Collision-Resistant Hash Functions

Lecture 21: Short digests, hard collisions

Lecture overview

Compression forces collisions to exist. We want finding one to be computationally hard, even when the attacker knows the hash function.

Based on the instructor's Lecture 21 [Maj26].

Abstract and lecture roadmap

A hash function replaces a long input with a short digest. What should happen if an attacker tries to make two different inputs share that digest?

1. Compare universal hashing, universal one-way hashing, and collision resistance.
2. Build collision resistance from the discrete-log assumption.
3. Extend a short-input hash to long messages with Merkle–Damgård.

Proof roadmap

For the algebraic construction, a collision gives an equation we can solve. For domain extension, a collision in the full computation reveals a collision in one compression step.

[BS23, §§8.1, 8.4, 10.6.1].

The two main results

Discrete logarithms give collision resistance

Let $G = \langle g \rangle$ have prime order q, and choose a public $y \xleftarrow{\$} G$. Then

$$h_y \colon \mathbb{Z}_q^2 \to G, \qquad h_y(x_1, x_2) = g^{x_1} y^{x_2}$$

is collision resistant under the discrete-log assumption. Every collision recovers $\log_g y$, with no loss in success probability.

A collision-resistant compression function handles long messages

Iterate a collision-resistant $h_k \colon \{0,1\}^{2B} \to \{0,1\}^B$ using the same public key k. With the length padding specified below, every collision for the resulting hash gives a collision for h_k.

We prove both reductions. The one-bit variant is the restriction $x_2 \in \{0,1\}$.

[BS23, Thm. 10.4, p. 415; Thm. 8.3, pp. 297–298].

The hash key is public

A family of efficiently computable maps

$$\mathcal{H}_\lambda = \{h_k \colon D_\lambda \to R_\lambda\}, \qquad k \xleftarrow{\$} \mathrm{Gen}(1^\lambda).$$

The parameter λ controls the security level. The attacker receives k and can evaluate h_k on any input.

- A **collision** is a pair $x \neq x'$ with $h_k(x) = h_k(x')$.
- If $|D_\lambda| > |R_\lambda|$, the pigeonhole principle guarantees a collision.
- Security asks that every probabilistic polynomial-time attacker find one with only negligible probability in λ.

We usually suppress λ in the notation.

[BS23, §8.1, Attack Game 8.1 and Def. 8.1, pp. 290–291].

Universal hashing: fix the pair first

Universal family

For every two distinct inputs $x, x' \in D$,

$$\Pr_{k \xleftarrow{\$} \mathrm{Gen}()} [h_k(x) = h_k(x')] \leq \frac{1}{|R|}.$$

The inputs are fixed before the random hash key is sampled.

This is an information-theoretic statement. It does not restrict anyone's computing power.

The order matters

A small collision probability for every fixed pair says nothing by itself about a pair selected after seeing k.

[BS23, §7.1, Def. 7.2 and eq. (7.1), p. 254]; source slides, p. 3 [Maj26].

A linear universal family

Over a finite field $\mathbb{F}_q$, sample (a, b) uniformly from $\mathbb{F}_q^2 \setminus \{(0,0)\}$ and set

$$h_{a,b}(x_1, x_2) = ax_1 + bx_2.$$

Counting the keys that cause a collision

Fix $x \neq x'$ and write $(u, v) = x - x' \neq (0,0)$. A collision occurs exactly when

$$au + bv = 0.$$

This equation has q solutions (a, b): one coordinate determines the other. Exactly $q - 1$ solutions are nonzero.

Therefore

$$\Pr[h_{a,b}(x) = h_{a,b}(x')] = \frac{q-1}{q^2-1} = \frac{1}{q+1} \leq \frac{1}{q}.$$

Construction from source slides, p. 3 [Maj26]; proof by direct counting.

After the key is public, collisions are easy

For every nonzero key (a, b), the vector $(b, -a)$ is nonzero and

$$h_{a,b}(x_1 + b, x_2 - a) = h_{a,b}(x_1, x_2).$$

A small example in $\mathbb{F}_7$

Take $(a, b) = (2, 3)$. Then $(b, -a) = (3, 5)$, and

$$h_{2,3}(0, 0) = 0, \qquad h_{2,3}(3, 5) = 6 + 15 = 0 \pmod 7.$$

The attacker can even commit to $(0, 0)$ before seeing the key, then output $(b, -a)$ afterward.

This family is universal, but it fails both computational notions we define next.

Universal one-way hashing: commit to a target

1. The attacker's first stage chooses a target $x \in D$ and saves a state s.
2. The challenger independently samples $k \xleftarrow{\$} \mathrm{Gen}(1^\lambda)$ and reveals k.
3. Given (k, x, s), the second stage outputs $x' \in D$.

Universal one-way hash function family (UOWHF)

Every efficient two-stage attacker has negligible probability of satisfying

$$x' \neq x \qquad \text{and} \qquad h_k(x') = h_k(x).$$

This is also called **target collision resistance** in this experiment.

The target is chosen before the key. The attempted partner is chosen after it.

[BS23, §8.11.2, Attack Game 8.5, Def. 8.7, pp. 337–338]; [NY89].

Why UOWHF security needs a computational limit

Suppose D and R are finite. An unbounded attacker chooses $x \xleftarrow{\$} D$ before the key, then searches all of D for another input with the same hash.

How often does the search succeed?

For each fixed key, at most $|R|$ inputs can be alone in their hash bucket: each such input uses a different output.

$$\Pr[\text{another input has the same hash}] \geq 1 - \frac{|R|}{|D|}.$$

The bound also holds after averaging over the random key.

When the function compresses substantially, this is a high-probability attack.

Efficient security is possible: Naor–Yung introduced UOWHFs, and Rompel constructed them from one-way functions. We use this existence result only as context.

[NY89, Rom90]; counting argument resolves the exercise on source p. 4 [Maj26].

Collision resistance: choose both inputs afterward

1. The challenger samples $k \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and gives k to the attacker.
2. The attacker outputs two inputs $(x, x') \in D^2$.

Collision-resistant hash function family (CRHF)

For every efficient attacker A,

$$\Pr\left[\begin{array}{c}(x,x') \leftarrow A(k), \quad x \neq x', \\ h_k(x) = h_k(x')\end{array}\right] \leq \mathsf{negl}(\lambda).$$

The probability includes key generation and the attacker's randomness.

With unlimited computation and $|D| > |R|$, exhaustive search succeeds for every key.

[BS23, Attack Game 8.1 and Def. 8.1, pp. 290–291].

The order of choices explains the difference

Notion	Before the random key	After the key is revealed
Universal hashing	Fix both x, x'	Test whether they collide
UOWHF	Choose target x	Try to find its partner x'
CRHF	No commitment	Try to find both x and x'

Every CRHF is a UOWHF

A CRHF attacker can run the UOWHF target stage without using the key, then give the key to its second stage. It outputs the resulting pair. The success probability is unchanged.

Universal hashing has a different quantifier order. Our linear example shows why it does not give either computational guarantee.

[BS23, §8.11.2, p. 338]; universal definition: [BS23, §7.1].

A short digest has a birthday limit

Imagine that hashing distinct inputs gives independent uniform B-bit values. After T inputs, the expected number of colliding pairs is

$$\binom{T}{2} 2^{-B}.$$

The scale of a generic collision search

A constant probability of a collision appears when T is on the order of $2^{B/2}$. In the ideal model,

$$\Pr[\text{a collision}] \approx 1 - \exp\left(-\frac{T(T-1)}{2^{B+1}}\right).$$

For a 256-bit digest, the classical birthday scale is about 2^{128} evaluations. This is an ideal-model calculation, not a security proof for a particular hash family.

[BS23, §8.3, pp. 293–295]; [HPS14, §5.4.1, pp. 246–247].

The algebraic setup

Let $G = \langle g \rangle$ be a cyclic group of known **prime order** q.

Two kinds of objects

exponents: $x, \alpha \in \mathbb{Z}_q$, group elements: $g, y \in G$.

Every $y \in G$ is uniquely g^α. We can compute powers efficiently, but recovering α from a random y is assumed hard.

- Sample $y \xleftarrow{\$} G$ as the public hash key.
- The attacker knows (G, q, g, y), but is not given $\alpha = \log_g y$.
- Group elements have canonical representations and efficient equality tests.

For asymptotic security, these groups form a family indexed by λ.

[BS23, Def. 10.6, p. 409; §10.6.1, p. 414]; [HPS14, §§2.2, 2.5].

One extra bit, one group element

The one-bit variant

$$h_y : \mathbb{Z}_q \times \{0,1\} \to G, \qquad h_y(x,b) = g^x y^b.$$

The two branches are $h_y(x,0) = g^x$ and $h_y(x,1) = g^x y$.

Each branch is a bijection from $\mathbb{Z}_q$ to G, so a collision must use different branches.

Where is the hard step?

The hidden exponent $\alpha = \log_g y$ is the shift between the two branches. Finding a collision tells us that shift.

The domain has $2q$ elements and the range has q. "One bit" refers to this factor-two reduction in cardinality.

Source construction, p. 6 [Maj26]; compare [BS23, §10.6.1].

A one-bit collision reveals the discrete logarithm

Suppose distinct (x, b) and (x', b') satisfy $g^x y^b = g^{x'} y^{b'}$.

1. If $b = b'$, cancellation gives $g^x = g^{x'}$, hence $x = x'$ in $\mathbb{Z}_q$. This would be the same input, so $b \neq b'$.
2. Writing $y = g^\alpha$ gives
$$x + \alpha b = x' + \alpha b' \pmod{q}.$$
3. Since $b - b' \in \{1, -1\}$, we can solve:
$$\boxed{\alpha = (x' - x)(b - b')^{-1} \pmod{q}.}$$

The reduction

Given a random discrete-log challenge y, use it as the hash key and run the collision finder. Every valid collision yields the correct answer.

Restriction of the reduction in [BS23, Fact 10.3 and Thm. 10.4, p. 415].

A small collision that exposes the hidden shift

Work in the order-11 subgroup $G = \langle 2 \rangle \subseteq \mathbb{Z}_{23}^*$. Take $g = 2$ and public key $y = 8$.

Suppose someone finds this collision

$$h_y(5,0) = 2^5 = 9 \pmod{23}, \qquad h_y(2,1) = 2^2 \cdot 8 = 9 \pmod{23}.$$

The inputs $(5,0)$ and $(2,1)$ are different.

The recovery formula gives

$$\alpha = (2-5)(0-1)^{-1} = 3 \pmod{11}.$$

Direct check

$2^3 = 8 \pmod{23}$, so the recovered exponent is exactly $\log_2 8$ in G.

Illustrative small parameters only; group background: [HPS14, §2.5].

Two exponents, one group element

The length-halving viewpoint

$$h_y : \mathbb{Z}_q \times \mathbb{Z}_q \to G, \qquad h_y(x_1, x_2) = g^{x_1} y^{x_2}.$$

This maps q^2 possible inputs to q possible outputs.

Proof idea

A collision gives two expressions for the same group element. After writing $y = g^{\alpha}$, equality becomes one linear equation in the unknown α.

In terms of cardinality, the input contains $2 \log_2 q$ bits of information and the output contains $\log_2 q$. Actual bit lengths depend on the group encoding.

[BS23, §10.6.1, pp. 414–415, including the encoding caveat].

Why prime order makes the reduction work

Let $(x_1, x_2) \neq (x_1', x_2')$ be a collision. With $y = g^{\alpha}$,

$$x_1 + \alpha x_2 = x_1' + \alpha x_2' \pmod{q}.$$

1. If $x_2 = x_2'$, then $x_1 = x_1'$, contradicting distinctness.
2. Thus $x_2 - x_2' \neq 0$ in the field $\mathbb{Z}_q$, so it has an inverse.
3. Recover

$$\boxed{\alpha = (x_1' - x_1)(x_2 - x_2')^{-1} \pmod{q}.}$$

Exact success bound

Let the solver B reject invalid collisions from A and apply the formula otherwise. Then

$$\mathrm{Adv}_{\mathrm{DL}}(B) = \mathrm{Adv}_{\mathrm{CR}}(A).$$

[BS23, Fact 10.3, Thm. 10.4, eq. (10.2), p. 415].

The two-exponent formula in numbers

Again use $q = 11$, $G = \langle 2 \rangle \subseteq \mathbb{Z}_{23}^*$, and $y = 8$.

A collision

$$h_y(7,4) = 2^7 \cdot 8^4 = 3 \pmod{23}, \qquad h_y(2,2) = 2^2 \cdot 8^2 = 3 \pmod{23}.$$

Since $2^{-1} = 6$ in $\mathbb{Z}_{11}$, the recovered logarithm is

$$\alpha = (2-7)(4-2)^{-1} = (-5) \cdot 6 = 3 \pmod{11}.$$

Check the equation in the exponent

$7 + 3 \cdot 4 = 19 \equiv 8 \pmod{11}$ and $2 + 3 \cdot 2 = 8$. Both group outputs are $2^8 = 3 \pmod{23}$.

Why we do not use the full group $\mathbb{Z}_p^*$ here

The full multiplicative group has order $p-1$, which is composite for odd p. A nonzero exponent difference need not be invertible modulo $p-1$.

An easy collision in a small full group

Take $p = 23$, a generator $g = 5$ of $\mathbb{Z}_{23}^*$, and $y = 10 = 5^3$. Both satisfy

$$g^{11} = y^{11} = -1 \pmod{23}.$$

Consequently,

$$h_y(1,1) = 5 \cdot 10 = 4 \quad \text{and} \quad h_y(12,12) = h_y(1,1)\, g^{11} y^{11} = 4 \pmod{23}.$$

The construction's hypothesis

Use a group of prime order q and exponents in $\mathbb{Z}_q$. This is what justifies division in the collision equation.

Prime-order setup: [BS23, §10.6.1, p. 414]; cyclic groups: [HPS14, §2.5].

From a compression function to a message hash

Starting point

Assume a collision-resistant family

$$h_k \colon \{0,1\}^{2B} \to \{0,1\}^B.$$

Think of an input as a B-bit state followed by a B-bit message block.

Fix a public initial value $\mathrm{IV} \in \{0,1\}^B$. For padded message blocks $M_1, \ldots, M_t$, compute

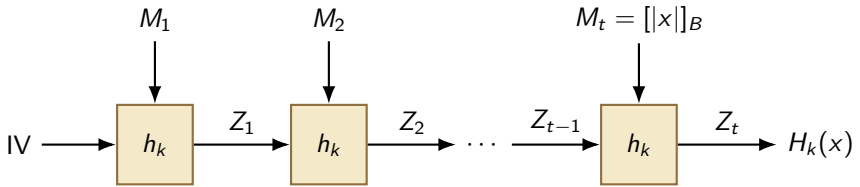
$$Z_0 = \mathrm{IV}, \qquad Z_j = h_k(Z_{j-1} \,\|\, M_j), \qquad H_k(x) = Z_t.$$

One public key throughout

Every step uses the same h_k. Security will come from finding one colliding step in two full computations.

[BS23, §8.4, Fig. 8.5 and construction, pp. 295–297]; [Mer90, Dam90].

Merkle–Damgård: carry the digest forward



Each horizontal state and each vertical block has B bits.

The last block records the original bit length

That block lets the proof distinguish messages of different lengths immediately. The preceding blocks contain the message and its alignment padding.

Adapted from the source diagram, p. 8 [Maj26]; [BS23, Fig. 8.5, p. 296].

Padding specifies the message domain

For the main construction, allow all bit strings x with $n = |x| < 2^B$.

An injective padding rule

Let $[n]_B$ be the B-bit binary encoding of n. Set

$$\mathrm{pad}(x) = x \,\|\, 1 \,\|\, 0^r \,\|\, [n]_B, \qquad 0 \le r < B, \qquad n + 1 + r \equiv 0 \pmod{B}.$$

The result splits into B-bit blocks, with the length in its final block.

For $B = 4$ and $x = 10110$, we have $n = 5$, $r = 2$, and

$$\mathrm{pad}(10110) = \underbrace{1011}_{M_1}\ \underbrace{0100}_{M_2}\ \underbrace{0101}_{[5]_4}.$$

The empty string also has a well-defined encoding. A B-bit trailer cannot encode every possible integer length, so the bound $n < 2^B$ is part of the definition.

Length padding: [BS23, §8.4, p. 296]. An all-finite-strings variant appears in the appendix.

The domain-extension theorem

Merkle–Damgård security

If $h_k\colon \{0,1\}^{2B} \to \{0,1\}^B$ is collision resistant, then the iterated hash H_k, with the preceding padding, is collision resistant on

$$\{x \in \{0,1\}^* : |x| < 2^B\}.$$

Every valid collision for H_k yields a valid collision for h_k under the same key.

Proof roadmap: start at the outputs and work backward

If the two original lengths differ, their final compression inputs already differ. If their lengths agree, repeatedly compare the inputs to the last common output until the two computations diverge.

The reduction preserves success probability and uses linearly many compression calls.

[BS23, Thm. 8.3 and proof, pp. 297–298].

Proof, case 1: different message lengths

Suppose $x \neq x'$ and $H_k(x) = H_k(x')$. Let their padded chains have t and t' blocks.

If $|x| \neq |x'|$, their final blocks are different:

$$M_t = [|x|]_B \neq [|x'|]_B = M'_{t'}.$$

The final compression calls form a collision

$$h_k(Z_{t-1} \,\|\, M_t) = H_k(x) = H_k(x') = h_k(Z'_{t'-1} \,\|\, M'_{t'}).$$

The two $2B$-bit inputs are distinct because their final B bits differ.

We do not need the two chains to have the same number of blocks.

[BS23, Proof of Thm. 8.3, p. 297].

Proof, case 2: equal message lengths

Now $|x| = |x'|$, so the padded strings have the same block count t. Start with $Z_t = Z'_t$ and compare

$$U_t = Z_{t-1} \,\|\, M_t, \qquad U'_t = Z'_{t-1} \,\|\, M'_t.$$

One backward step

If $U_t \neq U'_t$, return this collision for h_k.

If $U_t = U'_t$, then $Z_{t-1} = Z'_{t-1}$ and $M_t = M'_t$. Move one step backward and repeat.

Why the scan must find a collision

If every compared input pair were identical, all padded blocks would agree. The padding is injective, so $x = x'$, a contradiction.

[BS23, Proof of Thm. 8.3, pp. 297–298].

Collision resistance and authentication

A public digest alone does not authenticate a message

If an attacker replaces $(m, H_k(m))$ with $(m', H_k(m'))$, the recomputed digest matches. Anyone can evaluate a public hash.

Authentication requires a trusted digest, a MAC, a signature, or another authenticated part of the protocol.

A separate property: length extension

Merkle–Damgård exposes its final chaining state. Given a digest and the original length, one can continue the computation on additional blocks.

This does not itself produce a collision. It explains why simply using $H(K \parallel m)$ as a MAC is dangerous for this construction.

Hash-based authentication and extension attacks: [BS23, §8.7, p. 307]; [HPS14, Remark 8.1, p. 472].

What to remember

1. **The order of choices is part of security.** Universal hashing fixes both inputs first. UOWHF fixes one first. CRHF lets the attacker choose both after seeing the key.
2. **A discrete-log collision gives a linear equation.** Prime order lets us divide by the nonzero exponent difference.
3. **A long-message collision reveals a short collision.** Length padding and a backward scan make the reduction work.

Two details worth checking every time

Which set do the exponents belong to? Which message lengths does the padding actually encode?

Quick checks

1. For $h_{a,b}(x_1, x_2) = ax_1 + bx_2$ with a nonzero uniform key over $\mathbb{F}_q$, why is the fixed-pair collision probability $1/(q+1)$?

2. A UOWHF attacker commits to $(0, 0)$. After seeing (a, b), which input breaks the linear family?

3. Given a collision in $g^{x_1} y^{x_2}$, why is $x_2 - x_2'$ nonzero? Where is primality of q used?

4. If two Merkle–Damgård messages have different original lengths but equal digests, which two compression inputs collide?

The answers are the four proof moves from today: count a kernel, use $(b, -a)$, invert in $\mathbb{Z}_q$, and compare the final compression calls.

A literal domain of all finite bit strings

The fixed-width length trailer has a maximum length. For a simple construction on all of $\{0,1\}^*$, suppose $B \geq 2$ and choose three distinct B-bit blocks:

$$a = 0^B, \qquad b = 0^{B-1}1, \qquad c = 1^B.$$

A marker followed by one block per input bit

Let $e(0) = a$ and $e(1) = b$. For $x = x_1 \cdots x_n$, define

$$E(x) = c \parallel e(x_1) \parallel \cdots \parallel e(x_n), \qquad E(\varepsilon) = c.$$

Iterate h_k on these blocks from the same fixed IV.

The encoding is injective. No encoded message is a proper **block suffix** of another: the marker c occurs only at the beginning.

This simple variant uses $n + 1$ compression calls for n bits. Its purpose is an exact all-finite-strings statement.

A teaching variant of the backward-extraction argument in [BS23, Thm. 8.3]; proof on the next slide.

Why a suffix-free block encoding suffices

General form of the domain-extension lemma

Let E be an efficient, injective encoding into nonempty B-bit block sequences, with no encoding a proper block suffix of another. Iterating a CRHF h_k on $E(x)$ preserves collision resistance.

Proof. Given a hash collision, compare the final compression inputs.

1. If they differ, they are the desired collision for h_k.
2. If they agree, both final blocks and both preceding states agree. Strip the final blocks and repeat.
3. If a chain ends before a collision is found, its block sequence is a suffix of the other. Suffix-freeness forces equal block counts, and injectivity forces equal original messages. This contradicts the given collision.

The scan handles unequal chain lengths and the empty original message. As before, every valid collision is converted successfully.

References I

[BS23] Dan Boneh and Victor Shoup.
A Graduate Course in Applied Cryptography.
Version 0.6 edition, January 2023.
Online textbook.
URL: https://toc.cryptobook.us/book.pdf.

[Dam90] Ivan Bjerre Damgård.
A design principle for hash functions.
In Gilles Brassard, editor, *Advances in Cryptology—CRYPTO '89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1990.
doi:10.1007/0-387-34805-0_39.

[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.

References II

[Maj26] **Hemanta K. Maji.**
CS355, lecture 21: Collision-resistant hash function.
Purdue University. Instructor lecture slides supplied for this lecture, 2026.

[Mer90] **Ralph C. Merkle.**
One way hash functions and DES.
In Gilles Brassard, editor, *Advances in Cryptology—CRYPTO '89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1990.
doi:10.1007/0-387-34805-0_40.

[NY89] **Moni Naor and Moti Yung.**
Universal one-way hash functions and their cryptographic applications.
In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 33–43. ACM, 1989.
doi:10.1145/73007.73011.

References III

[Rom90] **John Rompel.**

One-way functions are necessary and sufficient for secure signatures.

In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 387–394. ACM, 1990.

doi:10.1145/100216.100269.

Assumptions and external inputs: discrete logarithms

The sole computational assumption used in our algebraic proofs

There is an efficiently described family of cyclic groups $G_\lambda = \langle g_\lambda \rangle$ of known prime order q_λ, with efficient group operations, canonical encodings, and uniform sampling. For every efficient A,

$$\Pr_{\alpha \xleftarrow{\$} \mathbb{Z}_{q_\lambda}} \left[A(G_\lambda, q_\lambda, g_\lambda, g_\lambda^\alpha) = \alpha \right] \leq \mathsf{negl}(\lambda).$$

Precise reference: Boneh–Shoup, version 0.6, Attack Game 10.4 and Definition 10.6, p. 409; the prime-order hash setup is in §10.6.1, p. 414 [BS23].

Group background: Hoffstein–Pipher–Silverman, second edition, §2.2, pp. 64–67, and §2.5, pp. 74–77 [HPS14]. The collision-to-logarithm reductions were proved in full above.

Assumptions and external inputs: scope of the results

Domain extension is conditional on a compression CRHF

The hypothesis is an efficiently computable collision-resistant family $h_k \colon \{0,1\}^{2B} \to \{0,1\}^{B}$, with $B = B(\lambda)$. The domain-extension proof needs no new hardness assumption.

Precise reference: Boneh–Shoup, version 0.6, §8.4, Theorem 8.3 and proof, pp. 297–298 [BS23]. We supplied the proof and an explicit padding rule.

Historical existence result used only for context

One-way functions suffice to construct UOWHFs. We do not prove or use this result in either main reduction.

Reference: Rompel, *One-Way Functions Are Necessary and Sufficient for Secure Signatures*, STOC 1990, §3.6, Theorem 1, p. 393 [Rom90]. UOWHFs were introduced by Naor–Yung, STOC 1989, pp. 33–43 [NY89].