Lecture 12
CCA, Digital Signatures

Instructor: Chris Peikert
Scribe: Rajiv Iyer

1 Chosen-Ciphertext Attacks

1.1 Motivation

In the preceding lectures, our notion of security for encryption was constrained to work against so-called *passive* adversaries. Such adversaries are permitted to see the ciphertexts for messages of their choice, and (in the public-key setting) to generate ciphertexts on their own. However, the adversary never gets to see the *decryptions* of any messages. For some applications, we will need a stronger definition in which the adversary gets (limited) access to the decryption machinery as well.

To establish the motivation for this stricter definition, imagine an auction scenario, where Alice and Eve are bidders for some item that Bob has for sale. Alice is the honest participant, while Eve is the malicious adversary who tries to outbid Alice by the smallest amount possible. Consider how it might work:

- Alice encrypts her bid b_A under a key k, and sends the resulting ciphertext c to Bob.
- Eve intercepts the ciphertext c and modifies some bit(s), dispatching the modified ciphertext c' to Bob. To Bob, c' looks like a ciphertext and is treated as such. Bob decrypts Alice's and Eve's ciphertexts, yielding bids b_A and b_E, and decides who has bid the highest.
- To the highest bidder, Bob sends a message to the effect of "Congratulations, you won with a bid of b!" where $b = b_A$ or $b = b_E$, whichever is larger.

Suppose that the encryption scheme used in this scenario allows Eve to modify the ciphertext c, so that the result c' encrypts a message that is related (in a predictable way) to b_A, *even though Eve does not know what b_A is*? For example, what if Eve can increase the underlying bid by 1? Then she will always win the auction with the lowest possible bid, *and* she will learn the value of Alice's bid!

It turns out that many encryption schemes are "malleable" in the above-described way. For example, consider our PRF-based scheme, where $c = (r, f_k(r) \oplus m)$. Flipping any bit of the second component causes the same bit in the underlying message to flip. So if Alice's bid happened to be even, Eve can increment it by 1 by flipping the least significant bit. (If Even suspects that Alice's bid is odd, she can flip other bits to increase it by 1 or more.)

The example illustrates some fine points:

- Eve is *generating* ciphertexts of her own, not just seeing ciphertexts that come from an encryption oracle. She is mounting a so-called "chosen-ciphertext attack" on the scheme.
- Bob is taking some action based on the result of $\mathsf{Dec}_k(c)$. In this case, when Bob sends his congratulations message, he is even acting as a *decryption oracle*!

These issues are not just philosophical curiosities. PKCS#1 is a family of public-key cryptographic standards implemented by web servers and browsers, smart cards, etc. In 1998, Bleichenbacher demonstrated a "chosen-ciphertext" attack against the PKCS#1 standard, which queried the decryption device (e.g., an SSL-equipped web server) on a large number of specially crafted ciphertexts, which gradually revealed information to the attacker. In practical terms, this meant that an SSL session key could be exposed in a reasonable amount of time, perhaps a day or less. (PKCS#1 has since been updated with a "chosen ciphertext-secure" protocol that withstands such attacks.)

1.2 Definition

We seek a very conservative definition that ensures security *even if the attacker is given access to a decryption oracle*. Of course, the attacker can learn the underlying message of any ciphertext simply by querying its decryption oracle, so how can we hope to achieve any meaningful notion of secrecy? The answer is that we slightly weaken the decryption oracle (in one of a couple of possible ways) to prevent the adversary from querying it on the challenge ciphertext.

Definition 1.1. We say that SKC is secure under *chosen-ciphertext attack* (IND-CCA-secure) if the following tuple of oracles is indistinguishable for $b = 0, 1$:

$$\langle k \leftarrow \mathsf{Gen} : \mathsf{Enc}_k(\cdot), \mathsf{D}_k(\cdot), \mathsf{C}_k^b(\cdot, \cdot) \rangle,$$

where $\mathsf{C}_k(m_0, m_1)$ is the usual challenge oracle that outputs $c^* = \mathsf{Enc}_k(m_b)$ and terminates, and $\mathsf{D}_k(c)$ is an oracle that outputs $\perp$ if $c = c^*$ (the challenge ciphertext), else it outputs $\mathsf{Dec}_k(c)$.

We say that SKC is IND-CCA1-secure (or secure under "lunchtime" attacks) if the above holds where D_k is a decryption oracle that answers queries *before* C_k is invoked, then terminates (i.e., it does not answer queries after the challenge ciphertext c^* has been produced).

("Lunchtime" security is weaker than full IND-CCA security. The name refers to the following whimsical scenario: Eve breaks into Bob's office while Bob is out to lunch, thus temporarily gaining access to his decryption machinery. She wants to formulate a sequence of queries so that later on, when Bob comes back from lunch and she no longer has a decryption oracle, she can violate the secrecy of encrypted messages sent to/from Bob.)

1.3 Achieving CCA Security

Our PRF-based scheme is not CCA-secure, as it is vulnerable to the bit-flipping attack described above.

To design a CCA-secure scheme, we wish to use the notion of *invalid ciphertexts*, i.e., we allow the decryption algorithm $\mathsf{Dec}_k(\cdot)$ to output $\perp$ on ciphertexts that don't look "well-formed." The intuition is to design the scheme so that the only way an adversary can obtain a well-formed ciphertext is by directly querying the encryption oracle. This way, it already "knows" the underlying message, hence the decryption oracle is "useless!" (This is just the intuition for how we think about designing a CCA-secure scheme.)

It can be shown that in the shared-key setting, combining a message authentication code (MAC) with an IND-CPA-secure in the right way yields Authenticated Encryption (AE), a concept from last lecture. Moreover, it can be proved that an Authenticated Encryption (AE) scheme is, in fact, IND-CCA-secure. (You will prove these facts in your homework.)

Question 1. Can textbook RSA ciphertexts be manipulated with predictable consequences to the underlying plaintexts (similar to the PRF-based scheme)? Recall that RSA encryption takes a plaintext $m \in \mathbb{Z}_n$ (where $n = pq$ for primes p, q) and returns $m^e \pmod{n}$.

Question 2. How does the "bit-flipping" trick for our PRF-based encryption scheme violate our precise definition of CCA security?

2 Digital Signatures

We now switch back to message authentication itself. Just as with encryption, a natural question is whether the tagging algorithm Tag and verification algorithm Ver must use the *same* key, or could they be *asymmetric*? That is, is it possible for verification to be performed by anyone (using a public verification key), while signing can be performed by only one party?

Digital signatures are the asymmetric analogue of MACs. For messages sent over an insecure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender.

2.1 Model

Very similarly to a MAC, a digital signature scheme SIG is a triple of algorithms:

- Gen: outputs a (public) verification key vk and a (secret) signing key sk.
- $\mathsf{Sign}_{sk}(m) = \mathsf{Sign}(sk, m)$: given a message $m \in \mathcal{M}$, outputs a signature σ.
- $\mathsf{Ver}_{vk}(m, \sigma) = \mathsf{Ver}(vk, m, \sigma)$: given a message m and signature σ, either accepts or rejects.

2.2 Definition

We seek a security definition in which the adversary (forger) may see signatures for messages that it selects arbitrarily (hence *chosen-message-attack* or CMA), but still cannot forge a signature for another message. Therefore, we give the adversary oracle access to $\mathsf{Sign}_{sk}(\cdot)$. Moreover, the forger should explicitly be given the signature verification key vk, because it is public information.

Definition 2.1. We say that SIG is *(strongly) unforgeable under chosen-message-attack* (UF-CMA-secure) if for every nuppt $\mathcal{F}$,

$$\mathbf{Adv}_{\text{SIG}}(\mathcal{F}) := \Pr_{(vk,sk)\leftarrow\mathsf{Gen}}\left[\mathcal{F}^{\mathsf{Sign}_{sk}(\cdot)}(vk) \text{ succeeds}\right] \leq \operatorname{negl}(n),$$

where "succeeds" means that $\mathcal{F}$ outputs (m', σ') such that $\mathsf{Ver}_{v_k}(m', \sigma') = 1$, and either

- *Standard* unforgeability: m' differs from all queries m_i to the Sign oracle;
- *Strong* unforgeability: (m', σ') differs from all query/response pairs (m_i, σ_i) of the Sign oracle.

As with a MAC, the definition of *strong* unforgeability corresponds to a looser notion of success, implying a stricter definition of security.

2.3 A (Failed) Proposal

We sketch out a scheme with the view of making it UF-CMA-secure, but which we will show fails to live up the definition. The scheme is based on a family of trapdoor one-way permutations. Recall that a TDP family is a family $\mathcal{F} = \{f_s : D_s \to D_s\}$ of one-way permutations, where the function sampler S outputs, along with $f_s \leftarrow \mathcal{F}$, a *trapdoor* t that makes it easy to compute f_s^{-1}. The intuition is that the signer can use the trapdoor to compute a signature by inverting f_s on the message, which the verifier can check by computing f_s in the "forward" direction.

Formally, we define the following triplet of algorithms:

- Gen: choose $(s, t) \leftarrow \mathsf{S}$. Output $vk = s$ and $sk = t$.
- $\mathsf{Sign}_t(m \in D_s)$: output $\sigma = f_s^{-1}(m)$.
- $\mathsf{Ver}_s(m, \sigma)$: accept if $f_s(\sigma) = m$, otherwise reject.

The scheme is correct by inspection. And f_s is "hard to invert" by assumption, so why isn't the scheme secure according to Definition 2.1? The key point is that f_s is hard to invert on a *random* value $y \in D_s$, but a forger gets to *choose* the message on which it inverts f_s! In fact, observe that there is a trivial attack on the above scheme (which doesn't even make use of a signing oracle): given $vk = s$, choose some $\sigma \in D_s$, let $m = f_s(\sigma)$, and output (m, σ) as a forgery. Clearly, Ver_s accepts this pair!

Designing a truly UF-CMA-secure signature scheme is quite tricky, and involves several steps. We will see how to do so in the next lecture.

Answers

Question 1. Can textbook RSA ciphertexts be manipulated with predictable consequences to the underlying plaintexts (similar to the PRF-based scheme)? Recall that RSA encryption takes a plaintext $m \in \mathbb{Z}_n$ (where $n = pq$ for primes p, q) and returns $m^e \pmod{n}$.

Answer. Yes. Multiplying a ciphertext that encrypts a message m by x^e yields a ciphertext that encrypts $m \cdot x$:

$$\mathsf{Enc}(m) \cdot x^e \equiv m^e \cdot x^e \equiv (mx)^e \equiv \mathsf{Enc}(mx).$$

Question 2. How does the "bit-flipping" trick for our PRF-based encryption scheme violate our precise definition of CCA security?

Answer. Let m be any $n - 1$ bitstring. On input k an attacker simply queries $c^* \leftarrow C_k^b(1\|m, 0\|m)$, then queries the decryption oracle on $\overline{c^*}$ and outputs the first bit of the resulting plaintext. Note that $\overline{c^*} \neq c^*$ so the decryption oracle does not output $\perp$. Since $D_k(\overline{c^*} = \overline{D_k(c^*)})$ by the "bit-flipping" trick, the first bit of the resulting plaintext will be 0 if c^* is an encryption of $1\|m$ and 1 otherwise.