1. This question is about password recovery (16 points).
Let $h : \{0,1\}^* \to \{0,1\}^{256}$ be a fixed 256-bit hash function. A website stores for each user a username string u and a 256-bit hash $a = h(p)$ of the user's password string p.

(a) Let $\mathcal{P}$ be the set of all alphabetic (i.e., 'a...z,A...Z') passwords of length 11. Compute the size of the set $\mathcal{P}$. 2 points
Answer. Each of the 11 characters can be one of 52 choices (26 in lower alphabet + 26 in upper alphabet), thus $|\mathcal{P}| = 52^11$.

(b) Explain how one can construct an efficient map $f : \{0,1\}^{256} \to \mathcal{P}$ from the hash space to the password space. It has to be approximately balanced, i.e., preimage sizes have to be approximately equal: $|f^{-1}(p_1)| \approx |f^{-1}(p_2)|$ for all $p_1, p_2 \in \mathcal{P}$. 5 points
Answer. $f(x)$*: reinterpret* $x \in \{0,1\}^{256}$ *as* $x \in \mathbb{N}$ *with* $0 \le x < 2^{256}$*. The key idea is to use* $\bmod 52$ *operation to obtain individual characters while ensuring the preimage space is balanced.*
For full points: give a complete description of the map; map is indeed correct from $\{0,1\}^{256}$ *to* $\mathcal{P}$*; map is polytime computable; shown that preimage spaces are balanced.*
A full solution is e.g.: reinterpret the input $x \in \{0,1\}^{256}$ *as an integer and write it as* $x = \sum_i c_i 52^i$ *with* $0 \le c_i < 52$*, and reinterpret* $c_0, \ldots, c_{10}$ *as the 11 characters. Note that for the partial sum* $(\sum_{0 \le i < 11} c_i 52^i) = (x \bmod 52^{11})$*, thus each partial sum has* $\approx 2^{256}/52^{11}$ *preimages. Hence, the map is approximately balanced.*

(c) Explain how to apply Hellman's time-memory trade-off attack to h to recover passwords from the given password space $\mathcal{P}$ with success probability about 0.8. 5 points
Answer. For full points: describe the setup of Hellman's attack: use $f \circ h : \mathcal{P} \to \mathcal{P}$ *as iteration function, use multiple* f_i *for multiple distinct tables, explain and specify parameters: e.g.* $r = m = t = |\mathcal{P}|^{1/3}$*, show that desired success probability holds, explain the application of offline and online work: offline work: compute r distinct tables of m chains, each of length t; online work: take a password hash x, for each table evaluate the chain starting from* $f_i(x)$ *and search each point in the table. On a match rewalk the chain from the start and try to find the desired password preimage to x.*
Example full solution: Hellman's attack uses chains using an iteration function $g = f \circ h : \mathcal{P} \to \mathcal{P}$*. Hellman's attack precomputes look-up tables with m chains, each of length t, starting from randomly selected starting points. It precomputes r tables like that, each with a different f similar to 1b (e.g.* $f_i(x) = f(x \oplus i)$*). Let* $N = |\mathcal{P}| = 52^{11}$*, then use Hellman's parameters* $m = t = r = N^{1/3}$ *which results in a success probability of* ≈ 0.8*. (For different parameters, apply the rule that when* $mt^2 = N$ *the success probability of each table is about* $0.80mt/N$*, i.e., ensure* $mt^2 = N$ *and* $r0.80mt/N = 0.8$*). Now given a password hash* $x = h(p)$*, for each table Hellman's attack evaluates a chain starting from* $f_i(x)$ *of length at most* $t - 1$ *and searches each point in the table. On a match it rewalks the chain from the start point and tries to find the desired password preimage p to x.*

(d) Assume an attacker has computing resources to compute 2^{38} evaluations of $f \circ h$ per second. Estimate the offline and online runtime complexity in

wall clock time (days, hours, seconds) for this attacker as well as the storage requirements. 4 points

Answer. Let $N = |\mathcal{P}| = 52^{11}$. *Offline complexity:* $rmt/2^{38} = N/2^{38} \approx 2.7 \cdot 10^7$ *seconds* $\approx$ *316.5 days.*
Online complexity: $rt + rt = 2N^{2/3} \approx 27.92$ *seconds, or without considering false positives it is half: 13.96 seconds.*
Memory cost: $rm = N^{2/3} \approx 3.8 \cdot 10^{12}$ pairs, *not bits or bytes. (Not necessary:* $\lceil \log_2(52^{11}) \rceil = 63$, *so each pair costs* 126 *bits to store. In total* $\approx$ 8 *TB.)*

3. This question is about hash-based signatures (16 points).

(a) Explain in your own words how the the Winternitz one-time-signature scheme works. 6 points

Answer: For full points: describe chains of length 2^w*; split message into radix* w*; use checksum, also split into radix* w*; explain key generation, signing and verification.*
Example full solution: WOTS uses a n*-bit hash function* f *and hash chains of length* w*:* $pk = f^{w-1}(sk)$. *A* k*-bit message is first split into* $l_1 = \lceil \log_w(2^k) \rceil$ *coefficients* $m_1, \ldots, m_{l_1}$. *To prevent trivial forgeries, it also signs a checksum* $c = l_1(w-1) - \sum_i m_i$ *split into* $l_2 = \lceil log_w(l_1(w-1)) \rceil$ *coefficients* $c_1, \ldots, c_{l_2}$. *Each message and checksum coefficient is signed using its own hashchain. Thus in total it uses* $l_1 + l_2$ *chains. Key generation: select* $l_1 + l_2$ *random secret key elements* $sk_i \in \{0,1\}^n$, *compute public key elements* $pk_i = f^{w-1}(sk_i)$. *Signing: compute message and checksum coefficients as above, each coefficient* x_i *has signature element* $\sigma_i = f^{x_i}(sk_i)$. *Verification: compute message and checksum coefficients as above, for each coefficient* x_i *and* σ_i *verify that* $f^{w-1-x_i}(\sigma_i) =? pk_i$.

(b) Consider Winternitz using a 256-bit hash function and using $w = 2^6$, compute the size of the public key, private key and signature. 4 points

Answer: Using $k = 256$, *we compute* $l_1 = \lceil \log_w(2^k) \rceil = \lceil k \log(2) / \log(w) \rceil = \lceil k / \log_2(w) \rceil = \lceil 256 / \log_2(2^6) \rceil = \lceil 256/6 \rceil = 43$. *And* $l_2 = \lceil \log_w(l_1(w-1)) \rceil = \lceil \log_2(43 \cdot 63) / \log_2(2^6) \rceil = 2$. *In total we have* $l_1 + l_2 = 43 + 2 = 45$ *coefficients of* 256*-bits for the secret key, the public key as well as the signature. Thus each has size* $45 \cdot 256 = 11520$ *bits or* 1440 *bytes. (Instead of* 1440 *bytes, the secret key can also be just 256 bits used to derive all* sk_i*.)*

(c) A user accidentally uses his Winternitz signature key twice. Explain how an attacker can use these signatures to create a new signature. 6 points

Answer: For full points: explain procedure, show (plausibel) existence, show checksum will be correct, show all signature coefficients can be computed.
Example full solution: Consider two messages m, m' *with signatures* σ, σ'. *Let the* $l_1 + l_2$ *coefficients for* m, m', σ, σ' *be denoted as* x_i, x'_i, σ_i, σ'_i, *respectively. The attacker can compute valid signature elements for any* $x''_i = \min(x_i, x'_i) + a_i$ *with* $a_i \geq 0$ *as either* $f^{a_i}(\sigma_i)$ *or* $f^{a_i}(\sigma'_i)$. *Thus any such sequence* $(x''_i)_{i=1}^{l_1+l_2}$ *that also satisfies the checksum relation is a valid signature. The two messages are valid solutions, and any other solution leads to a signature forgery. Show there are plausible cases for more than 2 solutions, e.g.: Let* c *and* c' *be the checksums for* m

and m'. For wlog $c < c'$, then using $c'' = c$ and $m_i'' = m_i' + a_i$ with $a_i \geq 0$ and $\sum a_i = c' - c > 0$ gives many forgeries, as there are many valid choices or $(a_i)_i$. For $c = c'$ then using $c'' = c$ and $m_i'' = \min(m_i, m_i') + a_i$ with $a_i \geq 0$ and $\sum_i a_i = (\sum_i |m_i' - m_i|)/2 > 0$ gives many forgeries, as there are many valid choices or $(a_i)_i$.

5. This question is about differential cryptanalysis (18 points).
Consider the same Toy Cipher from the lectures. Given a key $(k_1, k_2, k_3, k_4, k_5) \in \{0,1\}^{5\times 16}$ and a plaintext $P = P_1||\ldots||P_{16} \in \{0,1\}^{16}$, it encrypts P with K as follows:
Let S be the current state, we start with $S = P$.
Rounds $i = 1, 2, 3, 4$ perform (1) first key mixing

$$S \leftarrow S \oplus k_i,$$

then (2) substitution using a Sbox (defined below)

$$S \leftarrow Sbox(S_1 \ldots S_4)||\ldots||Sbox(S_{12} \ldots S_{16}),$$

and then (3) applies the permutation π_P (Table 1) on the state bits:

$$S \leftarrow S_{\pi_P(1)}||\ldots||S_{\pi_P(16)} = S_1||S_5||S_9||\ldots||S_{12}||S_{16}.$$

Directly after Round 4, another key mixing with round key k_5 is applied. After this, the cipher outputs the current state S as the ciphertext C.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(i)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Table 1: State bit permutation

In contrast to the lecture notes, we use the following SBox:

in	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
out	4	2	14	8	10	12	7	1	15	5	0	11	9	3	6	13

Note: most significant bit is left most bit, so 12 represents '1100' in binary.

Table 2: Sbox

This SBox has the following Difference Distribution Table (Table 3).

	Δout															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	8	0	0	0	4	4	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	4	0	0	4	4	4
3	0	0	0	0	4	4	0	0	0	0	0	4	4	0	0	0
4	0	0	0	0	0	0	8	0	0	4	0	0	0	0	4	0
5	0	0	0	0	0	0	0	0	4	0	0	0	4	4	0	4
6	0	0	0	4	4	0	0	0	4	4	0	0	0	0	0	0
7	0	0	8	4	0	4	0	0	0	0	0	0	0	0	0	0
Δin 8	0	2	0	4	0	0	0	2	0	0	0	2	2	0	2	2
9	0	2	0	0	0	4	0	2	2	2	2	0	0	2	0	0
10	0	4	2	0	2	0	0	0	0	2	0	0	2	2	2	0
11	0	0	2	0	2	0	0	4	2	0	2	2	0	0	0	2
12	0	2	0	0	0	4	0	2	2	2	2	0	0	2	0	0
13	0	2	0	4	0	0	0	2	0	0	0	2	2	0	2	2
14	0	0	2	0	2	0	0	4	2	0	2	2	0	0	0	2
15	0	4	2	0	2	0	0	0	0	2	0	0	2	2	2	0

Table 3: Sbox difference distribution table

(a) Construct a differential trail for this cipher over the first three rounds with only one active SBox in the second round and compute its estimated probability. 6 points

Answer: For full points: specify 3-round trail by at least describing which sboxes are active and which DDT entries they use; compute success probability correctly; everything must correctly align: first round sbox output difference must match third round sbox input difference and must also match which sboxes are active in the second round. The active sboxes in round 1 must match the sbox input difference in round 2, the active sboxes in round 3 must match the sbox output difference in round 2.

Example full solution: First consider the high probability DDT entries: prob 1/2: (1,6), (4,6), (7,2). Note the first 2 options have the fewest bits set: 3 in total, thus 3 active sboxes in round 1 and 3. Let's work out using (1,6) for the second round, using (4,6) goes analogously.

Since $1 = 0001$ *is the input difference of the second round, we'll exactly have sbox S14 active in the first round. Since* $6 = 0110$ *is the output difference of the second round, we'll exactly have sboxes S32 and S33 active in the third round. We can also use (1,6) for the third round, which implies that round 1 must use output difference 1, and that round 2 must have active sboxes matching mask* $1 = 0001$*, i.e., only S24 active.*

In summary: S14 active with output difference 1; S24 active with DDT(1,6) with prob 1/2; S32 and S33 active with DDT(1,6) with prob 1/2. What remains is to pick S14 DDT entry, e.g. DDT(10,1) with prob 1/4. We have $\Delta P = (0000\ 0000\ 0000\ 1010)$ *and third round sbox outputs* $\Delta Y_3 = (0000\ 0110\ 0110\ 0000)$*, with combined success probability* $p = (1/2)^3(1/4) = 1/32$*.*

(b) Explain in your own words how to build a partial key recovery attack on

k_5 from your differential trail and how many plaintext/ciphertext pairs it needs. 6 points

Answer: For full points: use correct round 4 sboxes (don't forget to apply π_P after 3rd round sboxes), describe procedure (sample pairs, invert round 4, count, use highest count as key guess), show how many pairs are sufficient with good/bad key analysis.
Example full solution: We have third round sbox outputs $\Delta Y_3 = (0000\ 0110\ 0110\ 0000)$, and thus third round output difference $\Delta O_3 = \pi_P(\Delta Y_3) = (0000\ 0110\ 0110\ 0000)$. This means that exactly S42 and S43 will be active in round 4.
An attacker can query say N pairs of plaintext-ciphertext with the given input difference ΔP. For each possible guess for the 8 bits of K_5 corresponding to the active sboxes S42 and S43 (i.e., bits 5-12), he can partially invert the K_5 key addition and invert S42 and S43 for every ciphertext. Then he counts how often the input difference for S42 and S43 matches with ΔO_3 for each key guess. The correct key guess should have a count of about Np, while bad key guesses should have a count of about $N2^{-8}$. We want the correct key guess count to be clearly higher than all bad key guess counts with high probability, so we recover the correct key guess by looking at the highest count. This holds if $Np > N2^{-8} + 4\sqrt{N2^{-8}}$, which is true for say $N = 6/p = 6/(1/32) = 192$ samples.

(c) Consider the boomerang with input plaintext difference

$$\Delta P = (0000\ 0000\ 0000\ 0100)$$

and output ciphertext difference

$$\Delta C = (0000\ 0000\ 0010\ 0000),$$

then a quartet $(P^{(1)}, P^{(2)}, P^{(3)}, P^{(4)})$ satisfies this boomerang if

$$P^{(1)} \oplus P^{(2)} = \Delta P, \quad P^{(3)} \oplus P^{(4)} = \Delta P, \quad \text{and}$$

$$C^{(1)} \oplus C^{(3)} = \Delta C, \quad C^{(2)} \oplus C^{(4)} = \Delta C.$$

Compute the total success probability of finding such quartets over all round 1 & 2 differentials with the given ΔP and all round 3 & 4 differentials with the given ΔC, i.e., compute

$$p = \left(\sum_{(\Delta P, \Delta O_1, \Delta O_2)} \Pr[(\Delta P, \Delta O_1, \Delta O_2)]^2 \right) \cdot \left(\sum_{(\Delta O_2, \Delta O_3, \Delta C)} \Pr[(\Delta O_2, \Delta O_3, \Delta C)]^2 \right)$$

6 points

(Hint: use the fact that in round 2 each Sbox has either input difference 0 or 1 (0001), so every *active* round 2 Sbox contributes a term

$$\sum_{(\Delta In = 4, \Delta Out \in \{0,\ldots,15\})} \Pr[(\Delta In, \Delta Out)]^2 = 2 \times (4/16)^2 + 1 \times (8/16)^2.$$

Likewise, in round 3 each active Sbox has output difference 2 (0010).)
Answer: First, each trail $\Delta P, \Delta O_1, \Delta O_2$ contributes the probability corresponding that both pairs $(P^{(1)}, P^{(2)})$ and $(P^{(3)}, P^{(4)})$ satisfy it: $\Pr[(\Delta P, \Delta O_1, \Delta O_2)]^2$.
If we just look at the $(\Delta O_1, \Delta O_2)$ part, then as hinted, each active sbox in round 2 has input difference 1. Note that for each active sbox in round 2, we need both pairs $(P^{(1)}, P^{(2)})$ and $(P^{(3)}, P^{(4)})$ to have the identical round 2 output difference ΔO_2 part, which has probability $A = 2 \times (4/16)^2 + 1 \times (8/16)^2 = 3/8$. If k sboxes are active in round 2 then the probability that ΔO_2 is the same for both pairs is $(3/8)^k$.
Now we can look at the first round: only S14 is active with input difference $4 =$ 0100. *Its DDT entries are:* $(4,6) = 8$, $(4,9) = (4,14) = 4$. *Since 6 and 9 imply 2 active sboxes in round 2 and 14 imply 3 active sboxes, the round 1-2 boomerang probability has 3 terms corresponding to these DDT entries and results in:*

$$\begin{aligned}
&\sum_{(\Delta P, \Delta O_1, \Delta O_2)} \Pr[(\Delta P, \Delta O_1, \Delta O_2)]^2 \\
&= \sum_{\Delta O_1} \left(\Pr[(\Delta P, \Delta O_1)]^2 \cdot \sum_{\Delta O_2} \Pr[\Delta O_2 | \Delta O_1]^2 \right) \\
&= \left(\frac{8}{16}\right)^2 A^2 + \left(\frac{4}{16}\right)^2 A^2 + \left(\frac{4}{16}\right)^2 A^3 = \frac{387}{8192}.
\end{aligned}$$

The analysis of round 3 goes analogously. Note that for each active sbox in round 3, we need both pairs $(C^{(1)}, C^{(3)})$ and $(C^{(2)}, C^{(4)})$ to have the identical round 3 input difference ΔI_3 part, which has probability $B = 4 \times (2/16)^2 + 1 \times (8/16)^2 = 5/16$.
Now we can look at the fourth round: only S44 is active with output difference $2 =$ 0010. *Its DDT entries are:* $(7,2) = 8$, $(10,2) = (11,2) = (14,2) = (15,2) = 2$. *Note that 10 implies 2 active sboxes in round 3, while 7, 11 and 14 imply 3 active sboxes and 15 implies 4 active sboxes. Thus the round 3-4 boomerang probability has 5 terms corresponding to these DDT entries and results in:*

$$\begin{aligned}
&\sum_{(\Delta O_2, \Delta O_3, \Delta C)} \Pr[(\Delta O_2, \Delta O_3, \Delta C)]^2 \\
&= \sum_{\Delta O_3} \left(\Pr[(\Delta O_3, \Delta C)]^2 \cdot \sum_{\Delta O_2} \Pr[\Delta O_2 | \Delta O_3]^2 \right) \\
&= \left(\frac{8}{16}\right)^2 B^3 + \left(\frac{2}{16}\right)^2 B^2 + 2 \times \left(\frac{2}{16}\right)^2 B^3 + \left(\frac{2}{16}\right)^2 B^4 = \frac{43025}{4194304}.
\end{aligned}$$

The boomerang success probability is thus $\frac{387}{8192} \cdot \frac{43025}{4194304} \approx 0.0004846$.