Bad News
Chosen-Ciphertext Attacks

Foundations of Cryptography
Computer Science Department
Wellesley College

Fall 2016

Table of contents

Ought oh!

- We have defined security against two types of attacks: *passive eavesdropping* and *chosen-plaintext* attacks.
- A *chosen-ciphertext attack* is even more powerful.
- The adversary can obtain both encryptions of messages of its choice (as before) and decryptions of ciphertexts of its choice.

Security against chosen-ciphertext attacks (CCA)

The experiment is defined for any private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, any adversary $\mathcal{A}$, and any value n for the security parameter: The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$:

1. A key k is generated by running $\mathsf{Gen}(1^n)$.
2. The adversary $\mathcal{A}$ is given 1^n and oracle access to $\mathsf{Enc}_k(\cdot)$ and $\mathsf{Dec}_k(\cdot)$. It outputs a pair of messages $m_0, m_1 \in \mathcal{M}$ of the same length.
3. A random bit $b \leftarrow \{0,1\}$ is chosen. A *challenge ciphertext* $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_k(\cdot)$ and $\mathsf{Dec}_k(\cdot)$, but is not allowed to query the latter on the challenge ciphertext. Eventually $\mathcal{A}$ outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. We write $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1$ if the output is 1 and in this case we say that $\mathcal{A}$ *succeeded*.

*CCA-secure**

Definition 3.33. A private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ had *indistinguishable encryption under a chosen-ciphertext attack* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that

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

where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used by the experiment (for choosing the key, the random bit b, and any random coins used in the encryption process).

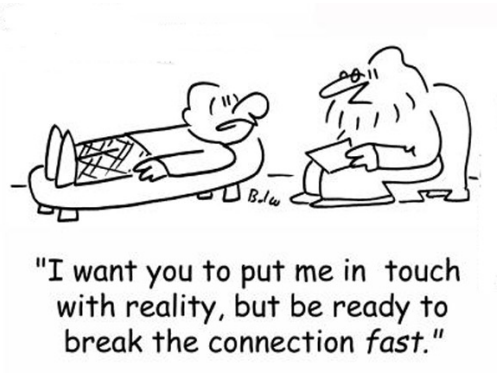
*The adversary's access to the decryption oracle is unlimited except for the restriction that it may not request decryption of the challenge ciphertext itself. It is however, allowed to come as close as it wishes.

Indistinguishability under multiple encryptions

- The natural analogue of Theorem 3.24 holds for CCA-security as well.
- That is, if a scheme has indistinguishable encryption under a chosen-ciphertext attack then it has indistinguishable multiple encryptions under a chosen-ciphertext attack.

Reality check

- But wait a minute, how realistic is it to assume that our adversary has such power?
- I mean with a little care can't we avoid such pedantic nonsense.*



"This is the kind of arrant pedantry up with which I will not put." – Winston Churchill

*Insecurity of the schemes we have studied**

Very bad news. None of the schemes we have studied are CCA-secure.

For example. Consider Construction 3.30, where encryption is carried out as $\mathrm{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$.

An adversary $\mathcal{A}$ running in the CCA indistinguishability experiment can choose $m_0 = 0^n$ and $m_1 = 1^n$. Upon receiving $c = \langle r, s \rangle$, the adversary can flip the first bit of s and ask for a decryption of the resulting ciphertext c'. Since $c' \neq c$ this is allowed, and the decryption oracle answers with either 10^{n-1} (in which case $b = 0$) or 01^{n-1} (in which case $b = 1$).

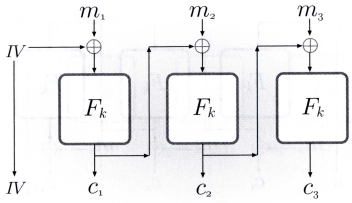
*Any encryption scheme that allows ciphertext to be manipulated in any "logical way" cannot be CCA-secure. More soon.

A less contrived example

- We show a chosen-ciphertext attack on a natural and widely used encryption scheme.
- The attack only requires the ability to determine whether or not a modified ciphertext decrypts correctly.
- This is easy to obtain. For example, a server might request a retransmission or terminate a session if it receives a ciphertext that does not decrypt correctly.

PKCS #5 padding

- Since CBC mode requires the length of the plaintext be multiple of the block length, the *message* must be padded before encryption resulting in *encoded data*.



- *PKCS #5* is a popular approach: Let L be the block length and b the number of bytes needed. Append to the message a string containing the integer b represented in 1 byte repeated b times.
- The receiver recovers the encoded data and checks that the last byte b is repeated b times. If okay, the padding is then removed. A *"bad padding"* error is generated otherwise.

The attack*

- Let IV, c_1, c_2 be a ciphertext observed by attacker, and let m_1, m_2 be the underlying encoded data.
- Note that
$$m_2 = F_k^{-1}(c_2) \oplus c_1,$$
where k is the key. The second block m_2 ends in $\underbrace{0xb \ldots 0xb}_{b \text{ times}}$.
- Let c_1' be identical to c_1 except in the last byte. The decryption of the ciphertext IV, c_1', c_2 is m_1', m_2' where $m_2' = F_k^{-1}(c_2) \oplus c_1'$.
- Note that m_2' is identical to m_2 except for a change in its final byte.

*We describe the attack on a 3-block ciphertext for simplicity.

More generally

- Similarly if c_1' is the same as c_1 except for its ith byte, then decryption of IV, c_1', c_2 will result in m_1', m_2' where m_2' is the same as m_2 except for a change in its ith byte.
- If $c_1' = c_1 \oplus \triangle$ for any string $\triangle$, then decryption of IV, c_1', c_2 yields m_1', m_2' where $m_2' = m_2 \oplus \triangle$
- The attacker begins by modifying the first byte of c_1. If decryption fails, then the the receiver is checking all L bytes of m_2' and therefore $b = L$.
- Otherwise, the attacker learns that $b < L$ and proceeds to change the second byte, and so on.
- The left-most modified byte for which decryption fails reveals the left-most byte being checked by the receiver, and so reveals b.

The attacker moves on

- We show how the attacker can learn the final byte of the message, say B. The attacker knows that m_2 ends in $0xB$ $0xb$ $0xb \ldots 0xb$ (with $0xb$ repeated b times).
- For $0 \leq i < 2^8$, define

$$\triangle_i \stackrel{\text{def}}{=} 0x00 \ldots 0x00 \; 0xi \overbrace{0x(b+1) \; 0x(b+1)}^{b \text{ times}}$$
$$\oplus \; 0x00 \ldots \; 0x00 \overbrace{0xb \ldots 0xb}^{b \text{ times}}.$$

- If ciphertext $IV, c_1 \oplus \triangle_i, c_2$ is sent, then the encoded data will end in $0x(B \oplus i) \; 0x(b+1) \ldots 0x(b+1)$ with $0x(b+1)$ repeated b times.
- Decryption fails unless $0x(B \oplus i) = 0x(b+1)$. The attacker tries all 2^8 values for $\triangle_i$ until ones succeeds at which point she knows $B = 0x(b+1) \oplus i$. You take it from here ...

CAPTCHAs

- A CAPTCHA is a distorted image of, say, an English word that is easy for humans to read, but hard for a computer to process. CAPTCHAs are used to ensure that a human user (not automated software) is interacting with a webpage.

- When a user $\mathcal{U}$ loads a webpage served by $\mathcal{S}_W$, $\mathcal{S}_W$ encrypts a random English word w using a key k shared between $\mathcal{S}_W$ and the CAPTCHA server $\mathcal{S}_C$ and sends the ciphertext c to $\mathcal{U}$ who forwards it to $\mathcal{S}_C$.
- $\mathcal{S}_C$ decrypts c and sends a distorted image of w to $\mathcal{U}$. Finally $\mathcal{U}$ sends the word w back to $\mathcal{S}_W$ for verification.

A padding-oracle attack on CAPTCHAs

- $\mathcal{S}_C$ will issue a "bad padding" error if decryption fails on ciphertext it received from $\mathcal{U}$.
- So $\mathcal{U}$ (who is robot after all) can carry out a padding-oracle attack and solve the CAPTACHA without human involvement, rendering CAPTCHA ineffective.