CSA E0 235: Cryptography 07/02/2016

Scribe for Lecture 8

Instructor: Arpita Patra *Submitted by: Kaushik Joarder*

Contents

1 Chosen Ciphertext Attack (CCA)

Chosen-Ciphertext Attack (CCA) is the type of attack where the adversary is provided not only with the ability to encrypt messages of its choice (as in a chosen-plaintext attack or CPA), but also with the ability to decrypt ciphertexts of its choice (with some restrictions). Formally, the adversary is given access to a decryption oracle (DO) in addition to an encryption oracle (EO).

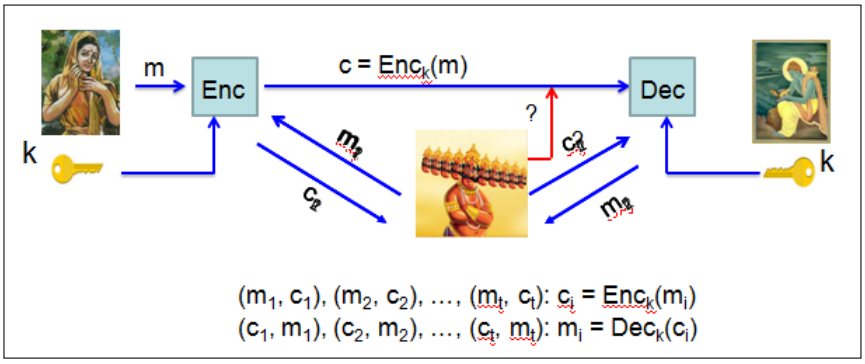


Figure 1: Diagramatic representation of CCA

2 DO practical example

In practical situation an adversary might be able to influence what gets decrypted, and learns some partial information about the result. Here are few examples:

1. It is conceivable that the US cryptanalysts might also have tried to send encrypted messages to the Japanese and then monitor their behavior. Such behavior (e.g., movement of forces and the like) could have provided important information about the underlying plaintext.
2. A user communicating with their bank, where all communication is encrypted. If this communication is not authenticated, then an adversary may be able to send certain ciphertexts on behalf of the user; the bank will decrypt these ciphertexts, and the adversary may learn something about the result. For example, if a ciphertext corresponds to an ill-formed plaintext, the adversary may be able to deduce this from the bank's reaction (i.e., the pattern of subsequent communication).
3. Encryption is often used in higher-level protocols; e.g., an encryption scheme might be used as part of an authentication protocol where one party sends a ciphertext to the other, who decrypts it and returns the result. In this case, one of the honest parties may act exactly like a decryption oracle.

3 A little help from DO can be very very detrimental

Availability of Decryption Oracle (DO) service to the Attacker makes him very powerful. Even the knowledge of whether a modified ciphertext decrypted correctly or not can help an attacker to completely find the underlying plaintext. For example, An Adversary can get complete knowledge of the plaintext encrypted in CBC mode using ***padding oracle attack***.

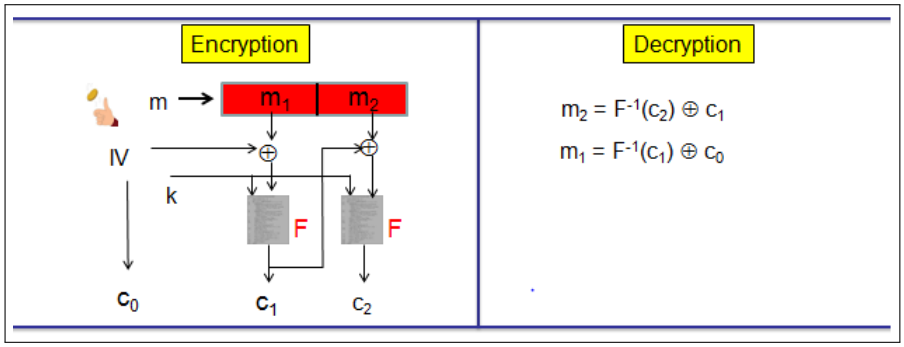
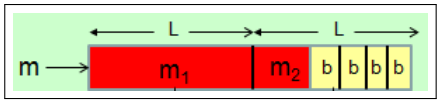


Figure 2: Padding Oracle Attack on CBC Mode

In popular padding system like PKCS#5 padding some extra bytes need to be appended in the last block of **m** to make its length **L** bytes. This procedure is called padding. Let **b** be the number of bytes need to be appended where $1 \leq \mathbf{b} \leq \mathbf{L}$. So each of this **b** bytes are filled with the integer value **b** before encryption. After decryption the receiver reads the final byte value **b**. If the last **b** bytes of all have value **b** then strip-off the pad and output **m**. Else output bad padding (request for re-transmission).



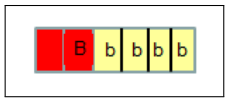
An attacker can modify the ciphertexts and learn **b** (|**m**| leaked) and subsequently the entire plaintext (**m**) from only the knowledge of successful/failure decryption by DO.

To know the value of b and therefore |m|:

The Attacker changes the 1st byte of c_1 and wait for the response of the receiver. The change in the 1st byte of c_1 also creates a change in the 1st byte of the last block in decrypted message. If **b=L**, the first byte of the last block which should be **b** is now changed due to change in c_1. So, It will show failure and the Attacker will gain the information **b=L**. For **b<L**, It will show success. Then the Attacker will repeat changing the 2nd byte, 3rd byte, and so on until the receiver show the result failure. If i is the least indexed modified

ciphertext corresponding to which Failure comes for then **b=L-i+1**

To know the encrypted message:



After having the value **b**, the Attacker tries to know the value of last byte in m_2. The Attacker changes Last **b+1** bytes of c_1 by $\triangle_1$.
$\triangle_1$ = (000...1 (b+1)⊕b (b+1)⊕b (b+1)⊕b
If **b=B**, then the Receiver will show success, otherwise failure.In that case the Attacker will change the value of $\triangle_1$ from 1 to 2,3,4...until it shows success. The Attacker need to run at most 256 times to know **B** exactly.

Morale: Attacker can have control over what is decrypted which can help the attacker to break the secrecy. So it is necessary to capture CCA in the security definition.

4 Comparison of security for SKE between CCA and CPA

The Threat and Break model for both CPA and CCA are quite similar.

Threat: Randomized, probabilistic polynomial-time (PPT) adversaries with capability of Chosen Ciphertext Attack (CCA)
Break: Given the knowledge of two messages (vector of messages), it cannot be distinguished if the ciphertext corresponds to the first or second message (message vector).

CCA Security is Stronger Than CPA-security. It comes automatically from the fact that in CCA, Decryption Oracle service is provided to the adversary (with some obvious restriction) in addition to Encryption Oracle service (which is also available to the Adversary in CPA). So, the protocol which is CCA secure (even after the Adversary has so power!!) is also CPA secure.
As an example it is possible to show that a CPA secure scheme can be easily broken by an attacker with a little help from DO.

Indistinguishability Experiment

An Attacker running the indistinguishability experiment can choose m_0 = (000..) and m_1 = (111..). Then, upon receiving a ciphertext **c** = (**r**, **s**), the adversary A can flip the

first bit of **s** and ask for a decryption of the resulting ciphertext $\mathbf{c}^*$. Since $\mathbf{c}^* \neq \mathbf{c}$, this query is allowed, and the decryption oracle answers with either (10000...) (in which case it is clear that $\mathbf{b} = 0$) or (01111..)(in which case $\mathbf{b} = 1$).
This example demonstrates why CCA security is so stringent.

5 CCA indistinguishability experiment

The Adversary(**A**) is given Oracle access to both Encryption and Decryption Oracle

Pre challenge Training Phase:

A adaptively submits its queries (any query is allowed in any order)to the verifier and receive ciphertexts as response.

Challenge:

A submits two equal length challenge plaintexts m_1,m_2. **A** is free to submit any message of its choice (including the ones already queried during the training phase). A random bit b $\in$ 0, 1 is chosen by the verifier, and then a ciphertext $\mathbf{c} \leftarrow Enc_k(m_b)$ is computed and given to **A**. We call **c** the challenge ciphertext.

Post Challenge Training Phase:

The adversary **A** continues to have oracle access to Enc_k and Dec_k, but is not allowed to query the latter on the challenge ciphertext itself.

Response Phase:

A finally submits its guess regarding encrypted challenge plain-text. **A** outputs a bit $\mathbf{b}^*$. The output of the experiment is defined to be 1(**A** Wins) if $\mathbf{b}^* = \mathbf{b}$, and 0 otherwise.

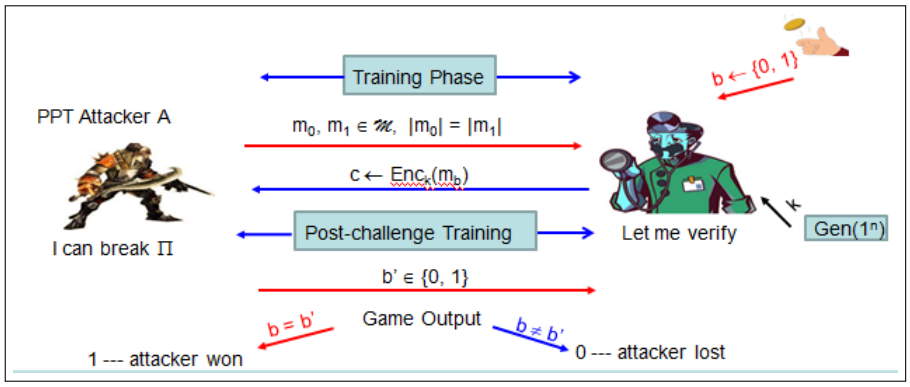


Figure 3: CCA Indistinguishability Experiment

SECURITY DEFINITION

A private-key encryption scheme Π has indistinguishable encryptions under a chosen-ciphertext attack (or is CCA-secure) if for all probabilistic polynomial-time(PPA) adversaries **A** there exists a negligible function negl such that:

$$\Pr[\ PrivK^{cca}_{A,\Pi}\ (\mathrm{n}) = 1] \leqslant \frac{1}{2} + \mathrm{negl}(\mathrm{n})$$

6 CCA Multiple Encryption

For multiple encryption CCA indistinguishability experiment is similar to the previous one except the fact that multiple messages are being encrypted and decrypted.

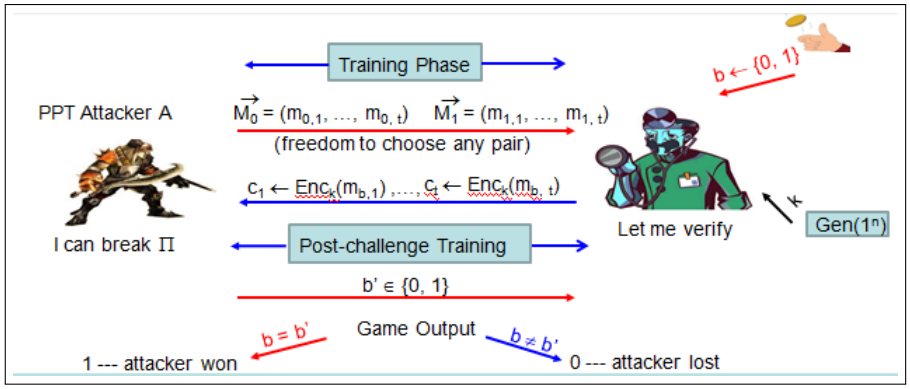


Figure 4: CCA-Mult Indistinguishability Experiment

SECURITY DEFINITION

Π is CCA-secure for multiple encryptions if for every PPT **A**, there is a negligible function negl, such that:

$$\Pr[\ PrivK^{cca-mult}_{A,\Pi}\ (\mathrm{n}) = 1] \leqslant \frac{1}{2} + \mathrm{negl}(\mathrm{n})$$

Any cipher that is CCA-secure for multiple encryption is also CCA-secure for single encryption. The converse is also true, i.e. any cipher that is CCA-secure for single encryption also satisfy CCA-security for multiple encryption. So, CCA-security for single message is sufficient.

7 Malleability

Malleability is the power of the Adversary to easily manipulate known ciphertexts to obtain new ciphertexts so that the relation between the underlying messages are known to him. Then he gets DO service on the changed ciphertext to get the message, and retrieves the original message using the relation.
So, malleability is a big issue in SKE. As we have already seen that how a little help from DO can cause a break in a CPA secure scheme, it can be concluded that CCA security does not guarantee non-malleability.

To implement non-malleability a SKE should satisfy the followings:

- It is nearly impossible to create a new ciphertext.
- Changing a ciphertext should either result in an incorrect ciphertext or should decrypt to a plaintext which is unrelated to the original plaintext.

Message Authentication Codes (MAC) helps us to get such a SKE.

8 Message Integration and Message Authentication

Message integrity and authentication are also part of secure communication.

Not all security concerns are related to the ability or inability of an adversary to learn information about messages being sent. In many cases, it is of equal or greater importance to guarantee message integrity or message authentication in the sense that each party should be able to identify a message received indeed originated from authentic party (issue of message authentication) and a message it receives was exactly the message sent by the other party (issue of message integrity)

In general, one cannot rely on the integrity of communication without taking specific measures to ensure it. Indeed, any unprotected online purchase order, online banking operation, email, or SMS message cannot, in general, be trusted to have originated from the claimed source. Unfortunately, people are in general trusting and thus information like the caller-ID or an email return address are taken to be 'proofs of origin' in many cases (even though they are relatively easy to forge): This leaves the door open to potentially damaging adversarial attacks.

9 Message Authentication in Private Key Setting

The aim of a message authentication code is to prevent an adversary from modifying a message sent by one party to another, without the parties detecting that a modification has been made. As in the case of encryption, this is only possible if the communicating parties have some secret that the adversary does not know (otherwise nothing can prevent an adversary from impersonating the party sending the message). Here, we will consider the private-key setting where the parties share the same secret key.

Two users who wish to communicate in an authenticated manner begin by generating and sharing a secret key **k** in advance of their communication. When one party wants to send a message **m** to the other, she computes a MAC tag **t** based on the message and the shared key, and sends the message **m** along with the tag **t** to the other party. The tag is computed using a tag-generation algorithm that will be denoted by Mac. Upon receiving **(m,t)**, the second party verifies whether this is a valid tag on the message **m** with respect to the shared key or not

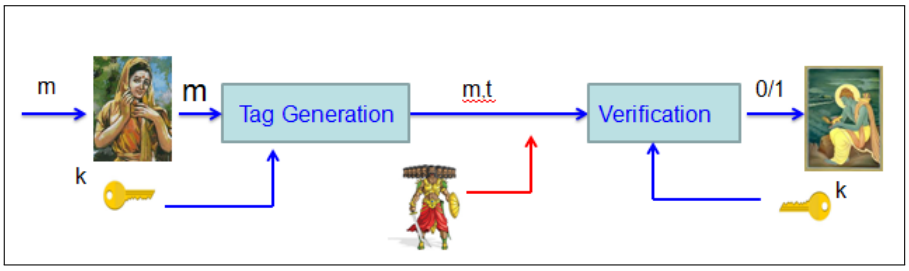


Figure 5: Diagramatic representation of Message Authentication in Private Key Setting

10 Syntax of Message Authentication Codes (MAC)

A message authentication code (or MAC) is a tuple of probabilistic polynomial-time algorithms (Gen, Mac, Vrfy) such that:

- The key-generation algorithm **Gen** takes as input the security parameter 1^n and outputs a key **k** with |k| $\geq$ n.
- The tag-generation algorithm $\mathbf{Mac}_k(m)$ takes as input a key **k** and a message **m**$\in$ (0, 1)*, and outputs a tag **t**. Since this algorithm may be randomized, we write this as **t**$\leftarrow Mac_k(m)$.
- The verification algorithm **Vrfy** takes as input a key **k**, a message **m**, and a tag **t**. It outputs a bit **b**, with **b** = 1 meaning valid and **b** = 0 meaning invalid. We assume without loss of generality that **Vrfy** is deterministic, and so write this as **b**$\leftarrow Vrfy_k(m,t)$.

It is required that for every **n** every key **k** output by **Gen**(1^n) and every **m**$\in$ (0, 1)* it holds that $Vrfy_k(m, Mac_k(m)) = 1$.
If (Gen, Mac, Vrfy) is such that for every **k** output by **Gen** (1^n) algorithm Mac_k is only defined for messages **m**$\in (0,1)^{l(n)}$ and $Vrfy_k$ outputs 0 for any **m** $\notin (0,1)^{l(n)}$ then we say that (Gen, Mac, Vrfy) is a fixed-length MAC for messages of length $l(n)$.

Any MAC defines the following three space (sets):

Key space (K): Set of all possible keys output by algorithm **Gen**.
Plain-text(message)space (M): Set of all possible legal messages.
Tag space (T): Set of all tags output by algorithm **Mac**.

11 Security for MAC

Threat Model: Randomized, probabilistic polynomial-time (PPT) adversaries with capability of Chosen Message Attack (CMA).
Break Model: It is not possible to come up with (m,t) if no tag on m is seen before.

A stronger security definition could be implemented where the threat and break models are different than the previous.

Threat Model: Randomized, probabilistic polynomial-time (PPT) adversaries with capability of Chosen Message and Verification Attack (CMVA).
Break Model: It is not possible to come up with (m,t) if (m,t) has not been seen before.

12 The message authentication experiment: $[mac - forge_{A,\Pi}(\mathbf{n})]$

The Verifier generates a random key **k** by running Gen (1^n).

Training Phase:

The adversary **A** is given oracle access to Mac_k. The adversary sends messages m and gets tag for several messages of its choice adaptively. Let **Q** denote the set of all queries that **A** asked to the Verifier; $\mathbf{Q} = (m_1, m_2..., m_I)$.

Challenge:

After the training **A** forges a tag (m,t) and sends for verification. The output of the experiment is defined to be 1(**A** succeeds) if and only if
(1) $Vrfy_k$(m,t) = 1 ,and
(2) m $\notin$ **Q**.

SECURITY DEFINITION:

A message authentication code Π= (Gen, Mac, Vrfy) is existentially unforgeable under an adaptive chosen-message attack, or CMA- secure, if for all probabilistic polynomial-time (PPT) adversaries **A**, there exists a negligible function negl such that:

$$\Pr[\ Mac - forge_{A,\Pi}\ (\mathrm{n}) = 1] \leqslant \mathrm{negl(n)}$$

MAC-Strong Experiment: $[mac - sforge_{A,\Pi}(\mathbf{n})]$

For stronger definition of MAC we can perform an experiment similar to MAC. Here the only change is that **Q** is now the set of all possible queries and corresponding tags. **Q** = (m_1, t_1),..,(m_l, t_l). So now the output of the experiment is defined to be 1(**A** succeeds) if and only if
(1) $Vrfy_k$(m,t) = 1, and
(2) (m,t) $\notin$ **Q**.

SECURITY DEFINITION:

A message authentication code Π= (Gen, Mac, Vrfy) is existentially unforgeable under an adaptive chosen-message attack, or strong CMA- secure, if for all probabilistic polynomial-time (PPT) adversaries **A**, there exists a negligible function negl such that:

$$\Pr[\ Mac - sforge_{A,\Pi}\ (\mathrm{n}) = 1] \leqslant \mathrm{negl}(\mathrm{n})$$

13 References

1. Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography, second edition*, CRC Press, 2014.
2. Arpita Patra. http://drona.csa.iisc.ernet.in/ arpita/Cryptography16.html. Course Materials.