Message Authentication Codes

Lecture 20: From pseudorandom functions

Lecture overview

A random function gives us a clean authentication proof. A pseudorandom function makes the construction efficient. We will prove the connection, then see why the message length is part of the security guarantee.

One reduction, one extension attack, and ways to repair the domain.

Based on the instructor's Lecture 20 [Maj26].

Abstract and lecture roadmap

Last time, a fresh entry in a secret random table still had to be guessed. Today we replace that enormous table with a short secret key.

1. Construct a MAC from a pseudorandom function (PRF).
2. Turn a successful forger into a PRF distinguisher.
3. Extend the GGM tree naively, and break authentication.
4. Fix the permitted message domain and discuss group endorsements.

Proof roadmap

Compare the real PRF with a random function. In the random world, use the fresh-entry argument from Lecture 19. The difference between the worlds is exactly what PRF security controls.

[BS23, §6.3, Thm. 6.2, pp. 224–225].

Main results

Fix a message length n and a key and tag length λ. Let $F_K \colon \{0,1\}^n \to \{0,1\}^\lambda$ be an efficiently computable keyed family.

PRFs give MACs

Tag m with $F_K(m)$ and verify by recomputing it. For any efficient attacker, one fresh-message forgery succeeds with probability at most

$$2^{-\lambda} + \text{the advantage of a corresponding PRF distinguisher.}$$

The distinguisher uses at most one more oracle query than the forger.

The message domain matters

If the same GGM tree accepts paths of different lengths, one observed tag reveals valid tags for all longer paths extending it. The attack succeeds with probability 1.

[BS23, Thm. 6.2, p. 224; §4.6.1, p. 152]; source slides, pp. 3–6 [Maj26].

From a secret table to a secret key

Lecture 19: random function

$R\colon \{0,1\}^n \to \{0,1\}^\lambda$

Store 2^n independent tags.

Secret storage: $\lambda 2^n$ bits.

An unqueried entry is uniform, even against unlimited computation.

Lecture 20: pseudorandom function

$F_K\colon \{0,1\}^n \to \{0,1\}^\lambda$

Evaluate a public algorithm using K.

Secret storage: λ bits here.

Efficient oracle algorithms cannot reliably distinguish it from R.

The algorithms are public. The sampled key K is secret.

[BS23, §4.4, pp. 131–134; §6.3, pp. 223–225].

Parameters: what each length means

Symbol	Meaning
λ	Security parameter, key length, and tag length in this construction.
n	Message length, fixed for the instance and polynomially bounded in λ.
$K \in \{0,1\}^{\lambda}$	Uniform secret key shared by the sender and verifier.
q	Maximum number of adaptive tagging queries.

The parameters in the source slides

Set $n = 100\lambda$. Then $\lambda = n/100$, so the key and tag are much shorter than the message. For example, $n = 12{,}800$ and $\lambda = 128$.

The factor 100 is only an example. The reduction works for any permitted n.

Source parameters: p. 3 [Maj26]. Asymptotic conventions: [BS23, §2.3.1, pp. 28–29].

PRF security: two possible oracles

An oracle algorithm D can choose inputs, receive outputs, and let later inputs depend on earlier answers.

Real world

Sample $K \xleftarrow{\$} \{0,1\}^\lambda$.

Answer each query m with $F_K(m)$.

Random world

Sample a uniform function R.

Answer each query m with $R(m)$.

$$\mathrm{Adv}^{\mathrm{prf}}_F(D) = \left|\Pr[D^{F_K} = 1] - \Pr[D^R = 1]\right|.$$

A secure PRF makes this quantity negligible in λ for every efficient D. Repeated inputs receive the same answer in both worlds.

[BS23, Game 4.2 and Def. 4.2, pp. 132–133].

The MAC: evaluate the PRF

Key generation

Sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and give it to the sender and verifier.

Tagging an n-bit message

For $m \in \{0,1\}^n$, output $\tau = \mathsf{Tag}_K(m) := F_K(m)$.

Verification includes the domain check

On $(\widetilde{m}, \widetilde{\tau})$, reject unless $|\widetilde{m}| = n$ and $|\widetilde{\tau}| = \lambda$. Otherwise accept exactly when $\widetilde{\tau} = F_K(\widetilde{m})$.

Correctness. Every honest tag is accepted because the verifier recomputes the same deterministic function value.

[BS23, §6.3, construction preceding Thm. 6.2, pp. 223–224]; source slide 3 [Maj26].

What the attacker must accomplish

1. The challenger samples $K \xleftarrow{\$} \{0,1\}^\lambda$.
2. The attacker A makes at most q tagging queries $m_1, \ldots, m_q \in \{0,1\}^n$. Each answer is $F_K(m_i)$.
3. A outputs one candidate (m^*, τ^*).

Let $\mathcal{Q}$ be the set of messages queried. The attacker wins exactly when

$$|m^*| = n, \qquad m^* \notin \mathcal{Q}, \qquad \tau^* = F_K(m^*).$$

Freshness concerns the message

The attacker may choose messages adaptively. The forged tag may equal a previously seen tag. Replaying a previously queried message does not win this experiment.

Write $\mathrm{Adv}^{\mathrm{mac}}_F(A)$ for this winning probability. There is one final verification attempt.

[BS23, Game 6.1, pp. 218–219]. Unique accepted tags make fresh-pair and fresh-message security coincide here.

The security theorem

PRF-to-MAC reduction

For every forger A making at most q tagging queries, there is an oracle algorithm D making at most $q + 1$ queries such that

$$\boxed{\mathsf{Adv}^{\mathsf{mac}}_F(A) \leq \mathsf{Adv}^{\mathsf{prf}}_F(D) + 2^{-\lambda}.}$$

D runs A once, with polynomial additional work.

How the proof will work

Give A tags from D's oracle. Use the same oracle to check the proposed forgery. A successful fresh-message guess is rare in the random world, so frequent success would reveal the real world.

[BS23, Thm. 6.2 and Eq. (6.6), pp. 224–225].

The distinguisher runs the forger

D receives an oracle $O\colon \{0,1\}^n \to \{0,1\}^\lambda$ and runs A.

Answering a tagging query m

Query $O(m)$, return the answer to A, and record m in $\mathcal{Q}$.

Checking A's final candidate (m^*, τ^*)

1. If $|m^*| \neq n$, $|\tau^*| \neq \lambda$, or $m^* \in \mathcal{Q}$, output 0.
2. Otherwise query $O(m^*)$.
3. Output 1 exactly when $\tau^* = O(m^*)$.

There are at most q queries for the simulation and one for the final check. D never needs the secret key.

Reduction in the proof of [BS23, Thm. 6.2, pp. 224–225].

The real world is the actual MAC game

When $O = F_K$

Every tagging answer is $F_K(m)$, exactly as in the real attack.

The final test accepts exactly the winning forgeries.

$$\Pr[D^{F_K} = 1] = \mathrm{Adv}^{\mathrm{mac}}_F(A).$$

This remains true when A chooses each query after seeing earlier tags. The entire interaction has the same distribution.

We have translated the forgery event into the distinguisher's output bit. Next we evaluate that same bit when the oracle is random.

[BS23, proof of Thm. 6.2, pp. 224–225].

The random world: a fresh row stays hidden

Imagine filling the table for R only when an input is first queried. A new row receives an independent uniform λ-bit string.

Fix A's random choices and a possible transcript H of all its tagging queries and answers. This fixes its proposed (m^*, τ^*).

If the candidate is valid and fresh

The row $R(m^*)$ has not been exposed. Thus

$$\Pr[R(m^*) = \tau^* \mid H,\ A\text{'s random choices}] = 2^{-\lambda}.$$

If the candidate is invalid or repeats a queried message, D outputs 0.

Averaging over transcripts and random choices gives $\Pr[D^R = 1] \leq 2^{-\lambda}$.

Random tables: [BS23, §4.4.2, pp. 133–134]. Conditioning: [HPS14, §5.3.2, pp. 233–235].

Putting the two worlds together

$$\begin{aligned}\mathrm{Adv}^{\mathrm{mac}}_{F}(A) &= \Pr[D^{F_K} = 1] \\ &\leq \left|\Pr[D^{F_K} = 1] - \Pr[D^{R} = 1]\right| + \Pr[D^{R} = 1] \\ &\leq \mathrm{Adv}^{\mathrm{prf}}_{F}(D) + 2^{-\lambda}.\end{aligned}$$

Why adaptivity causes no gap

The reduction answers every query using the same oracle. In the random world, conditioning on the interaction fixes only exposed rows. The final message must select an unexposed row.

This proves the theorem. Secure PRF evaluation and a long enough tag give negligible forgery probability.

[BS23, Thm. 6.2, Eq. (6.6), pp. 224–225].

What the bound says

One attempt, after many tagging queries

$$\Pr[\text{forge}] \leq \mathrm{Adv}^{\mathrm{prf}}_{F}(D) + 2^{-\lambda}.$$

There is no factor q in the guessing term. There is only one final fresh tag to guess. The PRF advantage does depend on D's resources, including its $q + 1$ queries.

- A uniform 128-bit tag is guessed with probability 2^{-128}.
- If tag length were fixed independently of λ, its guessing probability would not be asymptotically negligible.
- Multiple verification attempts require a security experiment and bound that account for those attempts.

[BS23, §6.2, pp. 220–223; Thm. 6.2, p. 224].

Recall GGM: each bit chooses a child

Let $G\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a secure pseudorandom generator. Split its output into two λ-bit halves:

$$G(s) = G_0(s) \,\|\, G_1(s).$$

For $m = m_1 \cdots m_n$, start at $s_0 = K$ and follow the path

$$s_j = G_{m_j}(s_{j-1}) \quad (1 \le j \le n), \qquad F_K(m) = s_n.$$

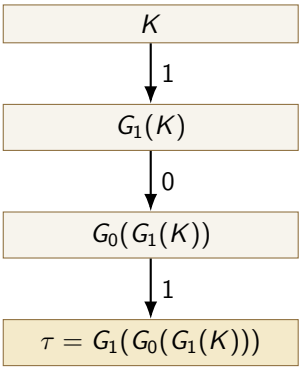
The fixed-depth GGM theorem

For polynomially bounded n, these functions form a secure PRF on $\{0,1\}^n$, assuming G is a secure PRG.

Evaluation takes n calls to G. The root is secret, but G_0 and G_1 are public algorithms.

[BS23, §4.6, Thm. 4.10, pp. 147–151]; original construction [GGM86].

A tag is a tree state



Example: the path 101

The tag reveals the state at the end of this path.

What can anyone compute next?

From τ, anyone can compute $G_0(\tau)$ and $G_1(\tau)$.

These are the states for paths 1010 and 1011 in the same tree.

At a fixed depth, those descendants are outside the allowed message domain.

Tree viewpoint and extension attack: [BS23, §4.6.1, pp. 151–152].

The dangerous change: accept every depth

Suppose we modify the verifier to accept arbitrary nonempty bit strings using the same root key. Write the variable-depth evaluation as

$$\mathrm{Tree}_K(m) = G_{m_{|m|}}\big(\cdots G_{m_2}(G_{m_1}(K))\cdots\big).$$

One-query attack

1. Ask for the tag $\tau = \mathrm{Tree}_K(m)$ on any nonempty m.
2. Compute $\tau' = G_0(\tau)$ using the public generator.
3. Output $(m0, \tau')$.

The new message is fresh, and

$$\mathrm{Tree}_K(m0) = G_0(\mathrm{Tree}_K(m)) = G_0(\tau) = \tau'.$$

The modified verifier accepts with probability 1.

Source slides, pp. 5–6 [Maj26]; [BS23, §4.6.1, p. 152].

One exposed state reveals its whole subtree

More generally, after seeing (m, τ), take any nonempty suffix $u = u_1 \cdots u_r$ and compute

$$\tau_u = G_{u_r}\big(\cdots G_{u_2}(G_{u_1}(\tau))\cdots\big).$$

Then $\tau_u = \mathrm{Tree}_K(mu)$, so (mu, τ_u) is a valid forgery.

What failed?

The tag exposed a state from which the public algorithm can continue to another accepted message. The attacker does not recover K, invert the PRG, or distinguish its output from uniform.

The fixed-length theorem still applies

The original verifier rejects mu because $|mu| > n$. Changing the accepted domain changes the construction that needs to be proved secure.

[BS23, §4.6.1, pp. 151–152]; source slide 6 [Maj26].

The source repair: a separate key for each length

For each supported length ℓ, establish an independent secret key K_ℓ. Use that key only for ℓ-bit messages.

Message length	Tagging rule
1024 bits	$\mathrm{Tag}(m) = F^{(1024)}_{K_{1024}}(m)$
2048 bits	$\mathrm{Tag}(m) = F^{(2048)}_{K_{2048}}(m)$

Verification chooses the key from the actual message

Compute $\ell = |m|$, reject unsupported lengths, and verify using K_ℓ. Do not let an unchecked length field choose a different key.

Extending a tag stays in the old tree. Messages of the new length use an independent root.

Source slide 7 [Maj26]. Fixed-domain security: [BS23, Thm. 6.2, p. 224].

Why independent keys repair the problem

Suppose L message lengths are supported, where L and all lengths are polynomially bounded in λ.

Proof roadmap: replace one length at a time

Replace the PRF for each length by an independent random function. Each replacement changes the forgery probability by at most the advantage ε_ℓ of the corresponding PRF distinguisher.

Once all replacements are complete, any fresh message selects an unexposed uniform row in the table for its length. Therefore

$$\Pr[\text{forge}] \leq \sum_{\ell \text{ supported}} \varepsilon_\ell + 2^{-\lambda}.$$

For GGM with polynomially bounded lengths and resources, all these advantages have a common negligible bound $\varepsilon(\lambda)$. Their sum is at most $L\varepsilon(\lambda)$, which is negligible.

Hybrid proof using [BS23, Def. 4.2, p. 133; Thm. 6.2, pp. 224–225].

Optional: valid paths can be prefix-free

A set of strings is **prefix-free** if no string in the set is a proper prefix of another. Distinct valid paths then never end at an ancestor of one another.

A simple encoding

For a bit string m, define $E(m) = 1^{|m|}0m$.

$$E(0) = 100, \qquad E(1) = 101, \qquad E(10) = 11010.$$

The encoded length is $2|m| + 1$.

Why prefix-free? If $|m| < |m'|$, position $|m| + 1$ is 0 in $E(m)$ and 1 in $E(m')$. If the lengths agree, the encodings have equal length and are distinct whenever the messages are distinct.

Prefix-free GGM: [BS23, §4.6.1, Thm. 4.11, pp. 152–154]. Encoding proof shown here.

Optional: a single-root construction with encoding

Authenticate the encoded path

Set $\mathrm{Tag}_K(m) = \mathrm{Tree}_K(E(m))$, where $E(m) = 1^{|m|}0m$. The verifier recomputes $E(m)$ from the received message and checks the same tree value.

The theorem needed for this repair

GGM is a secure PRF when all queried paths belong to a prefix-free set, assuming the underlying PRG is secure. This holds for polynomially many queries of polynomially bounded length.

Apply that theorem to the tagging queries *and* the final verification query. All paths are in $E(\{0,1\}^*)$, so the earlier MAC reduction gives the same guessing term $2^{-\lambda}$.

Separate keys per length are one sufficient repair. This encoding gives another, with a single root key.

[BS23, Thm. 4.11, pp. 152–154; Thm. 6.8, p. 235].

Food for thought: several parties endorse a message

There are N parties and one verifier. We want the verifier to accept only after at least t distinct parties endorse the same message m.

Start with a simple target

Allow the endorsement to contain several tags. How could each party authenticate its own approval? What must the verifier count?

- What stops one party from being counted twice?
- What happens if different tags concern different messages?
- Can the verifier itself generate endorsements?

First solve the problem with a list of tags. Requiring one short combined tag is an additional goal.

Source exercise: p. 8 [Maj26]. MAC/signature distinction: [HPS14, §4.1, pp. 193–195].

Discussion answer: independent tags from distinct parties

Party i shares an independent MAC key K_i with the verifier. Assume a secure MAC for the agreed message domain.

An endorsement certificate

Collect (i, τ_i) for $i \in S$, where $|S| = t$ and $\tau_i = \mathrm{Tag}_{K_i}(m)$. The verifier checks that there are exactly t distinct registered identities and every listed tag verifies on the *same* received message m.

Security intuition against an outsider or a small coalition

If fewer than t parties have authorized m (counting corrupted parties as authorizing it), an accepted list needs a fresh-message forgery under some honest party's key.

This uses t tags. The verifier knows every key and can generate the whole list, so it is not public proof of the parties' endorsements.

Exercise solution. Shared-key MACs: [BS23, §6.1, pp. 217–219]; signatures: [HPS14, §4.1].

Questions to check the argument

1. Why does the PRF distinguisher need one extra query?
   It checks the candidate tag at the fresh message.

2. Why is random-function forgery not impossible?
   A uniform tag can still be guessed with probability $2^{-\lambda}$.

3. Does $(m0, G_0(\tau))$ break the fixed-length MAC?
   No. That verifier rejects the longer message.

4. Why is adding the bit 0 at the end of every message not enough?
   The encoding $m \mapsto m0$ is not prefix-free: 00 is a prefix of 000.

[BS23, Thm. 6.2, pp. 224–225; §4.6.1, pp. 151–154].

The ideas to carry forward

The reduction

A successful PRF-based MAC forgery becomes a successful test for the PRF oracle. The random world contributes only the chance of guessing a fresh tag.

$$\Pr[\text{forge}] \leq \mathrm{Adv}^{\mathrm{prf}}_{F}(D) + 2^{-\lambda}.$$

The domain

A proof for n-bit messages guarantees security on that domain. Accepting more inputs requires a new argument. With raw GGM tree states, accepting extensions gives an immediate forgery.

The source repair separates keys by message length. Prefix-free encoding offers another route, justified by a stronger GGM theorem.

[BS23, Thms. 4.10, 4.11, 6.2, 6.8].

Reading guide

Boneh–Shoup, version 0.6

- §6.3, Thm. 6.2, pp. 223–225: the PRF-to-MAC construction and proof.
- §4.6, Thm. 4.10, pp. 147–151: fixed-depth GGM.
- §4.6.1, pp. 151–154: the extension attack and prefix-free security.
- Optional: §6.6, Thm. 6.8, p. 235, for the encoding principle.

Hoffstein–Pipher–Silverman, second edition

§5.3.1–5.3.2, pp. 228–235: probability and conditional probability, used in the random-table argument.

§4.1, pp. 193–195: digital signatures, for comparison with the shared-key endorsement exercise.

[BS23, HPS14]. All page numbers refer to printed book pages.

References

[BS23] Dan Boneh and Victor Shoup.
A graduate course in applied cryptography, 2023.
Version 0.6, January 2023.
URL: https://toc.cryptobook.us/book.pdf.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali.
How to construct random functions.
Journal of the ACM, 33(4):792–807, 1986.
doi:10.1145/6490.6503.

[HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman.
An Introduction to Mathematical Cryptography.
Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.
doi:10.1007/978-1-4939-1711-2.

[Maj26] Hemanta K. Maji.
Lecture 20: Message authentication codes from PRF.
CS355, Purdue University, instructor-provided lecture slides, 2026.
Fall 2026. Source file: 20.pdf, 8 slides.

Technical inputs: PRF security and random functions

Assumption for the main construction

For every efficient oracle algorithm D, $\mathrm{Adv}^{\mathrm{prf}}_F(D)$ is negligible in λ. Inputs have the specified fixed length, and output length is λ.

Precise reference: Boneh–Shoup, v0.6, Game 4.2 and Def. 4.2, pp. 132–133. The MAC reduction is Thm. 6.2, Eq. (6.6), pp. 224–225, reproved here.

Random-function fact and conditioning

Unexposed entries of a uniform random function remain independent and uniform after the exposed entries are fixed. See Boneh–Shoup, §4.4.2, pp. 133–134.

Averaging conditional probabilities uses the law of total probability. Background: Hoffstein–Pipher–Silverman, 2nd ed., §5.3.1–5.3.2, pp. 228–235. The fresh-entry application is proved in the lecture.

Technical inputs: the GGM theorems

Fixed-depth GGM

Let $G\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be an efficient secure PRG, with halves G_0, G_1. For polynomially bounded n, the family

$$F_K(m_1 \cdots m_n) = G_{m_n} \circ \cdots \circ G_{m_1}(K)$$

is a secure PRF on $\{0,1\}^n$.

Reference: Boneh–Shoup, v0.6, Thm. 4.10, p. 148, proof pp. 148–151. Original: [GGM86].

Prefix-free GGM, used only in the optional repair

Under the same PRG assumption, the tree is secure for polynomially many queries on nonempty prefix-free paths of polynomially bounded length.

Reference: Boneh–Shoup, Thm. 4.11, pp. 152–154. Encoding conversion: Thm. 6.8, p. 235.

Technical conventions: lengths and endorsement security

Length and query conventions

All resources and query lengths are polynomially bounded in λ. The MAC experiment allows adaptive tagging queries and one final fresh-message forgery, with no verification oracle.

The independent-key repair supports polynomially many lengths. Its hybrid proof uses only PRF security.

References: Boneh–Shoup, §2.3.1, pp. 28–29; Game 6.1, pp. 218–219; verification-query variants in §6.2, pp. 220–223.

Scope of the endorsement exercise

The trusted verifier shares independent keys with registered parties. Message encoding is unambiguous. The adversary controls a fixed subset. With fewer than t authorizations for m, acceptance forces a fresh forgery under an honest key. The certificate contains t tags.