High-speed Encryption and Authentication

John Viega
viega@securesoftware.com

Requirements

- Assume key exchange happened securely
- **Message secrecy:** What the attacker sees reveals no new information about messages, even if attacker can control some messages
- **Message integrity:** The recipient can detect whether the message is in its intended form, or whether there has been tampering

- MIC = Message Integrity Code (aka MAC)
- In reality, absolute assurance is not practical
- Integrity is more often important than secrecy

SecureSoftware™
eliminating vulnerabilities at the source

More potential requirements

- **Efficient in software**
  - Not hard
- **High speeds in hardware**
  - 10 Gigabits +
  - Lowest cost best
  - Requires parallelizability / pipelinability
- **High assurance**
  - Provable security
  - Minimal assumptions
- **Fast setup**
- **Ability to check integrity of plaintext headers**

SecureSoftware™
eliminating vulnerabilities at the source

A non-option

- **"Encryption with redundancy"**
- **Depends on the redundancy function, but...**
- **Usually doesn't work**
- **Attacks against many proposed schemes**
  - XOR message blocks
  - XOR ciphertext blocks
  - Kerberos PCBC mode
- **Minimal redundancy: a secure keyed MIC**

SecureSoftware™
eliminating vulnerabilities at the source

Composition Approaches

- **Combine encryption and integrity schemes**
- **Select a suitable encryption mode and MIC**
- **Example: SSL/TLS**
  - Block ciphers run in CBC mode or RC4
  - HMAC-SHA1 or HMAC-MD5
- **How to combine primitives?**
  - Should be easy, but it isn't!
- **Three paradigms**
  - MAC-then-encrypt
  - Encrypt-then-MAC
  - Encrypt-and-MAC
- **OpenSSL CBC ciphersuites had a timing attack**

SecureSoftware™
eliminating vulnerabilities at the source

Generic Composition: Cipher modes

Mode	*Requirements*	*Precomputable*	*Parallelizable*
CBC	Random IV	✘	✘
CTR	Unique nonce	✓	✓
OFB	Unique nonce	✓	✘

Nonces

- **Data that is unique per-message**
- **Repeats must occur with very low probability**
- **Common contents**
  - Message counter
  - Session ID
  - Info uniquely identifying client/sender
  - Random value
- **Nonce bits can be valuable!**
- **Easy + good to throw in all possible distinguishers**

SecureSoftware™
eliminating vulnerabilities at the source

Generic Composition: MACs

MAC	*Parallelizable*	*Hardware suitable*	*Patent free*
HMAC	✘	✘ (Not high speed)	✓
CBC-MAC	✘	✘ (Not high speed)	✓
UMAC	✓	✘ (Too complex)	✓
XOR-MAC	✓	✓	✘

SecureSoftware™
eliminating vulnerabilities at the source

MAC Algorithms

- **HMAC: choose a cryptographic hash function**
  - SHA1 or MD5
  - MD5 is low assurance in many respects
  - Security proof assumptions are "weak"
- **XOR-MAC: choose hash or cipher**
  - Security proof assumptions are strong
  - Hash function will generally be more efficient
  - Block ciphers are fast enough
  - Single primitive means fewer assumptions
  - A bit slow in software, but okay
- **Crypto community focuses on block ciphers**
  - AES much higher assurance than SHA1
- **Only appropriate combo: CTR + XOR-MAC**

SecureSoftware™
eliminating vulnerabilities at the source

Authenticated Encryption Schemes

- **Single primitive for encryption and integrity**
  - One key (may turn into multiple keys internally)
  - Good provable security
  - Built upon a single cryptographic assumption
- **OCB: Phil Rogaway et al.**
  - Great in software
  - Very good in hardware
  - Patented
- **CCM: Whiting, Housley, Fergusen**
- **EAX: Bellare, Rogaway, Wagner**
  - Not appropriate for high-speed environments
  - We'll ignore these two
  - Though, CCM is a FIPS standard

SecureSoftware™
eliminating vulnerabilities at the source

More Authenticated Encryption Schemes

- **CWC: Kohno, Viega, Whiting**
  - Combines a "universal hash" with AES-CTR
  - Universal hash is built on multiplying 127-bit values
  - Great on 64-bit platforms
  - Good in hardware and 32-bit platforms
  - Bad on 16-bit and 8-bit platforms
- **GCM: McGrew, Viega**
  - Also based on universal hash plus AES-CTR
  - Hash relies on GF(2^{128}) multiplies
  - Multiplies implemented with XORs
  - Great in hardware
  - Good in software (8K key-dependent tables)
  - Minor refinements in the next 30 days

SecureSoftware™
eliminating vulnerabilities at the source

Feature Comparison

	OCB	*CWC*	*GCM*	*CTR + XOR-MAC*
Software	Best	32/64	Precomp	Good
Hardware	Excellent	Okay	Best	Excellent
Keying	1 Key	Subkeys	1 Key†	2 Keys*
Patent-Free	✘	✓	✓	✘
Nonce	16 bytes	12 bytes	Any	< 16 bytes
Associated Data	✘*	✓	✓	✓**

SecureSoftware™
eliminating vulnerabilities at the source

Questions?

http://www.zork.org/gcm/
http://www.zork.org/cwc/
http://www.secureprogramming.com/

viega@securesoftware.com