Cryprography and Network Security, PART III: Authenticated Encryption

Timo Karvi

11.2013

Cryptographical Protocols

- We have seen now encryption primitives and key agreement methods and protocols. When a protocol sets up a connection, the key agreement is usually the first task. This involves authentication, too.
- After the first phase, data transfer can begin. If encryption is used, it must be used with chaining methods as we have seen. This, however, does not protect against modification of packets.
- If data is binary, it may be quite difficult to detect, if packets have been tampered. That is why integrity checks are needed in addition to encryption.
- Moreover, in many cases it is advisable to use authentication as well. In this part we go through theory and methods to add integrity checks and authentication to encryption.
- First we present some attacks against CBC method.

Attack against IPsec using only CBC I

- Consider a situation, where a user sends packets through a router to a private network. There is an IPsec connection between an outsider and the router, but packets are delivered in plain text inside the private network. Assume that an encrypted packet is meant to port 80. Furthermore, there is an insider with port 25 and this insider tries to read the packet meant for port 80.
- The first encrypted packet consists of the port number, in this case 80, and data. These both fields are encrypted in the first block using CBC. Thus

$$c[0] = E(k, m[0] \oplus IV), \quad m[0] = D(k, c[0]) \oplus IV.$$

Attack against IPsec using only CBC II

- Now the owner of port 25 can redirect this packet to himself and in decrypted form. Namely, he can try to change the IV in such a way that the destination port changes from 80 to 25. Thus the new IV should be

$$IV' = IV \oplus (\ldots 80 \ldots) \oplus (\ldots 25 \ldots)$$

- In this attack, the attacker knows the original IV and the destination port 80, but he does not know the secret encryption key. With these data he is able to modify the encrypted text in a meaningful way.

Attack by an outsider

- Consider now the situation where each keysroke is encrypted with CTR mode. Thus the packet would consist of IP header, TCP header, 16 bit TCP checksum T ,and 1 byte keystroke D, and all is encrypted.
- Suppose that a client sends this kind of messages to a server. The server sends back an acknowledgement, if the checksum is valid, nothing otherwise.
- An attacker can now change checksum and keystroke fields by making xor operations and follow, if acknowledgements are sent. Checksum is calculated from the header and D in plaintext. This reveals D.
- Remember that if we xor an encrypted text with s, the decrypted plain text is also xored with s.

Authenticated Encryption: General Definition

An authenticated encryption system (E, D) is a cipher where

$$\begin{aligned} E &: K \times M \times N \longrightarrow C, & (1) \\ D &: K \times C \times N \longrightarrow M \cup \{\perp\}. & (2) \end{aligned}$$

Here, K is a key space, M a message space, and N is a nonce space. Nonce is in this case an extra field containg integrity and authentication information. The decryption may produce a plain text or a rejection $(\perp)$, if the integrity or authentication is not valid.

The system must provide semantic security under a CPA attack and ciphertext integrity. The latter means that an attacker cannot create new ciphertexts that decrypt properly. This property can be defined formally.

Ciphertext Integrity I

Consider the following game an attacker plays with a challenger.

1. Adversary prepares plaintexts $m_1, m_2, ..., m_q$ and sends them to the challenger.
2. The challenger encrypts the messages with his secret key and sends the ciphertexts $c_1, c_2, ..., c_q$ to the attacker.
3. The attacker prepares a new ciphertext c and sends it to the challenger.
4. The challenger outputs $b = 1$, if $D(k, c) \neq \perp$ (i.e. c is not rejected) and $c \neq c_i$, $i = 1, ..., q$. Otherwise the output is $b = 0$.

Ciphertext Integrity II

Definition

(E, D) *has ciphertext integrity, if for all efficient algorithms A*

$$\mathrm{Adv_{CI}}[A, E] = \mathrm{Prob}[\text{challenger outputs } 1]$$

is negligible. $\square$

Definition

A cipher (E, D) *provides authenticated encryption (AE), if it is*

a) *semantically secure under CPA, and*

b) *has ciphertext interity.*

- CBC with random IV does not provide AE, because $D(k, \cdot)$ never outputs $\perp$, hence an adversary easily wins CI game.
- If (E, D) provides AE and $D(k, c) \neq \perp$, then a receiver knows a message is from someone who knows k. Thus the encryption provides authentication, but the message could be a replay.
- Also, AE provides security against chosen ciphertext attacks which we will see in the next slides.
- It is possible in practice that anadversary can fool a server into decrypting certain ciphertexts and then learn partial information about plaintext. AE wants to prevent thse kind of attacks.

Chosen Ciphertext Security I

- We assume that an adversary can make both CPA and CCA.
- Especially, he can obtain the encryption of arbitrary messages of his choice and
- he can decrypt any ciphertext of his choice, other than challenges.
- An adversary's goal is to break semantic security.

Consider the following game consisting of two kind of experiments, EXP(b), $b = 0, 1$:

1. An adversary prepares two series of plaintexts: $m_{1,0}, m_{2,0}, ..., m_{q,0}$ and $m_{1,1}, m_{2,1}, ..., m_{q,1}$. Assume $|m_{i,0}| = |m_{i,1}|$. He sends the both series to a challenger.
2. The challenger chooses randomly $b = 0$ or $b = 1$, and encrypts all the messages $m_{i,b}$, $i = 1, ..., q$. Then he sends these encrypted messages $c_1, c_2, ..., c_q$ back to the adversary. This part of the game is the adversary's CPA query.

Chosen Ciphertext Security II

3. After this a CCA query starts. The adversary prepares a ciphertext c and sends it to the challenger. It is assumed that

$$c \notin \{c_1, ..., c_q\}$$

.

4. Now the challenger decrypts c and sends it back to the attacker.
5. After this, the adversary tries to guess, if the ciphertexts $c_1, ..., c_q$ are the ciphertexts of the messages $m_{i,0}$ or $m_{i,1}$. So he outputs $b' \in \{0, 1\}$.

Definition

$(E; D)$ is CCA secure, if for all efficient algorithms A,

$$\mathrm{Adv}_{\mathrm{CCA}}[A, E] = |\mathrm{Prob}[EXP(0) = 1] - \mathrm{Prob}[EXP(1) = 1]|$$

is negligible.

Example: CBC is not CCA-secure

Consider a situation where an adversary sends two plaintext messages m_0 and m_1 to be encrypted. A challenger chooses b and sends

$$c = E(k, m_b) = (IV, c[0])$$

back to the adversary. Notice that IV is known to the adversary, because he has to encrypt messages, too. Now he can send $c' = (IV \oplus 1, c[0])$ to the challenger to be decrypted. This is a new ciphertext, because IV has been modified. The challenger then sends back

$$D(k, c') = m_b \oplus 1$$

and the adversary can easily deduce b from the result. So the advantage is even 1.

AE implies CCA security

Theorem

Let (E, D) be a cipher that provides AE. Then (E, D) is CCA secure.

In particular, for any $q - query$ efficient A there exist efficient B_1, B_2 such that

$$\mathrm{Adv}_{\mathrm{CCA}}[A, E] \leq 2q \cdot \mathrm{Adv}_{\mathrm{CI}}[B_1, E] + \mathrm{Adv}_{\mathrm{CPA}}[B_2, E].$$

We skip the proofs. AE ensures confidentiality against an active adversary that can decrypt some ciphertexts. However, AE does not prevent replay attacks nor side channel attacks.

In practice, AE is achieved by using an encryption with chaining and adding an integrity tag to the message with the encrypted data. In the computing of the tag a secret parameter must be used. Typically a tag is computed using a hash function with a secret key or using encryption with chaining in such a way that the result is only one block.

AE: History I

- AE was introduced quite recently, in 2000.
- Before this softwares offered separetely CPA-secure encryption and MAC integrity checks.
- Every project had to combine the two itself without a well-defined goal.
- There were also pitfalls: not all combinations provide AE.
- Consider for example solutions in SSL, IPsec and SSH. Lets denote by k_E an encryption key and by k_I a MAC key. Tag is calculated with some one-way function S.
- SSL had the folllowing procedure to add the MAC tag:

$$msg \;\; m \Longrightarrow msg \;\; m \;\; tag \Longrightarrow E(k_E, m||tag),$$

where $tag = S(k_I, m)$.

AE: History II

- IPsec:

$$msg\ \ m \Longrightarrow E(k_E, m) \Longrightarrow E(k_E, m) \,||\, tag,$$

  where $tag = S(k_I, c)$. This is always correct.

- SSH:

$$msg\ \ m \Longrightarrow E(k_E, m) \Longrightarrow E(k_E, m) \,||\, tag,$$

  where $tag = S(k_i, m)$.

CBC-MAC

Let F be a pseudo random function. That is to say, F takes a message as an argument and produces a fixed sized random string. Typically, F could be an encryption like AES. Let $m_0, m_1, ..., m_n$ be plain text message blocks. Then the integrity tag is calculated as follows:

$$\begin{aligned} tag_0 &= m_0, & (3)\\ tag_i &= F(k, m_i \oplus tag_{i-1}), \quad i = 1, ..., n, & (4)\\ tag &= F(k', tag_n). & (5) \end{aligned}$$

Notice the last step, applying F with a different key. It can be shown that this last step is crucial. Without it the method would be vulnerable.

NMAC

NMAC or nested MAC works as follows:

$$\begin{aligned} tag_0 &= F(k, m_0), && (6)\\ tag_i &= F(tag_{i-1}, m_i), && (7)\\ tag &= F(k', tag_n || padding) && (8) \end{aligned}$$

Notice again the last step without which the method is vulnerable. The padding is necessary, if the block size of messages is larger than the key size. F produces outputs of key size.

Digression: Counter Mode I

Counter mode or CTR is a popular chaining method nowadays, even if it is an old suggestion. Encryption is done in the form

$$C_i = P_i \oplus E(K, L_i),$$

where

- P_i is i'th plaintext,
- L_i the value of the counter on the i'th round,
- K is a secret symmetric key,
- $\oplus$ is a xor operation.

Typically a counter has an initial value which is increased by one for every round. No other chaining is used.
Advantages:

- Efficient at hardware level.

Digression: Counter Mode II

- Efficient also at software level.
- Preprosessing possible (encryption).
- Blocks can be prosessed in random order.
- CTR is as strong as other chaining methods.
- Only encryption operation necessary, no decryption operation needed.

AE Theorems

Let (E, D) be a CPA secure cipher and (S, V) a secure MAC (S is a MAC function, V a verification method). Then it is possible to prove:

Theorem

If encryption is done first and after that MAC is calculated, then this always provides AE. If MAC is done first and after that encryption is done, then this may be insecure against CCA attacks. However, when (E, D) is used in CTR mode or CBC mode with random initial vector, then MAC-then-Encryption provides AE.

Standards I

- Galois/Counter Mode (GCM). AE mode for symmetric key cryptographic block ciphers. NIST standard 2007. Used for example in IEEE802.IAE, IETF IPsec, SSH, TLS.
- Counter mode with CBC-MAC (CCM). CM mode was designed by Russ Housley, Doug Whiting and Niels Ferguson. CCM mode went on to become a mandatory component of the IEEE 802.11i standard.
- EAX mode. EAX mode was submitted on October 3, 2003 to the attention of NIST in order to replace CCM as standard AEAD mode of operation, since CCM mode lacks some desirable attributes of EAX and is more complex.The authors of EAX mode, Mihir Bellare, Phillip Rogaway, and David Wagner placed the work under public domain and have stated that they were unaware of any patents covering this technology. Thus, EAX mode of operation is believed to be free and unencumbered for any use.

Standards II

- A modification of the EAX mode, so called EAX' or EAXprime, is used in the ANSI C12.22 standard for transport of meter-based data over a network. In 2012 Kazuhiko Minematsu, Hiraku Morita and Tetsu Iwata published a flaw in this mode that breaks the security.

Performance

Cipher	**Code size**	**Speed** MB/s
AES/GCM	large	108
AES/CCM	smaller	61
AES/EAX	smaller	61

Example: TLS

- Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols which are designed to provide communication security over the Internet. They use X.509 certificates and hence asymmetric cryptography to assure the counterparty whom they are talking with, and to exchange a symmetric key. This session key is then used to encrypt data flowing between the parties.
- This allows for data/message confidentiality, and message authentication codes for message integrity and as a by-product message authentication. Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). An important property in this context is *forward secrecy*, so the short term session key cannot be derived from the long term asymmetric secret key.

- TLS is quite an extensive general-purpose cryptographic protocol. Here we concentrate only on its message authentication part. In order to explain this, we need details of some features.
  - TLS uses unidirectional keys. We denote these k_{bs} (from Browser to Server) and k_{sb} (from Server to Browser).
  - k_{bs} consists of two parts: k_{mac} for integrity and k_e for encryption.
  - Each side maintains two 64-bit counters: ctr_{bs}, ctr_{sb}.
  - These are initialized to 0 when session started. The purpose of these counters is to protect against replays.
- Browser side encryption proceeds as follows:

  step 1: $tag := S\,(k_{mac},\ [++\,ctr_{bs} \,||\, \text{header} \,||\, \text{data}])$.
  step 2: pad [header || data || tag] to AES block size.
  step 3: CBC encrypt with k_e and new random IV.
  step 4: prepend header.

Server side decryption proceeds as follows:

step 1: CBC decrypt record using k_e.

step 2: Check pad format: send `bad_record_mac` if invalid.

step 3: Check tag on $[++ctr_{bs} \,||\, \text{header} \,||\, \text{data}$. Reject if invalid.

step 4: prepend header.

The above provides autenticated encryption, provided no other information is leaked during the decryption process.

However: As a consequence of choosing X.509 certificates, certificate authorities and a public key infrastructure are necessary to verify the relation between a certificate and its owner, as well as to generate, sign, and administer the validity of certificates. While this can be more beneficial than verifying the identities via a web of trust, the 2013 mass surveillance disclosures made it more widely known that certificate authorities are a weak point from a security standpoint, allowing man-in-the-middle attack

Bug in older versions

- IV for CBC was predictable. Then IV for next record is the last ciphertext block of current record. Method was no more CPA secure. A practical attack was so called BEAST attack.
- TLS acted as a padding oracle: During the decryption, if pad is invalid, *decryption failed* alert was sent. If mac was invalid, `bad_record_mac` alert was sent. So an attacker learns information about the plaintext.
- Lesson: When decryption fails, do not explain why.

Leaking the length

- The TLS header leaks the length of the TLS records. Lengths can also be inferred by observing network traffic.
- For many web applications, leaking lengths reveals sensitive info:
  - In tax preparation sites, lengths indicate the type of return being filed which leaks information about the user's income.
  - In healthcare sites, lengths leaks what page the user is viewing.
  - In Google maps, lengths leaks the location being requested.
- There are no easy solutions to this problem.