Signatures on Arbitrary-length Messages

Lecture 24b: Binding the blocks together

Lecture overview

A signature on each block does not automatically authenticate the whole message. Three attacks reveal exactly which context each block must carry.

Based on the instructor's Lecture 24b, pp. 1–10 [Maj26].

Abstract and lecture roadmap

We start with a secure signature scheme for fixed-size inputs and build a scheme for long messages. The central issue is how to prevent valid block signatures from being rearranged or reused.

1. Break three natural constructions: truncation, permutation, and splicing.
2. Authenticate the block count, index, and a common random identifier.
3. Write the algorithms and prove security with a birthday-bound loss.

The question that drives the lecture

What must a signed block say about its place in the *whole* message?

An optional extension uses collision-resistant hashing to remove the explicit block-count limit.

Source sequence: [Maj26, pp. 2–10]. Hash-and-sign: [BS23, §13.2].

The results to keep in view

Attacks on the first three attempts

Signing blocks separately permits truncation. Adding a count still permits permutation. Adding positions still permits splicing across messages.

Security of the fourth attempt

Let $B = 4b$. A common fresh b-bit identifier gives, for at most q signing queries,

$$\mathsf{Adv}_{\Sigma^*}(\mathcal{A}) \leq \mathsf{Adv}_{\Sigma}(\mathcal{B}) + \binom{q}{2} 2^{-b}.$$

Scope. The block construction supports $\alpha < 2^b$ padded blocks. Hash-and-sign removes this limit under an additional assumption.

Construction: [Maj26, pp. 3–10]. The quantitative block-scheme proof is supplied in this deck.

The basic signature scheme

Let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ sign messages in $\{0,1\}^B$.

The signer

Generate a key pair:

$$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda).$$

For a B-bit message x, compute

$$\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, x).$$

Anyone can verify

Given (pk, x, σ), compute

$$\mathsf{Ver}(\mathsf{pk}, x, \sigma) \in \{0,1\}.$$

Correctness requires

$$\mathsf{Ver}(\mathsf{pk}, x, \mathsf{Sign}(\mathsf{sk}, x)) = 1.$$

The new scheme will keep the same key pair and call the basic signing algorithm on carefully encoded blocks.

[BS23, Def. 13.1, pp. 528–529]; undergraduate background: [HPS14, §4.1, pp. 193–195].

What counts as a forgery?

1. The attacker receives pk.
2. It adaptively requests signatures on messages $m^{(1)}, \ldots, m^{(q)}$.
3. It outputs a candidate (m^*, σ^*).

The attacker wins if both conditions hold

$$\mathrm{Ver}^*(\mathrm{pk}, m^*, \sigma^*) = 1, \qquad m^* \notin \{m^{(1)}, \ldots, m^{(q)}\}.$$

This is **existential unforgeability under chosen-message attack**, or EUF-CMA.

Write $\mathrm{Adv}_{\Sigma^*}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$. Reusing old signatures can win if the message is new.

Replaying a signature on the same message does not win this game. Ordinary EUF-CMA is sufficient for today's construction.

[BS23, Attack Game 13.1 and Def. 13.2, pp. 529–530].

Padding must preserve the message

For payload-block width $w \geq 1$, use the deterministic encoding

$$\mathrm{Pad}_w(m) = m \,\|\, 1 \,\|\, 0^k, \qquad 0 \leq k < w, \qquad |m| + 1 + k \equiv 0 \pmod{w}.$$

Example: $w = 4$

$$m = 10110,$$
$$\mathrm{Pad}_4(m) = 1011 \,\|\, 0100.$$

Even the empty message gets a block: $\mathrm{Pad}_4(\varepsilon) = 1000$.

Why this encoding is injective

Remove all final zeros, then remove the final 1. This uniquely recovers m.

Thus two different messages cannot have the same padded block sequence.

For attack diagrams, take block width at least 2 and distinct blocks X, Y ending in 1. Every displayed concatenation is valid canonical padding.

Making the source's "standard padding" precise: [Maj26, pp. 3, 5, 7, 9].

First attempt: sign every block separately

Pad and split into B-bit blocks:

$$\mathsf{Pad}_B(m) = m_1 \parallel \cdots \parallel m_\alpha.$$

Signing

Compute $\sigma_i \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m_i)$ for $1 \leq i \leq \alpha$. Return

$$\sigma = (\sigma_1, \ldots, \sigma_\alpha).$$

Verification

Recompute the padded blocks, check that there are exactly α signatures, and accept if every $\mathsf{Ver}(\mathsf{pk}, m_i, \sigma_i) = 1$.

Each signature authenticates the contents of one block. The signed input contains no information about the other blocks.

[Maj26, p. 3].

Prefix attack: discard the last block

Request a signature on the message whose padding is $X \| X$.



The verifier accepts a new message

The shorter padded message X still has a valid signature on every block. The signature never committed to the total number of blocks.

For example, $X = 0001$: the query message is 0001000, and the forgery message is 000. Both use canonical padding.

Source attack: [Maj26, p. 4]. The example also checks padding validity.

Second attempt: include the block count

For this attempt, suppose $B = 2w$. Split into w-bit payload blocks and require $1 \leq \alpha < 2^w$.

The basic message signed at position i

$$\boxed{[\alpha]_w \parallel m_i} \qquad (w \text{ count bits} + w \text{ payload bits}).$$

Here $[\alpha]_w$ is the unique w-bit binary encoding of α.

Return all block signatures. The verifier recomputes α from the message and checks every signature on $[\alpha]_w \parallel m_i$.

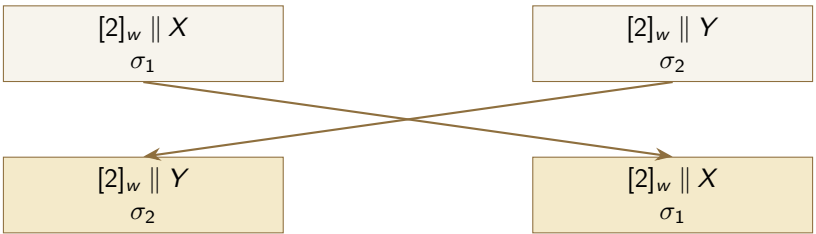
What the count accomplishes

Truncating from α to $i < \alpha$ changes the input that every retained signature must authenticate.

[Maj26, p. 5].

Permutation attack: exchange the two blocks

Request a signature on the message whose padding is $X \parallel Y$, with $X \neq Y$.



The count is correct in both orders

The forged padded message $Y \parallel X$ has two blocks, and both reused signatures verify.

Next repair. Authenticate the *position* of each block.

[Maj26, p. 6]. Take X, Y ending in 1 so both orders are valid padding.

Third attempt: include count and position

For this attempt, suppose $B = 3w$. Split into w-bit payload blocks and require $1 \leq \alpha < 2^w$.

The basic message signed at position i

$$\boxed{[\alpha]_w \parallel [i]_w \parallel m_i} \qquad (1 \leq i \leq \alpha).$$

Each of the three fields has exactly w bits.

The verifier reconstructs the count and each position from the received message.

What the position accomplishes

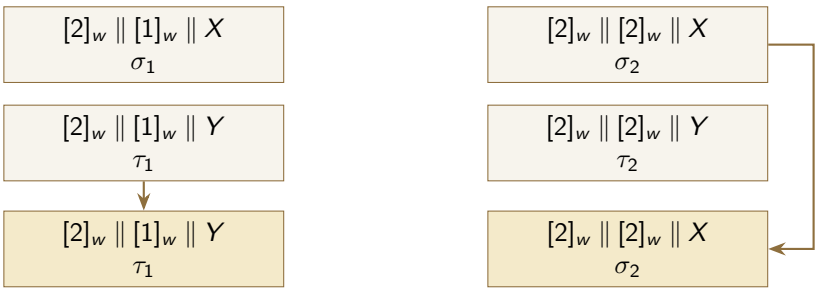
A signature authenticating position i cannot simply be moved to position $j \neq i$: the verifier now checks a different basic message.

The next attack uses more than one signing query, as allowed by the security game.

[Maj26, p. 7].

Splicing attack: take a block from another message

Request signatures on padded messages (X, X) and (Y, Y), where $X \neq Y$.



Both blocks have the right count and position

The forged padded message (Y, X) is new. Nothing says that its two signed blocks belong to the same signing query.

[Maj26, p. 8]. Again choose X, Y ending in 1.

Each attack identifies missing context

Attempt	Basic input being signed	Remaining attack
First	m_i	Prefix / truncation
Second	$[\alpha]_w \parallel m_i$	Permutation
Third	$[\alpha]_w \parallel [i]_w \parallel m_i$	Splicing
Fourth	$[\alpha]_b \parallel [i]_b \parallel s \parallel m_i$	Security proof next

The final missing field

Use one random identifier s for the *entire signing operation*, and authenticate that same identifier in every block.

Field widths change across attempts to keep every basic signed input exactly B bits.

The progression in source pp. 3–9 [Maj26].

Fourth attempt: a common random identifier

From now on, let $B = 4b$. Write $\mathrm{Pad}_b(m) = m_1 \| \cdots \| m_\alpha$, where $1 \le \alpha < 2^b$, and sample one fresh $s \xleftarrow{\$} \{0,1\}^b$.



Every block carries the same message identifier

$$x_i = [\alpha]_b \| [i]_b \| s \| m_i, \qquad \sigma_i \xleftarrow{\$} \mathrm{Sign}(\mathsf{sk}, x_i).$$

Return $\sigma^* = (s, \sigma_1, \ldots, \sigma_\alpha)$.

The identifier is public and must accompany the signatures so the verifier can reconstruct every x_i.

[Maj26, p. 9]; the returned signature explicitly includes s.

The new key generation and signing algorithms

Key generation. Run $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and return the same keys.

$\mathsf{Sign}^*(\mathsf{sk}, m)$

1. Compute $\mathsf{Pad}_b(m) = m_1 \parallel \cdots \parallel m_\alpha$.
2. If $\alpha \geq 2^b$, return $\perp$ (the message is outside the supported domain).
3. Sample a fresh uniform identifier $s \xleftarrow{\$} \{0,1\}^b$.
4. For each $i = 1, \ldots, \alpha$, compute
$$\sigma_i \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, [\alpha]_b \parallel [i]_b \parallel s \parallel m_i).$$
5. Return $(s, \sigma_1, \ldots, \sigma_\alpha)$.

Use fresh s even when signing the same message again. No counter or state is required.

Algorithms completing the source's exercise: [Maj26, pp. 9–10].

The new verification algorithm

$\mathsf{Ver}^*(\mathsf{pk}, m, \sigma^*)$

1. Recompute $\mathsf{Pad}_b(m) = m_1 \parallel \cdots \parallel m_\alpha$. Reject if $\alpha \geq 2^b$.
2. Parse $\sigma^* = (s, \sigma_1, \ldots, \sigma_\alpha)$. Reject malformed encodings, an identifier of length other than b, or the wrong number of signatures.
3. Accept exactly when, for every $i = 1, \ldots, \alpha$,

$$\mathsf{Ver}(\mathsf{pk}, [\alpha]_b \parallel [i]_b \parallel s \parallel m_i, \sigma_i) = 1.$$

One identifier, reconstructed context

The verifier uses the *same* s in every check. It derives the count and positions itself from m.

Completing the algorithm requested in source p. 10 [Maj26].

A complete encoding example

Take $b = 4$, so the basic scheme signs $B = 16$ bits. Let $m = 10110$.

$$\mathrm{Pad}_4(m) = \underbrace{1011}_{m_1} \| \underbrace{0100}_{m_2}, \qquad \alpha = 2, \qquad s = 0110.$$

	Count	Index	Identifier	Payload
x_1	0010	0001	0110	1011
x_2	0010	0010	0110	0100

Return $(0110, \sigma_1, \sigma_2)$, where σ_i signs the 16-bit row x_i.

A second signing operation uses a new identifier. All blocks *within* one operation use the same identifier.

These tiny parameters illustrate the encoding only. A four-bit identifier gives no meaningful collision protection.

Worked instance of [Maj26, p. 9].

Correctness of the fourth construction

Proof idea. The verifier reconstructs exactly the basic messages that the signer used.

For any supported message m, both algorithms compute the same canonical padding and the same α.

The signature supplies the signer's identifier s. At each position i, the verifier checks

$$x_i = [\alpha]_b \parallel [i]_b \parallel s \parallel m_i.$$

Correctness follows block by block

Basic correctness gives $\mathsf{Ver}(\mathsf{pk}, x_i, \sigma_i) = 1$ for every i. Therefore

$$\boxed{\mathsf{Ver}^*(\mathsf{pk}, m, \mathsf{Sign}^*(\mathsf{sk}, m)) = 1.}$$

This proves honest signatures verify. Unforgeability requires a separate argument about every possible attacker.

Basic correctness: [BS23, Def. 13.1, pp. 528–529].

What happens to the three attacks?

Change	The basic input that changes
Delete blocks	The recomputed count $[\alpha]_b$ changes.
Move a block	Its position $[i]_b$ changes.
Import a block	If the two identifiers differ, the verifier checks the imported signature on a different basic input.

There are two ways a new forgery could survive

It supplies a valid signature on a basic input never signed before, or two signing operations happened to receive the same identifier.

The next proof makes this dichotomy rigorous for every attack.

Source intuition: [Maj26, pp. 9–10].

The birthday bound for message identifiers

Let $s_1, \ldots, s_q$ be independent uniform strings in $\{0,1\}^b$. Define

$$\text{Bad} = \{s_j = s_k \text{ for some } j < k\}.$$

Proof idea. For each pair, the probability of equality is 2^{-b}. Apply the union bound.

Collision probability

$$\Pr[\text{Bad}] \leq \sum_{1 \leq j < k \leq q} \Pr[s_j = s_k] = \boxed{\binom{q}{2} 2^{-b}}.$$

Adaptive choices of messages do not change the signer's fresh independent sampling of identifiers.

For $b = 256$ and $q \leq 2^{64}$, this bound is less than 2^{-129}.

Birthday background: [BS23, §B.1, Thm. B.1, p. 1101]; [HPS14, §§5.4.1–5.4.2, pp. 246–249].

The security theorem and its proof roadmap

Theorem: security reduces to the basic scheme

If $\mathcal{A}$ makes at most q extended signing queries containing at most L padded blocks in total, there is a basic-signature attacker $\mathcal{B}$ making at most L signing queries such that

$$\boxed{\mathsf{Adv}_{\Sigma^*}(\mathcal{A}) \leq \mathsf{Adv}_{\Sigma}(\mathcal{B}) + \binom{q}{2} 2^{-b}.}$$

1. Simulate each extended signature using the basic signing oracle.
2. Prove that, with distinct query identifiers, a new extended forgery contains a previously unsigned basic input.
3. Extract that input and its valid signature. Add the collision probability.

For $b = \lambda$ and polynomial q, L, the collision term is negligible.

EUF-CMA convention: [BS23, Attack Game 13.1, pp. 529–530]. Block construction: [Maj26, p. 9].

The key lemma: some basic input must be new

Assume all signing-query identifiers are distinct. Consider an accepted forgery on a new message m^*, with identifier s^*.

Proof by contradiction. Suppose every forged basic input had been signed before.

A contradiction in four steps

1. The first forged input contains s^*. If it was signed before, s^* identifies a unique query j.
2. Every other forged input contains s^*, so it too must come from query j.
3. Equality of count and index fields forces $\alpha^* = \alpha_j$ and $m_i^* = m_i^{(j)}$ at every position i.
4. The padded messages agree. Injectivity of padding gives $m^* = m^{(j)}$, contradicting freshness.

Padding always supplies at least one block. If s^* never appeared in a query, every forged basic input is immediately new.

A complete proof of the source's security claim [Maj26, p. 10].

The reduction: extract a basic forgery

1. Run $\mathcal{A}$ with the basic challenger's public key.
2. Pad each message query. Return $\perp$ if it is too long. Otherwise sample s, obtain every σ_i from the basic oracle, and record its input.
3. Given an accepted new-message forgery, find a basic input x_i^* absent from the record and output (x_i^*, σ_i^*).

The lemma guarantees extraction outside Bad

The simulation is exact and uses at most L oracle calls. Consequently,

$$\begin{aligned}\mathrm{Adv}_{\Sigma^*}(\mathcal{A}) &\le \Pr[\mathcal{A} \text{ wins and } \neg\mathrm{Bad}] + \Pr[\mathrm{Bad}] \\ &\le \mathrm{Adv}_{\Sigma}(\mathcal{B}) + \binom{q}{2} 2^{-b}.\end{aligned}$$

No block guessing or additional collision term for the final identifier is needed.

Reduction for the source's fourth construction [Maj26, pp. 9–10].

How long a message does this construction support?

Canonical padding gives

$$\alpha = \left\lfloor \frac{|m|}{b} \right\rfloor + 1.$$

Since the count has b bits and must be positive,

$$\alpha \leq 2^b - 1 \quad \Longleftrightarrow \quad \boxed{|m| \leq b(2^b - 1) - 1.}$$

The usual asymptotic interpretation

If $b = \lambda$, every polynomial message length fits for all sufficiently large security parameters λ.

Cost of signing a long message

There are α basic signing calls and α basic verification checks. If each basic signature has ℓ bits, the signature fields total $b + \alpha\ell$ bits, before serialization overhead.

Field-width qualification of the construction in [Maj26, p. 9].

An arbitrary-length domain via hash-and-sign

Optional extension. Add a collision-resistant hash family with public key h:

$$H_h : \{0,1\}^* \longrightarrow \{0,1\}^B.$$

Signing

Generate h independently of the basic signature keys and include it in the public key.

$$\sigma \xleftarrow{\$} \mathrm{Sign}(\mathsf{sk}, H_h(m)).$$

There is one basic signing call.

Verification

Recompute the digest and check

$$\mathrm{Ver}(\mathsf{pk}, H_h(m), \sigma) = 1.$$

The signature length is that of one basic signature.

Collision resistance: given h, finding distinct m, m' with $H_h(m) = H_h(m')$ is computationally hard.

[BS23, §13.2, eq. (13.1), p. 532]; hashing background: [HPS14, §8.1, pp. 472–474].

Why hash-and-sign is secure

Proof idea. Separate new digests from repeated digests.

For an accepted forgery (m^*, σ^*), put $d^* = H_h(m^*)$.

Two exhaustive cases

New digest. If d^* was never submitted for signing, then (d^*, σ^*) is a basic-signature forgery.

Repeated digest. Otherwise $d^* = H_h(m^{(j)})$ for an earlier query. Freshness gives $m^* \neq m^{(j)}$, so $(m^*, m^{(j)})$ is a collision.

Hash-and-sign theorem

$$\mathsf{Adv}_{\text{hash-sign}}(\mathcal{A}) \leq \mathsf{Adv}_{\Sigma}(\mathcal{B}_{\text{sig}}) + \mathsf{Adv}^{\text{cr}}_{H}(\mathcal{B}_{\text{hash}}).$$

Thus EUF-CMA security and collision resistance suffice.

[BS23, Thm. 13.1, p. 533].

Checks for understanding

1. In the fourth construction, what fails if the signer reuses one fixed identifier forever?
2. What goes wrong if each block uses an independently sampled identifier and the verifier checks each one separately?
3. Why does the security proof need to find only *one* previously unsigned basic input?
4. With $q \leq 2^{40}$ signing queries, how large should b be to make the collision bound at most 2^{-80}?

The principle to remember

Each signed block must authenticate its contents *and* the context that makes those contents part of one particular message.

Exercises accompanying the constructions in [Maj26, pp. 3–10].

References and reading

[BS23] Dan Boneh and Victor Shoup.
A Graduate Course in Applied Cryptography.
Online textbook, 2023.
Version 0.6, January 2023.
URL: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_6.pdf.

[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.
CS355, Lecture 24b: Signatures on Arbitrary-length Messages.
Instructor's lecture slides, Purdue University, 2026.
Source supplied by the instructor: 24b.pdf, 10 slides.

Technical inputs for the block construction

Basic signature assumption: EUF-CMA

For every efficient $\mathcal{B}$ with polynomially many signing queries,

$$\Pr[\mathsf{Ver}(\mathsf{pk}, x^*, \sigma^*) = 1 \text{ and } x^* \text{ was never queried}] \leq \mathsf{negl}(\lambda).$$

Basic correctness holds for every B-bit input. We take $B = 4b$, with $b = \lambda$ sufficient for the stated asymptotic conclusion.

Local requirements, all explicit in this deck

Use injective canonical padding, fixed-width field encodings, and fresh independent uniform identifiers. Accept only $1 \leq \alpha < 2^b$ and a well-formed signature tuple.

Padding injectivity, the birthday upper bound, and the extraction lemma were proved here. No further cryptographic assumption is used.

[BS23, Def. 13.1, pp. 528–529; Attack Game 13.1 and Def. 13.2, pp. 529–530].

Additional input for the optional extension

Collision-resistant hashing

Let $h \xleftarrow{\$} \mathsf{HGen}(1^\lambda)$ define an efficiently evaluable family $H_h : \{0,1\}^* \to \{0,1\}^B$. For every efficient $\mathcal{C}$,

$$\Pr\big[(m, m') \leftarrow \mathcal{C}(h) : \ m \neq m' \text{ and } H_h(m) = H_h(m')\big] \leq \mathsf{negl}(\lambda).$$

Generate h independently of the basic signature keys and publish it.

Imported theorem: hash-and-sign

A secure basic signature scheme combined with collision-resistant hashing gives a secure signature scheme on the hash function's message domain.

Precise source. Boneh–Shoup, v0.6 (January 2023), §13.2, eq. (13.1), p. 532, and Theorem 13.1, p. 533. The two-case proof appears on slide 26.

[BS23, §13.2, Thm. 13.1]; background: [HPS14, §8.1].