Problem Set 1
Eamonn W. Postlethwaite

Cryptography, 6CCS3CIS
September 29, 2025

Problem 1. *On integrity.*

We formalise the sketch of the cryptographic approach[1] for integrity given in the first recorded lecture. Recall, a message authentication code (MAC) is a triple of algorithms $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vf})$ where

- the algorithm KGen outputs an integrity key ik,
- on input an integrity key ik and a message m the algorithm $\mathsf{Tag}(ik, m)$ outputs a tag τ,
- on input an integrity key ik, a message m and a tag τ the algorithm $\mathsf{Vf}(ik, m, \tau)$ outputs either 0 or 1.

In the above KGen denotes 'key generation' and Vf denotes 'verify'. The intuition is that a communicating pair (Alice and Bob) share ik. When Alice wishes to send a message m to Bob she creates a tag $\tau = \mathsf{Tag}(ik, m)$ and sends (m, τ). Bob then checks $\mathsf{Vf}(ik, m, \tau) = 1$. If Vf outputs 1 then we say τ is a valid tag for m under ik, else it is invalid.

i. What should correctness mean for a MAC?

You may wish to consider the definition of correctness given for SKE in the first recorded lecture.

ii. Design a correct MAC. It does not have to satisfy any security properties.

iii. In the first recorded lecture we informally used a MAC to ensure the integrity of ciphertexts. How would we achieve this in the syntax of $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vf})$ given above?

We are going to consider how to define a security property for a MAC. Assume the MAC is correct, as per your answer to (*i*.), and that some ik has been output by KGen. We say an adversary can 'forge a tag for message m' if it can create a valid tag on m without knowing ik, that is if it can create some τ such that $\mathsf{Vf}(ik, m, \tau) = 1$. Here is a list of potential definitions for a security property.

- (D1) An adversary should not be able to recover ik by observing (m_i, τ_i) such that $\tau_i = \mathsf{Tag}(ik, m_i)$.
- (D2) Let an adversary have access to $\mathsf{Tag}(ik, \cdot)$. The adversary should not be able to forge a tag on any message m.
- (D3) Let an adversary have access to $\mathsf{Tag}(ik, \cdot)$ and Q be the set of messages this adversary has input into $\mathsf{Tag}(ik, \cdot)$. The adversary should not be able to forge a tag on any message not in Q.

The notation $\mathsf{Tag}(ik, \cdot)$ means the adversary has a black-box function which takes as input a message m and outputs the tag $\mathsf{Tag}(ik, m)$. The adversary cannot 'see inside' this function, i.e. does not know what ik is.

Recall that a definition for a security property comprises two elements; an action that should not be possible for an adversary ($*$) and a description of the abilities of an adversary ($\dagger$).

iv. What are ($*$) and ($\dagger$) for each of (D1)–(D3)?

To achieve a definition an object (here a MAC) that satisfies it must be shown to exist. To show a definition cannot be achieved it must be shown that no such object exists.

One definition is at least as strong as another if it is at least as hard to achieve. One definition is stronger than another if it is harder to achieve. That is, (DX) is at least as strong as (DY) whenever every MAC satisfying (DX) also satisfies (DY). Furthermore, (DX) is stronger than (DY) if it is at least as strong as (DY) and there exists a MAC satisfying (DY) that does not satisfy (DX).

v. What is the strongest definition of (D1)–(D3)? Can it be achieved?

vi. Which definition from (D1)–(D3) should we select as a security definition of a MAC? Why?

vii. Can you think of a definition that is at least as strong as your answer to (*vi*.) but still achievable? (Hint: Tag can be randomised, i.e. on a given message may output multiple valid tags. Consider (D3).)

[1] Throughout all objects that a communicating pair should keep secret are boxed like this. Generally these are 'keys'.

We did not give the adversary access to Vf in any of our potential definitions. The next question examines whether this is an important omission.

viii. Assume that

- $\mathsf{Vf}(ik, m, \tau)$ computes $\tau' = \mathsf{Tag}(ik, m)$ and returns 1 if $\tau = \tau'$ and 0 otherwise, and
- $\mathsf{Tag}(ik, \cdot)$ is deterministic, i.e. on a given message always outputs the same tag.

This is called canonical verification for a deterministic MAC.

Given these assumptions, is an adversary with access to $\mathsf{Tag}(ik, \cdot)$ made more powerful by giving it access to $\mathsf{Vf}(ik, \cdot, \cdot)$? (Hint: use an adversary's access to $\mathsf{Tag}(ik, \cdot)$ to 'build' canonical verification.)

Problem 2. *On encryption.*

Recall, a (secret key) encryption scheme (SKE) is a triple of algorithms $\mathsf{SKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ where

- the algorithm KGen outputs a secret key sk,
- on input a secret key sk and a message m the algorithm $\mathsf{Enc}(sk, m)$ outputs a ciphertext c,
- on input a secret key sk and a ciphertext c the algorithm $\mathsf{Dec}(sk, c)$ outputs a message m.

In the above Enc denotes 'encrypt' and Dec denotes 'decrypt'. Correctness and informal security definitions for a SKE are discussed in the first recorded lecture. In the below assume a correct SKE and that some sk has been output by KGen.

i. Assume an adversary has access to $\mathsf{Enc}(sk, \cdot)$. Describe a scenario where Enc being deterministic, i.e. on a given message always outputs the same ciphertext, is undesirable. (Hint: consider scenarios where you might often expect to see similar and stereotyped messages.)

Now also consider a MAC that is 'perfect'. (This is not a formal definition, merely meant to imply that no adversary can forge under any circumstances.) We will discuss how the following approach may be used to counter replay, reordering and reflection attacks, as introduced in the first recorded lecture.

To send a message m given (sk, ik)

- let $c = \mathsf{Enc}(sk, m)$ and $\tau = \mathsf{Tag}(ik, c)$,
- send $\tilde{c} = (c, \tau)$.

To receive a message $\tilde{c} = (c, \tau)$ given (sk, ik)

- check that $\mathsf{Vf}(ik, c, \tau) = 1$ and if not abort,
- if there was no abort, let $m = \mathsf{Dec}(sk, c)$.

ii. How might you add information to the message m to prevent replay, reordering and reflection attacks? (Hint: consider information that unambiguously defines 'freshness', 'order' and 'direction', respectively.)

iii. If the adversary is able to selectively deny delivery of some $\tilde{c}$, how does this affect your answer to (*ii.*)?

Problem 3. *On cryptography as munitions.*

i. Read either of the articles cited in the intermission of the first recorded lecture.

ii. Write (pen/pencil and paper) your thoughts about the article you have read. Try for a side of A4. Are you surprised how quickly 'purely technical' objects, like cryptography and source code, become socially and legally vexed?