 20230118174827.0

006 m d

007 cr cnu---unuuu

008 111201t20112011gw ob 101 0 eng d

- 016 7_ \$a015975113\$2Uk
- 020 \$a9783642252839 (electronic bk.)
- 020 \$a3642252834 (electronic bk.)
- 020 \$z9783642252822
- 035 \$a(0CoLC)ocn765365879
- 035 \$a(WU)8873711-uwmadisondb
- 035 \$a(EXLNZ-01UWI_NETWORK)9910112054302121
- 040 \$aGW5XE\$beng\$cGW5XE\$dUKMGB\$dGZM
- 049 \$aGZMA
- 050 _4 \$aQA76.9.A25\$bI58 2011
- 111 2_ \$aINTRUST (Conference)\$n(2nd :\$d2010 :\$cBeijing, China)
- 245 10 \$aTrusted systems :\$bsecond international conference, INTRUST 2010, Beijing, China, December 13-15, 2010, Revised selected papers /\$cedited by Liqun Chen, Moti Yung.
- 264 _1 \$aBerlin ;\$aNew York :\$bSpringer,\$c[2011]
- 264 _4 \$c©2011
- 300 \$a1 online resource (xiii, 362 pages).
- 336 \$atext\$btxt\$2rdacontent

- 337 \$acomputer\$bc\$2rdamedia
- 338 \$aonline resource\$bcr\$2rdacarrier
- 490 1_ \$aLecture notes in computer science,\$x0302-9743 ;\$v6802
- 490 1_ \$aLNCS sublibrary. SL 4, Security and cryptology
- 504 \$aIncludes bibliographical references and author index.
- 520 8_ \$aAnnotation This book constitutes the proceedings of the International Conference on Trusted Systems, held in Beijing, China, in December 2010.The 23 contributed papers presented together with nine invited talks from a workshop, titled "Asian Lounge on Trust, Security and Privacy" were carefully selected from 66 submissions. The papers are organized in seven topical sections on implementation technology, security analysis, cryptographic aspects, mobile trusted systems, hardware security, attestation, and software protection
- 650 _0 \$aComputer security\$vCongresses.
- 650 _0 \$aComputer networks\$xSecurity measures\$vCongresses.
- 650 _0 \$aData encryption (Computer science)\$vCongresses.
- 653 _4 \$aComputer science

- 653 _4 \$aComputer Communication Networks
- 653 _4 \$aData Protection
- 700 1_ \$aChen, Liqun.
- 700 1_ \$aYung, Moti.
- 730 0_ \$aSpringer Computer Science e-Book Collection.
- 830 _0 \$aLecture notes in computer science ;\$v6802.
- 830 _0 \$aLNCS sublibrary.\$nSL 4,\$pSecurity and cryptology.
- 856 40 \$uhttp://dx.doi.org/10.1007/978-3-642-25283-9\$zAvailable through Springer Computer Science e-Book Collection.
- 949 \$a20111220\$bjmv\$cc\$dnt\$ee\$fcall:n\$grepl:n\$hgls
- 994 \$aC0\$bGZM
- 997 \$aMARCIVE

MMS IDs**Document ID:** 9911119338002121**Network Electronic IDs:** 9911119338002121, 9910112054302121**Network Physical IDs:****mms_plt_ids:** 991013495631902128, 993871473702128**mms_st_ids:** 991013957985202131**mms_mad_ids:** 9988737113602122