0368.4162: Introduction to Cryptography Ran Canetti

Lecture10

05 January 2009 Fall 2008

Scribes:O.Singer,M.Shaked

Topics for today

- Public-Key Encryption
- Public Key Infrastructure (PKI)
- Authenticated Key Exchange

Note: These three topics are very wide, and we could have had an entire course on just the main issues here. The lecture today will only give a flavor of the basic concepts.

1 Semantically secure PKE

Public-key encryption extends symmetric encryption in much the same way that digital signatures extend message authentication codes (MACs). That is, instead of having the same key give the ability to both encrypt and decrypt messages, here there are two keys: An encryption key, which is thought of as public, and a decryption key that is known only to a privileged party. This way "anyone" can encrypt "to a party", with the confidence that only that party can decrypt. (This is true, of course, only to the extent that the encryptor has the correct encryption key that's associated with the recipient; we'll deal with this issue later).

Definition: A public key encryption (PKE) scheme for domain $M = \{M_n\}_{n\in N}$ consists of three algorithms: $E = (GEN, ENC, DEC)$ such that:

1. $GEN(1^n) = (ek, dk)$
2. for any $m \in M_n$,
   $Prob[(ek, dk) \leftarrow GEN(1^n), DEC(dk, ENC(ek, m)) = m] = 1$

Note: Here the encryption is inherently stateless - it doesn't make sense to have a scheme that keeps memory from one encryption to another, since different encryptions can be made by different parties. (This corresponds to the fact that verification is inherently stateless in a digital signature scheme.)

Semantic security and security by indistinguishability are extended from the case of symmetric encryption in a straightforward way. Here will provide some definitions that somewhat differ from those in the previous lecture. Apart from the fact that here we will be dealing with asymmetric encryption, the changes are:

1. We define attacks that are CPA (as opposed to KPA)
2. We add in the concept of environment directly into the definition

These changes will simplify the definition without changing the mathematical meaning. See Figure 1 for a graphical description of the definition.

Explanation:

- In the real interaction, Gen sends ek to E, then E sends m to ENC, then ENC sends c to ENV, then ENC outputs a bit b.
- In the ideal interaction, the simulator S sends ek to ENV, then ENV sends m to the ideal encryption module IE, then IE sends $l(m)$ to S, then S sends cipher-text c to IE, who forwards it to ENV.

Definition: SEM-CPA (Semantic security for asymmetric encryption) An asymmetric encryption scheme (GEN, ENC, DEC) is semantically secure with message domain $M = \{M_n\}_{n \in N}$ and leakage function $l()$ if there exists a polynomial algorithm S ("Simulator") such that for every polynomial E ("Environment"):

$Prob[(ek, dk) \leftarrow GEN(1^n), (m, s_{ENV}) \leftarrow E(ek), c \leftarrow ENC(ek, m), E(s_{ENV}, c) = 1] -$
$Prob[(ek, s_{ENV}) \leftarrow S(1^n)], (m, s_{ENV}) \leftarrow E(ek), c \leftarrow S(s_{SIM}, l(m)), E(s_{ENV}, c) = 1] < \nu(n)$
where $\nu(n)$ is some negligible function.

Definition: IND-CPA (security by indistinguishability for asymmetric encryption) Given an asymmetric encryption scheme $E = (GEN, ENC, DEC)$ for domain M and leakage function $l(m)$, and an adversary B, define the following game:

- $(ek, dk) \leftarrow GEN(1^n)$
- B receives ek
- B chooses two messages $m_0, m_1 \in M$ such that $l(m_0) = l(m_1)$
- B receives $ENC(ek, m_b)$ for a random bit b
- B outputs a bit b'.

B wins the game if $b = b'$.

The scheme E is secure by indistinguishability with respect to domain M and leakage function $l()$ if for any poly time adversary B we have

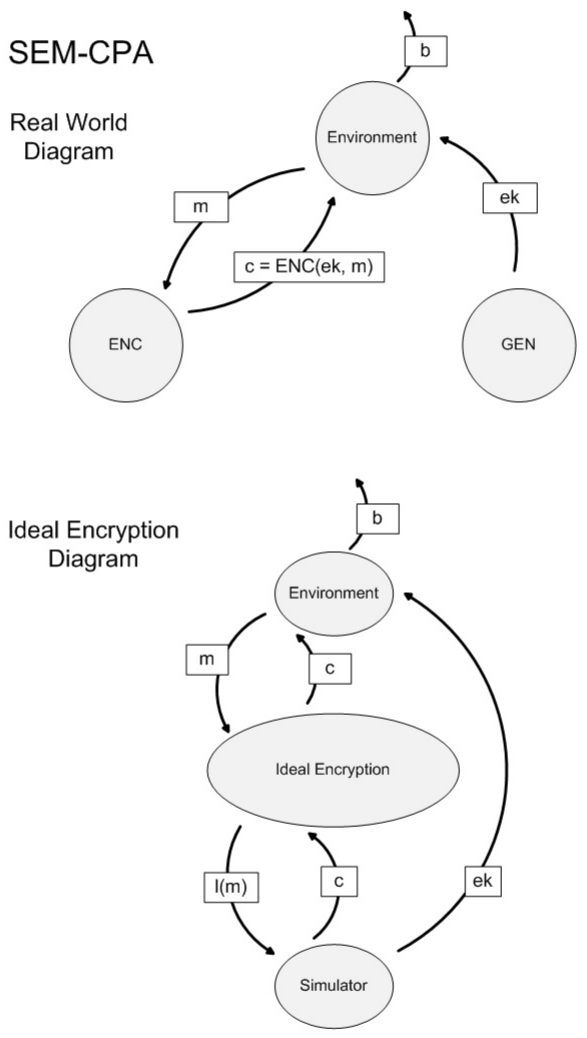

Figure 1: Real Execution and Ideal Execution for CPA Attack on PKE

$Prob[(ek, dk) \leftarrow GEN(1^n), b \leftarrow \{0,1\}, (m_0, m_1, s) \leftarrow B(1^n), b' \leftarrow B(1^n, ek, ENC(ek, m_b), s) : l(m_0) = l(m_1) \& b' = b] < \frac{1}{2} + \nu(n)$

for some negligible function $\nu(n)$.

(Here s represents the fact that B keeps state throughout the experiment.)

We have: As in the symmetric case, the notions are equivalent:

Theorem: A PKE scheme is semantically secure iff it is secure by indistinguishability.

The proof is a simple extension of the symmetric case.

- A SEM/IND PKE scheme must be probabilistic (since there cannot be any state).
- As in the symmetric case, SEM/IND for a single message implies SEM/IND for multiple messages.
- Security against chosen plain-text attacks (CPA) is a stronger notion that's captured by letting B see the public key before choosing m_0, m_1.
- Unlike the symmetric case, IND-CPA for a single message is equivalent to IND-CPA for multiple messages. This can be seen in a straightforward way using hybrids. (The difference from the symmetric case is that here the adversary has the encryption key and thus can encrypt on its own.)
- The definition for multiple messages is the same as that for a single message except that the adversary generates many pairs (m_o^i, m_1^i) and receives $ENC(ek, m_b^i)$ for each i.

Claim: If an encryption scheme is IND-CPA for a single message then it is IND-CPA for multiple messages.

Proof:

Given an adversary A' that wins the game after hearing q cipher-texts, construct an A that wins after hearing one cipher-text:

- A chooses $i \leftarrow [1..q]$
- All queries (m_0^i, m_1^i) of A', up to i^{th} query are answered by $ENC(m_0^i)$ (where A generates the encryption by itself)
- The i^{th} query is forwarded to the oracle of A and the response is forwarded to A'.
- All queries (m_0^i, m_1^i) of A', from the $i+1$ query on are answered by $ENC(m_1^i)$
- When A' outputs b', A' output b'

To analyze A:

Define the j^{th} hybrid distribution H_j to be the output of A' when the j first queries are answered by m_0 and the remaining queries are answered by m_1

We know that $|H_q - H_0| > 2e$.

$Prob[A\, wins] =$

$\frac{1}{2}(Prob[A\, outputs\; 1|b=1] + Prob[A\, outputs\; 0|b=0]) =$

$\frac{1}{2} + \frac{1}{2}Prob[A\, outputs\; 1|b=1] - prob[A\, outputs\; 1|b=0]) =$

$\frac{1}{2} + \frac{1}{2}q\sum_{j=1}^{q}(Prob[A\; outputs\; 1|b=1, i=j] - Prob[A\; outputs\; 1|b=1, i=j) =$

$\frac{1}{2} + \frac{1}{2}q\sum_{j=1}^{q}(H_j - H_{j-1}) =$

$\frac{1}{2} + \frac{1}{2}q(H_q - H_0) > \frac{1}{2} + \frac{e}{q}$

Historical note:

Semantic security and security by indistinguishability were proposed in the seminal paper by Goldwasser and Micali ("probabilistic encryption"). Before this paper, there was no rigorous notion of what it means for an encryption scheme to be "secure" (beyond Shannon secrecy) . In fact, this paper marks the beginning of the rigorous treatment of security in modern cryptography. for instance:

- It is the first to use an "abstract" definition of security.
- It is the first to rigorously capture secrecy against computational bounded adversaries.
- It introduces the notion of encryption as an inherently randomized process (the "Simulation paradigm") which permeates all of modern cryptography.
- It is the first to use the "hybrids method".

2 CPA-IND Public Key Encryption schemes

We'll see two schemes:

- The El Gamal scheme
- A HCP+TDP scheme

2.1 The El Gamal scheme

Let G_n be some group of size $q \sim 2^n$, where q is prime. (Can think of G_n being the group of squares in Z_p^*, where$p = 2q+1, p, q$ primes.) Let g be a generator of G. The domain is $M_n = G_n$.

- $GEN(1^n): \ ek = (g, g^x), \ \ dk = x, \ x \leftarrow [1..q]$
- $ENC(m) = g^r, m \cdot g^{xr}$ where $r \leftarrow [1..q]$
- $DEC(a, b) = b/a^x$

Correctness is straightforward: $m \cdot g^{xr}/g^{xr} = m$.

Theorem: The EG scheme is IND-CPA secure.

Proof: By reduction to the DDH assumption.

Recall the DDH assumption: $g, g^x, h, h^r \approx_c g, g^x, h, h^x$

Assume an adversary B that wins the IND-CPA game with probability $\frac{1}{2}+e$. Construct an adversary A that distinguishes g, g^x, h, h^r from g, g^x, h, h^x with probability e.

A works as follows, on input $\alpha, \beta, \gamma, \delta$

- A runs B on public key α, β
- When B generates m_0, m_1, A returns the "cipher-text" $(\gamma, \delta \cdot m_b)$ for random b.
- When B outputs b', A outputs $b + b'$.

(Note that by the above claim it suffices to consider a single cipher-text.)

Analysis:

- If A's input is a random quadruple then B's view is independent from b, thus A outputs 1 with probability exactly $\frac{1}{2}$.
- If A's input is a "DH tuple" then B's view is exactly that of an IND-CPA attack for EG. Thus by assumption $b' = b$ w.p. $\frac{1}{2} + e$.

2.2 A HCP+TDP scheme:

Let $P = (G, S, F, F^{-l})$ be a trapdoor permutation (TDP) and let B be a hard core predicate for P. The domain is $M = \{0, 1\}$

1. $GEN(l^n) = G(1^n) = (i, t)$ (i.e.: $ek = i, dk = t$)
2. $ENC(i, m): \ x \leftarrow S(i)$, output: $c = (F(i, x), m + B(x))$
3. $DEC(t, y, b) = b + B(F^{-l}(t, y))$

Correctness is straightforward.

Theorem: The HCP+TDP scheme is IND-CPA.

Proof: By reduction to the assumption that $F(i,x), B(x) \approx_c F(i,x), r$ where $i \leftarrow G(l^n), x \leftarrow S(i), r \leftarrow \{0,1\}$.

This is very similar to the previous proof and we'll skip.

How can we extend this scheme to encryption of many bits? One option is to encrypt each bit individually and concatenate: $|c| = n \cdot |m|$.

Can you find a way to shorten the cipher-text, based on the same assumptions? (Can achieve $|c| = n + |m|$)

3 Security against chosen cipher-text attacks (CCA)

So far we considered adversaries that only listen in on messages. What about adversaries that try to attack the system by generating "malicious cipher-texts" and observing the behavior of the decryption algorithm when run on these cipher-texts?

In the case of symmetric encryption we "sidestepped" this concern by introducing the concept of a secure channel: The authenticity guarantee of the channel makes sure that the receiver rejects any message that was not generated by the sender (i.e. by the prescribed encryption algorithm). Since the sender is the one who's interested in keeping the messages secret, we could avoid considering what happens when the receiver sends strange cipher-texts.

This logic does not work in the asymmetric case, since we want a decryptor to be able to decrypt, using the same decryption key, cipher-texts that were generated (under the same public key) by multiple different parties. Even authenticating the encryption doesn't help, since a cipher-text can be completely authentic (in the sense that it comes from the claimed source) and still be maliciously formed.

While this might seem like a scenario made up by a paranoid cryptographer, it is actually not so far fetched. For example, the Bleichenbacher attack on SSL encryption (Crypto '98) was made possible due to the underlying PKE being susceptible to such attacks.

So how to capture the security of encryption schemes against such attacks?

Two ways:

1. extend the IND definition:
2. extend the SEM definition

We'll do both, and will again get equivalent notions.

Let's start with the IND definition. We'll extend the IND game, giving the adversary also access to a decryption oracle.

Definition: $EN = (GEN, ENC, DEC)$ is IND-CCA if every adversary B wins the following game with probability $< \frac{1}{2} + \nu(n)$

IND-CCA game, for scheme $EN = (GEN, ENC, DEC)$, domain M_n and leakage function $l()$:

1. $(ek, dk) \leftarrow Gen(l^n), b \leftarrow \{0, 1\}$
2. A gets ek
3. Repeat:
   (a) If A generates cipher-text c then A receives $DEC(dk, c)$
   (b) If A generates (m_0, m_1) s.t. $l(m_0) = l(m_1)$ then A gets $c* = ENC(ek, m_b)$
   (c) If A outputs b' then game ends. If there is already a pair (m', c) recorded for some m', then output an error message and halt.

This is an unsatisfiable definition because A can always win the game by asking DEC to decrypt $c*$ in the second iteration (A will get m_b from $DEC(dc, c*)$) and outputting 0 on $m_b = m_0$ otherwise outputting 1.

The fix: change DEC so that it doesn't decrypt $c*$. This may seem like a suspicious fix– clearly without the fix the definition is too strong, but why is it meaningful with the fix? One answer is that it is meaningful since it will be equivalent to the semantic definition.

Remarks:

- WLOG can restrict A to ask only a single encryption query (same argument as before). This is not true for decryption queries.
- A can ask decryption queries either before or after receiving $c*$. This gives much power, since A can direct its queries based on the challenge cipher-text. (Indeed, the [B] attack used this type of adaptivity.

Still, it is sometime interesting to consider schemes where the adversary can ask decryption queries only before seeing $c*$. This type of security is often called "non-adaptive CCA security", or "security against lunchtime attacks" or "CCAl security".

Historically, we first knew how to protect against CCAl attacks before we knew how to protect against full CCA (or, CCA2) attacks.

4 Semantic security against CCA

We'll formulate an extension of semantic security that captures chosen cipher-text attacks, and will show that it is equivalent to both IND-CCA and NM-CCA. This will give us a "robust characterization" of security of encryption in the presence of active adversaries.

As usual, we'll:

- Define the real execution
- Define the ideal scheme
- Define "same as"

4.1 Real execution

1. Participants: (GEN, ENC, DEC), Environment E
2. Receiver runs: $GEN(l^n) \rightarrow (ek, dk)$, gives ek to E
3. Repeat:
   (a) E sends a message m to the sender
   (b) Sender sends $ENC(ek, m)$ to A
   (c) A sends some information v to E and a cipher-text c to the receiver
   (d) The receiver sends $DEC(dk, c)$ to Z
4. Continue repeating until E outputs a bit.

Let $REAL_{GEN,ENC,DEC,E,A}$ denote the output of E in this experiment.

4.2 Ideal execution

Participants: Ideal Encryption (IE), Environment E, Adversary S

1. S gives a key ek to E
2. Repeat:
   (a) E sends $("encrypt", m)$ to IE and IE sends $l(m)$ to S
   (b) S chooses a cipher-text c and sends it to IE
   (c) IE records the pair (m, c) and sends c to E
   (d) E sends $("decrypt", c')$ to IE. If c' appears in a recorded pair (m, c') then IE sends m to E
      i. Else, IE sends c' to S, gets a value m' from S, and outputs m' to E

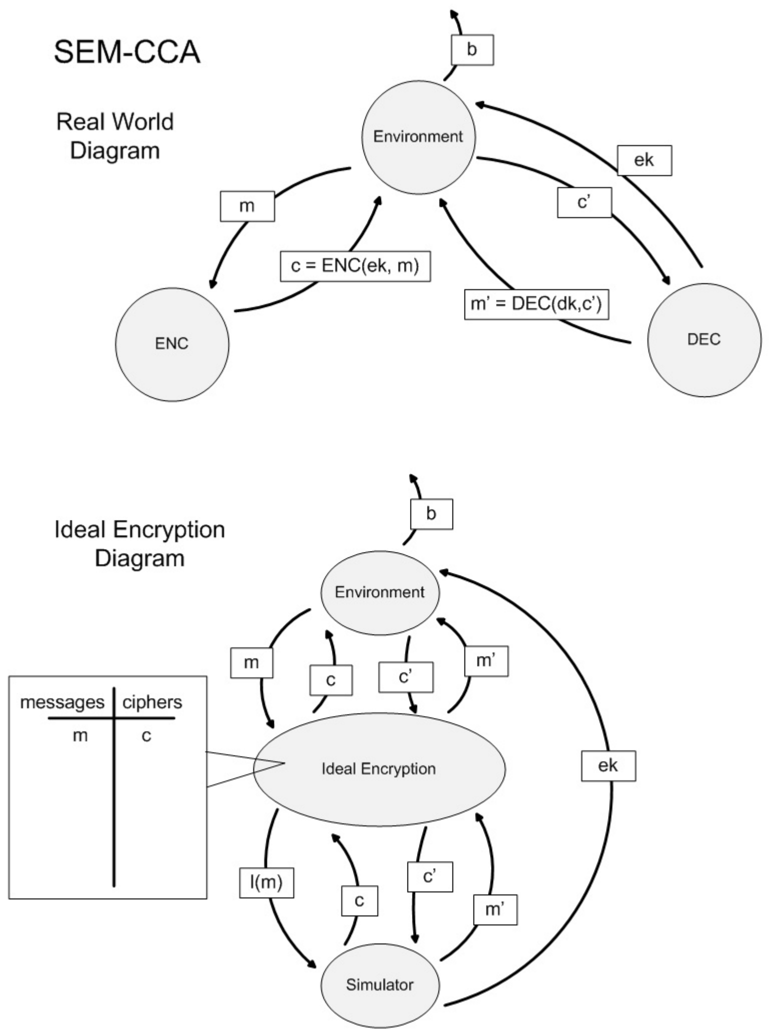

Figure 2: Real Execution and Ideal Execution for CCA

3. Continue repeating until E outputs a bit.

Let $IDEAL_{E,S,IE}$ denote the output of E in this experiment.

Rationale:

In the ideal execution:

- Correct decryption of honestly generated cipher-texts is guaranteed unconditionally.
- Secrecy of correctly encrypted plain-texts is guaranteed unconditionally.
- We don't care how the cipher-text look, nor do we care about decryption of maliciously formed cipher-texts. Thus we let the simulator choose these.

Definition: A PKE scheme (GEN, ENC, DEC) is SEM-CCA if there exists an adversary S such that for any environment E, $REAL_{GEN,ENC,DEC,E} \approx_c IDEAL_{E,S,IE}$.

Theorem: Let EN=(GEN,ENC,DEC) be a PKE. Then EN is SEM-CCA iff it is IND-CCA.

Proof Sketch: IND-CCA => SEM-CCA

Assume EN is IND-CCA. We construct a simulator S for A and EN:

1. S runs $GEN(l^n) \rightarrow (ek, pk)$, chooses $b \leftarrow \{0, 1\}$
2. When S receives $l(m)$ from IE, it chooses some $m*$ such that $l(m*) = l(m)$, and returns $c = ENC(ek, m')$
3. When S receives c from IE, it returns $m = DEC(dk, c)$

Now, assume there is an E that distinguishes between the REAL and the IDEAL executions. We use E to construct an adversary B that wins the IND-CCA game (here we use the version with multiple encryption queries):

- B runs E.
- Whenever E gives message m to the sender, B generates an encryption query (m_0, m_1, where $m_b = m$, and $m_{l-b} = m*$), where $m*$ is the message that S encrypts when it gets $l(m)$.
- Whenever E gives a cipher-text c to the receiver, B generates a decryption query c and returns the answer to E.
- If E outputs "real", then B outputs b, if E outputs "ideal", B outputs $1 - b$.

The claim is that when bit b that is chosen by B is the same as b' chosen in the IND-CCA game, algorithm E sees the a real-world run. When $b \neq b'$, E sees an ideal run. In order to make this into a real proof, we need to go over the details to see that they all fit in– we will not do this.

SEM-CCA => IND-CCA

Assume there exists an adversary B that wins the IND-CCA game.

We construct an environment E that cannot be "simulated" by any S:

- E chooses a bit r at random, and then runs B:
  - E gets ek, hands to B
  - B generates decryption requests c, E hands to A, gets corresponding decryption m, hands back to B
  - B generates m_0, m_1, E sends m_r to be encrypted, gets back cipher-text $c*$ from A, and hands it to B
  - B outputs b, E outputs $b + r$

  The claim is that E is in the real interaction, then B sees a standard CCA game for EN, thus predicts r. But, if E is in the ideal interaction then B's view is independent from r, since b is independent of r.

Are the TDP+HCP and El Gamal schemes CCA secure? In the case of TDP+HCP, an adversary can win the IND-CCA game by getting the cipher-text c of some message $m \leftarrow \{0,1\}$ then creating c' by flipping the last bit of c. By requesting the decryption of c', the attacker will be able to learn whether c was an encryption of 0 or 1.

Similarly, El Gamal is not CCA secure because an adversary can win by getting the cipher-text (c_1, c_2) of some message m, then creating a cipher-text $(c_1, 2c_2)$ which will be valid for the message $2m$.

5 Non-malleability of encryption

Let us now consider yet another security concern regarding PKE. This notion is in part motivated by the above attacks on EG and HCB+TDP. So far, we were only concerned about adversaries that try to explicitly "learn", or "output" the plain-text. But this is not the only possible way to attack an encryption scheme. What about an adversary that can generate, given an encryption of 1, an encryption of 0, as in the above attack? More generally, what about an adversary that can, given an encryption of m, generate an encryption of m' such that (m, m'), satisfy some other relation? This is a concern in its own right, even, say, in the case of CPA attacks.

As we saw from the examples of EG and HCB+TDP, this concern does not follow from the plain notion of semantic security. This is somewhat surprising, since we claimed that semantic security means that the adversary "learns nothing" about the plain-text. This may be viewed as another manifestation of the "frailty" of the cryptographic notions of security - and even more so the inadequacy of our intuition.

Non-malleability is a notion that's aimed at making sure that cipher-texts cannot be modified (or, "mauled") into other ones while preserving some properties of the underlying plain-texts.

Formalization: Let's extend the IND-CPA and IND-CCA games to capture this concern.

NM-CPA game, for adversaries A,D:

1. scheme: $E = (GEN, ENC, DEC)$, domain: M_n and leakage function $l()$
2. $Gen(l^n) -> (ek, dk), b \leftarrow \{0, 1\}$
3. A gets ek
4. A generates (m_0, m_1) s.t. $l(m_0) = l(m_1)$ and gets $c* = ENC(ek, m_b)$
5. A outputs c
6. D gets $DEC(dk, c), m_0, m_1$ and outputs b'
7. A, D win if $c \neq c*$ and $b' = b$

The idea here is that A no longer needs to explicitly guess b. It suffices for A to generate a cipher-text such that the corresponding plain-text relates to m_b in a different way than it relates to m_{l-b}. Then, D gets the decryption of c and gets to compare the plain-texts.

NM-CCA game, for adversaries A,D:

1. scheme $E = (GEN, ENC, DEC)$, domainM_n and leakage function $l()$
2. $(ek, dk) \leftarrow Gen(1^n), b \leftarrow \{0, 1\}$
3. A gets ek
4. Repeat:
   (a) If A generates cipher-text c then A receives $DEC(dk, c)$
   (b) If A generates (m_0, m_1) S.t. $l(m_0) = l(m_1)$ then A gets $c* = ENC(ek, m_b)$
5. When A outputs $("test", c)$, D gets $DEC(dk, c), m_0, m_1$ and outputs b'
6. A, D win if $c \neq c*$ and $b' = b$

Same intuition, with the presence of a decryption oracle.

Perhaps surprisingly (or perhaps not), in the case of CCA attacks the two notions are the same:

Theorem: A PKE scheme is IND-CCA iff it is NM-CCA

Proof:

> Assume there are adversaries A, D that win the NM-CCA game against $E = (GEN, ENC, DEC)$ w.p. e. Construct A' that wins the IND-CCA game w.p. e. A' will run A, and when A outputs $"test"c$, A' will use it decryption oracle to decrypt c to get m, and will run $D(m, m_0, m_1)$. (Note that $c \neq c*$, or else A doesn't win.)
>
> Now, assume there is an A' that wins the IN-CCA game w.p. e, construct A, D that win the NM-CCA game:
>
> A'runs A, gets the guess b' of A, and outputs $ENC(ek, m'_b)$. D outputs 1 iff the first and second messages are the same.

So, in all, we have three equivalent notions of security of encryption in the face of active attacks: IND-CCA, NM-CCA, SEM-CCA. This is indeed a robust and very useful notion of security in many applications. We will refer to these as CCA secure schemes.

6 Constructions of CCA secure encryption

As opposed to the case of CPA (where there are many different constructions), there are much fewer constructions of CCA secure schemes - and these tend to be rather complex.

We'll only mention a heuristic construction, based on a TDP (G, S, F, F^{-l}) and a CHF H:

- $GEN(l^n) = G(l^n) = (i, t)$
- $ENC(i, m) = F(r),\ H(0, r) + m,\ H(1, r, m)$
- $DEC(t, c = (y, a, b))\ :\ r \leftarrow F - 1(y),\ m \leftarrow H(0, r) + a$. If $H(l, r, m) = b$ then output m; Else output 1.

Note the paradigm here: The decryption performs some "consistency check" on the cipher-text before decrypting. This paradigm persists in all CCA schemes.

We know how to prove that this scheme is CCA secure when we abstractly model H as a public random function (a random oracle). But we don't know how to prove it based on "reasonable" assumptions on concrete F and H.

Also, as in the case of signatures, the constructions we know based on security of more basic primitives look very different.

7 Public key infrastructure

Digital signatures and PKE are a great enabler for cryptography in an open, multi-user environment: Without them each pair of parties would need to setup a trust association and establish a shared key in order to communicate. With public keys, this process can be greatly simplified: Parties can post public keys associated with their names (or, more generally, their identities). Then, when one party wants to communicate with another party, it only needs to get their public key from a place it trusts, and then it can start communicating right away.

For this purpose people have constructed "certificate authorities" (CAs), which serve as places for posting public keys. The idea is simple:

- The user U presents a public key PK to the CA
- The CA signs (using its own signing key) the pair (U,PK) and gives the signature S to U.
- now, U can publicize the tuple (U,PK,S,CA-NAME), and everyone that has the CA's public verification key can get convinces that U's public key is PK, without having ever seen or talked to U.

This is all very nice, but there are many annoying details that need to be handled. Some sample:

1. What does the CA verify when it issues the certificate?
2. Does the CA verify the identity of U using and "ID card" of some kind?
3. Does the CA verify that U "knows the secret key" associated with PK? (maybe U simply copies the PK of someone else?) and if so, then how?
4. For how long is the certificate valid?
5. How to "revoke" certificates? by time limits? by revocation lists? more mundane issues like: How to keep the signing key both accessible and secret? How to deal with high volume of traffic? (the standard solution - distribution and replication - is problematic here)

For more discussion on these and other issues, see the books by **Menezes van Oorschot Vanstone** and **Schneier**, both found in the course reference list.

8 Authenticated key exchange

One of the main uses of signatures, PKE and PKI is to setup pairwise secure communication channels between pairs of parties. There are many protocols that do it: SSL, SSH, TLS, IPSEC, etc etc. We have seen many ingredients of these protocols:

1. Key exchange that assumes authentic communication (DH)
2. A secure channels protocol that does not assume authentic communication
3. A way to bind identities to public keys in an untrusted environment (PKI)

But there is an ingredient missing: a way to establish pairwise secret keys over an untrusted environment. This is the task of AKE (authenticated key exchange).

AKE is a vast topic (can have an entire course only in AKE protocols and their analysis). We'll see just a glimpse:

How to define AKE? Two main approaches:

1. Via a game
2. a semantic definition (via ideal model) Definitions are "almost" equivalent, but there are several discrepancies.

-How to construct AKE? Two protocols for example:

8.1 ISO 9798-3 (SIG-DH)

The ISO 9798-3 protocol describe a key agreement mechanism based on the threepass authentication mechanism . The mechanism described involves three passes, and establishes a shared secret key between A and B with mutual entity authentication. This version of the three pass ISO 97983 mechanism requires two types of algorithms: the DiffieHellman key agreement scheme and a public key signature scheme. We assume the basic Diffie-Hellman setting with a prime p and exponentiation modulo p. In addition, it is assumed that each entity has a personal digital signature scheme (GEN, SIG, VER), consisting of a signing algorithm SIG and a signature verification algorithm VER . We also assume that each entity has access to authenticated copies of the public verification algorithm VER of all other entities. The key agreement protocol runs as follows:

1. A generates signature and verification keys using GEN algorithm. $(sk_{A,}vk_A) = GEN(1^n)$;
2. B generates signature and verification keys using GEN algorithm. $(sk_{B,}vk_B) = GEN(1^n)$;
3. A randomly and secretly generates $r_A \in 1, ..., p$, computes $g^{r_A}\ mod\ p$, constructs the key token $KT_{A1} = g^{r_A}\, mod\, p$ and sends it to B.
4. B randomly and secretly generates $r_B \in 1, ..., p$, computes $g^{r_B}\ mod\ p$, constructs the key token $KT_B = SIG_B(sk_B; g, g^{r_A}, A)$ and sends it back to A.

5. A verifies B's signature on the key token KT_B using B's public verification algorithm $VER_B(vk_B, g; g^{r_A}; KT_B)$, and checks that the received value g^{r_A} agrees with the one sent in step 3. If the check is successful, A proceeds to compute the shared key as $K_{AB} = (g^{r_B})^{r_A}$.

6. A constructs and sends to B the signed key token $KT_{A2} = SIG_A(sk_A, g^{r_A}; g^{r_B}; B)$

7. B verifies A's signature on the key token KT_{A2} , using A's public verification algorithm $VER_A(vk_A, g; g^{r_B}; KT_{A2})$, and checks that the received value g^{r_B} agrees with the one sent in step 4. If the check is successful, B proceeds to compute the shared key as $K_{AB} = (g^{r_A})^{r_B}$.

The generated key $g^{r_A r_B}$ is then the shared secret key between A and B.

8.2 The Needham-Schroeder-Lowe protocol (NSL)

The Needham-Schroeder (public-key) protocol was originally proposed by Needham and Schroeder in 1978. However, it was later broken by Lowe in 1995, who then proposed a fix and proved his fix correct. By the "Needham-Schroeder-Lowe" protocol, we mean the Needham-Schroeder protocol as fixed by Lowe. At its most basic form, this protocol consists of three messages between an initiator A and a responder B. Both parties are assumed to have the public key of the other party.

The protocol begins with the initiator A generating a new random- string symbol (denoted N_a for it's original designation as "A's nonce").

1. The first message consists of A's name and this string, encrypted in B's public key (K_B)

   $A \longmapsto B:\ ENC(K_B,\ |A|N_a|)$

2. B upon receiving this message, creates a random string N_b of its own. It then sends back its name, the received random string, and the new random string, all encrypted in A's public key:

   $B \longmapsto A:\ ENC(K_A,\ |B|N_a|N_b|)$

3. A, upon receiving this message, checks two things: that the first component is the name of the intended responder, and that second component is the random string that it recently created. If so, it re-encrypts the third component in B's public key:

   $A \longmapsto B:\ ENC(K_B,\ |N_b|)$

At this point, A terminates and signals a successful protocol execution. B will do the same upon receiving the third message iff the plain-text is its recently-generated random string.

It is unclear what formal security goals the protocol was originally proposed to fulfill: few formal, appropriate definitions existed in 1975. In the intervening years, however, consensus has settled upon the mutual authentication security goal. Informally speaking, this goal requires that A only successfully complete a run of the protocol with input B only if B has begun a run of the protocol with input A, and vice-versa. However, it has also been noted that the protocol could also achieve the key-exchange security goal. Very roughly, this protocol should also guarantee secrecy of the encrypted random strings, which can then be used for generating a symmetric key. This protocol have 2 versions. In the first version, the participants output the random string chosen by the initiator, and in the second they output the random string chosen by the responder. One of the two versions is secure and the other one isn't, and it is left as homework to figure out which is which.