CSA E0 235: Cryptography (February 7th, 2016)

Scribe for Lecture # 8

Instructor: Arpita Patra *Submitted by: (Arya Boudaie)*

1 Organizational Notes

- Wednesday class (3-2-16) was moved to Thursday (4-2-16) 9:30-11
- Next week: Monday (8-2-16) will be a tutorial (going over problems in the book)
- Wednesday (10-2-16) will be Chalk and Talk sessions

2 Recall

- Wednesday class was moved to Thursday 9:30-11
- Next week: Monday will be a tutorial (going over problems in the book)
- Wednesday will be Chalk and Talk sessions

3 Recall

- Last class we moved away from COA (Ciphertext only) Security to CPA (Chosen plaintext) security
- To do this we showed PRFs (pseudo-random functions), and then constructed an SKE from PRF, then proved the security of that SKE
- Finally we discussed practical SKEs from PRF/PRP/SPRP - ECB, CBC, OFB, and CTR. We showed that ECB was not even CPA secure, and that CTR had the most features.

4 Summary of Today

- Upgrade from CPA to CCA (Chosen Ciphertext Attack)
- As always, model it in a security definition, and show that it is stronger than CPA.
- Attack the CBC encryption that was proven to be CPA secure, and also attack the theoretical PRF-based construction
- Discuss the assumptions and primitives needed to build a CCA-secure construction
- Begin talking about MAC (message authentication code), which provides message authentication and integrity, something we have not feared before (people messing with the ciphertext).

5 Modeling CPA security

- Compared to CPA security, there is no difference in the break model- "Given the knowledge of two messages, it cannot be distinguished if the ciphertext corresponds to the first or second message."
- Compared to the threat model in CPA, the attacker is still Randomized and PPT, but they can now run a CCA attack
- In the CCA attack, the adversary can now influence the honest parties, as opposed to just being an observer - can send them ciphertext and see how they respond
- This is modeled by giving the attacker not just oracle access to the encryption machine, but also oracle access to the decryption machine - the question is given this oracle access can they figure out the content of a new ciphertext.
- In practical terms, this is a much more realistic threat, as getting access to the decryption machine is much easier than getting access to the encryption machine, and a little help from the decryption oracle is very detrimental

6 Example of a practical CCA attack

- Assume a customer sends a message to their bank along the lines of "Transfer X to account #Y" - which is encrypted before sending
- The adversary can then take this ciphertext c, and try to make a c'' that says "Transfer 1000X to account #Y"
- When the adversary sends the new c', if the edit was successful, the bank might reply by sending the customer a call asking "Did you make this transfer?", which will verify to the adversary that they correctly broke the scheme.
- Important to note that the adversary is now malicious! Not just a passive force in the system.
- Another example might be an adversary sending a c' message to an army's general, and seeing what the army does in response, to hint at the meaning of c''
- The honest party might even give access to the decryption oracle as part of their service

7 Recap on CBC and Intro to Padding

- Access to the DO is very powerful, even knowing if a modified ciphertext was decrypted correctly or not can help the adversary break the system
- An example is the padding oracle attack done on CBC Encryption on a message m, where the length of m is a multiple of block length L in bytes

- Remember from last class that CBC (Cipher Block Chaining) works as follows:
  1. a message m is broken up into blocks length L
  2. An initialization vector IV length L is chosen as c_0
  3. For each block (starting at i=1): Encrypt it with $c_i = F_k(m_i \oplus c_{i-1})$ where F_k is a PRF with randomly chosen key k
  4. The receiver can decrypt each block with $m_i = F_k^{-1}(c_i \oplus c_{i-1})$
- If $|m|$ isn't divisible by L, you have to introduce padding to the end of a block. One such padding method is PKCS#5 padding
- In this padding scheme, there are b bytes that need to be appended to the last block of m to make its length L bytes. Add b bytes to the last block of m, with each byte representing the integer value of b
- if the length of m is divisible by L, the entire last block needs to be padding, so there is no ambiguity
- Keep in mind that there are 2^8 possibilities for each byte, so L needs to be smaller than that. Practical sizes of L are either 16 or 8 bytes.
- After the padding, encrypt as usual. The receiver will decrypt as usual, and make sure the last b bytes of the last block are the integer values of that byte. If they are, then the decryption algorithm will strip off those values, otherwise it will report a bad encoding and request another message.
- Because of this, if $|m|$ is divisible by L, the last block needs to be padding, in order to avoid ambiguity (what if the original message happened to end with the byte values of 1,2,3...?)

8 Padding Oracle Attack

- With the CCA attack scheme, an attacker can modify ciphertexts and learn b (which makes finding $|m|$ trivial) and m itself!
- To make it easier to discuss, we assume m has been split into m_1 and m_2
- If the attacker changes the i^{th} byte of c_1, then both m1 and m2 will change at byte i. As long as the attacker doesn't modify the padding, the message will be accepted, so to find b, all the adversary has to do is keep moving through the message and modifying the i^{th} byte of c_1 (for i from 1 to $|m|$) until the message is rejected by the decryption service.
- If i is the bit that was modified when the decryption service rejected the ciphertext, then b = L i + 1, we now know b, L, and $|m|$

- Now that the adversary knows b, it can use this information to try to crack m. It knows that m_2 is of the form m+B+b+b..., where m is the legitimate message, B is the first padding bit, and the rest of the b's are the next padding bits. The adversary can try to extend the message length by modifying the padding bits - if the decryption oracle reports a success, it means that B+1 was successfully changed to the integer value of b+1 - the adversary only has to try 256 times to get it to work. We now know what modification of the ciphertext leads to a specific plaintext (using $\oplus$).
- Run this for each B to take over the entire message!
- The moral of this story is that the attacker can control what is decrypted in this new attack, and that helps them in all new ways. We must now capture CCA in a security definition

9 CCA Indistinguishability Experiment

- The CCA Indistinguishability Experiment is very similar to the experiment for CPA. The main difference is that the adversary now gets to submit queries to the Encryption and Decryption oracles (as opposed to just the encryption oracle).
- It begins with a training phase, where the adversary is allowed to send a polynomial number of queries to either the encryption or decryption oracle
- After is the challenge phase, where the adversary sends two messages (can be messages encrypted in training), and the challenger encrypts a random one of the two and sends it back to the adversary as c.
- After the adversary gets its message, it has a post-training phase, with the only restriction being the adversary cannot decrypt c (as that would be too easy).
- After the post-training phase, the adversary has to guess if it was m_1 or m_2 which was encrypted. The game outputs 1 if the adversary was correct, 0 otherwise.
- As before, a scheme is CCA secure if for every PPT adversary A, there is a negligible function negl such that the probability of A winning the CCA Indistinguishibility game with n bits is $\frac{1}{2}$ + negl(n)

10 CCA Security for Multiple Encryptions

- We can also design a cca-mult game that seems stronger. It is basically the same, except in the challenge phase the adversary sends two vectors of messages, and the challenger randomly encrypts every message in one of these vectors and sends it back, and the adversary has to guess which vector was encrypted after the same post-training phase.
- We can prove that this experiment is equivalent to the regular CCA game, but it requires a hybrid proof model. For now we can just prove it's secure for cca, cca-mult comes for free.

11 Proving CCA is stronger than CCA

- Imagine an adversary playing the CCA game against an encryption scheme $Enc_k(m)$ = (r, F_k(r) $\oplus$ m)
- A possible strategy is for the adversary to skip the pre-training phase, and request the encryption of either m_1 = (000..0) or m_2 = (111..1) in the challenge phase.
- The challenger will select a random bit 0 or 1, and will send back c* = (r, s*) = (r, F_k(r) $\oplus$ m_b)
- All the adversary would have to do to break the encryption would be to ask the Decryption oracle to decrypt c = (r, s), where s=s* but with the first bit inverted. The challenger would send back m = Dec_k(c). Then if m is (1000..0), the adversary will choose b=0, and if m is (0111..1) then the adversary will choose b=1, a strategy which will win with 100% probability.
- We have to think about it - what about the scheme makes it so easy for the adversary to win? It's mostly the fact that it's very easy to manipulate the ciphertexts - we know that changing a byte in the ciphertext will directly affect that byte of every block in the plaintext. In short, CPA doesn't guarantee non-malleable SKEs.
- We need an SKE where creating a new (valid) ciphertext is nearly impossible, and one where changing an existing ciphertext either makes it invalid or changes it completely, making oracle access to the decryption machine useless.
- We basically need MAC (Message Authentication Codes)

12 Intro to MAC

- MAC gives us a guarantee that we haven't worried about the entire course. The guarantee that a message was sent by a person, and that it hasn't been modified in transit.
- This is a completely different problem than privacy, but still extremely important. It is complimentary to the SKE problem, but a completely different domain.
- How does it work? In short, two parties share a key in advance. When the sender wants to send a message, they'll pass the message and the key through a tag generation algorithm, and send along the tag and the message. The receiver has a verification algorithm, which can take the message, tag, and key, and report if the tag is valid for the message or not. That way, if a 3rd party tries to change the message, the verification algorithm will detect the change and the receiver will know the message was tampered with.

13 MAC Specifications

These are the algorithms for MAC (All Polynomial)

1. Gen() - just like the other KeyGen algorithms we have dealt with
2. $Mac_k(m)$ - Tag generation algorithm that takes a key and a message, and returns a tag. Either deterministic or randomized
3. $Vrfy_k(m,t)$ - Verification algorithm that takes a message and a tag, and returns either 0 (if tag is not valid) or 1 (if tag is valid). Usually deterministic.

Mac also has the following spaces:

1. $\mathcal{K}$ - the set of all possible keys produced by Gen()
2. $\mathcal{M}$ - the set of all possible messages supported by Mac
3. $\mathcal{T}$ - the set of all possible tags produced by $Vrfy_k(m,t)$ (Defined by $\mathcal{K}$ and $\mathcal{M}$)

Other notes

- Any MAC is defined by specifying (Gen, Mac, Vrfy) and $\mathcal{M}$
- To prove correctness, show that for every key and message,$Vrfy_k(m, Mac_k(m)) = 1$ (Basically that the tag generated on a message is also valid for that message).

14 Break and Threat

- As always, we have to model our threat, and our break
- We have two different threat models: The first is a CMA (Chosen Message Attack), where the adversary can see messages and train itself with corresponding tags
- A stronger model would be the CMVA (Chosen Message Verification Attack), where the adversary can both send messages to get tags, and verify messages on tags
- There are two break models as well. The first says our MAC is secure when after this training it is not possible to come up with (m,t) if no tag on m is seen before. The second says our MAC is secure if after the training it is not possible to come up with (m,t) if (m,t) has not been seen before.
- To model the break models in an experiment, we have a similar setup to before. In the MAC-forge game (modeling the first), an adversary has a training phase, where they can send a polynomial number of messages to the challenger, who will give the corresponding tag. After the training phase, the adversary will send a message and a tag, and the game will output 1 if the message has never been seen before, and the tag is valid for the message, and 0 otherwise.
- To model the second break model, we have a similar experiment, but the game outputs 1 if that message/tag pair have never been seen before, and the tag is valid for the message. This gives the adversary much more leniency.