Fundamentals of Cryptography — Handout 7.

Yu Yu

Cryptographic Hash Functions.

1 Collision Resistant Hash Functions (CRHFs)

Informally, a collision resistant hash function is a function for which it is infeasible for PPT adversaries to find any collisions.

Definition 1 (Collision Resistant Hash Functions)**.** A (ensemble of) family of functions $\mathcal{G} \subseteq \{g : \{0,1\}^n \to \{0,1\}^{n-s(n)}\}$, is a collision resistant hash function (CRHF) if:

- Efficient: There exists a PPT Gen that on input 1^n produces a random element g of $\mathcal{G}$ as output, i.e., $g \leftarrow \mathsf{Gen}(1^n)$ where $g \overset{\$}{\leftarrow} \mathcal{G}$. Further, every $g \in \mathcal{G}$ is polynomial-time computable.
- Shrinking: The difference between input and output lengths, **shrinkage** (i.e. $s(n)$), is positive.
- Collision Resistant: For any probabilistic PPT A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that
$$\Pr_{g \overset{\$}{\leftarrow} \mathcal{G},\ (x,x') \leftarrow \mathsf{A}(1^n,g)} [\ x \neq x' \wedge g(x) = g(x')\ ] \ \leq\ \mathsf{negl}(n) \quad .$$

As slight abuse of notation, in the above we use g to refer to a function or the description of the function (the textbook uses s), which is obvious from the context.

Note that a hash function is public knowledge, and unlike many other cryptographic primitives there are no secrets.

Birthday Attacks. Assume that $g : \{0,1\}^n \to \{0,1\}^m$ $(n > m)$ behaves like a random function (i.e., $g(U_n)$ is close to U_m), there is a trivial attack. Namely, sample distinct random inputs x_1, …, x_q for $q = O(\sqrt{2^m})$ we have that with high probability (e.g., 1/2) we will get a collision $g(x_i) = g(x_j)$ for $x_i \neq x_j$. Refer to Section 4.6.3 and Appendix A.4 for more details. This means that any collision resistant hash function with output size m has security no more than $m/2$ bits.

Constructions of Collision Resistant Hash Functions. Despite the appealing property of collision resistance, cryptographers found out that it is impossible to construct CRHFs from one-way functions (or even one-way permutations) in a black box away. It is known we can base CRHFs on hard number theoretic problems such as RSA and Discrete Logarithm, but they are not efficient. In practice, cryptographers design CRHFs from scratch and compete for getting their designs standardized by organizations such as NIST. There are some known candidate[1] CRHFs such as the MD5, SHA-1, and SHA-3 standards. Finally, note that any of these standards are talking about a single function rather than a set of functions.

[1]Some designs are known not so secure as intended to be. Google "Xiaoyun Wang" for more details.

2 Universal One-way Hash Functions (UOWHFs)

A universal one-way hash function (UOWHF) has a similar but weaker guarantee of collision resistance. Now the adversary is given a specific point (considered as a target, which is independent of the hash functions) x and she needs to find another point x' such that $g(x) = g(x')$. We give the formal definition of UOWHF as below, where the only difference is the third condition, and it is easy to see that a CRHF is also a UOWHF.

Definition 2 (Universal One-way Hash Functions)**.** A (ensemble of) family of functions $\mathcal{G} \subseteq \{g : \{0,1\}^n \to \{0,1\}^{n-s(n)}\}$, is a universal one-way hash function (UOWHF) if:

- Efficient: There exists a PPT Gen that on input 1^n produces a random element g of $\mathcal{G}$ as output, i.e., $g \leftarrow \mathsf{Gen}(1^n)$ where $g \xleftarrow{\$} \mathcal{G}$. Further, every $g \in \mathcal{G}$ is polynomial-time computable.
- Shrinking: The difference between input and output lengths, **shrinkage** (i.e. $s(n)$), is positive.
- Target Collision Resistant (TCR): For for any $x \in \{0,1\}^n$, any PPT A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{g \xleftarrow{\$} \mathcal{G};\ x' \leftarrow \mathsf{A}(1^n, x, g)} [\ x \neq x' \wedge g(x) = g(x')\ ] \ \leq\ \mathsf{negl}(n) \quad .$$

We will often use an equivalent (although seemingly weaker) condition for target collision resistance.

- Target Collision Resistant 2 (TCR2): For any PPT A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n, g \xleftarrow{\$} \mathcal{G};\ x' \leftarrow \mathsf{A}(1^n, x, g)} [\ x \neq x' \wedge g(x) = g(x')\ ] \ \leq\ \mathsf{negl}(n) \quad .$$

Note that TCR property implies TCR2, so it suffices to prove the other direction. If $\mathcal{G} = \{g : \{0,1\}^n \to \{0,1\}^{n-s}\}$ has property TCR2, then the following UOWHF has property TCR. $\mathcal{G}' = \{g'(x) \overset{\text{def}}{=} g(x \oplus a) | g \in \mathcal{G}, a \in \{0,1\}^n\}$, i.e., $\mathcal{G}'$ is described by g and string a.

The good thing about UOWHF is that although weaker, it suffices for many applications and does not suffer from the birthday attacks, and it is possible to construct it from one-way functions. We will introduce the simplified case, namely, the construction of UOWHFs from one-way permutations by Naor and Yung.

Theorem 1 (UOWHFs from One-way Permutations)**.** *Let $f : \{0,1\}^n \to \{0,1\}^n$ be any (t, ε)-one-way permutation, let $\mathcal{H}$ be a family of universal hash permutations over $\{0,1\}^n$, i.e.,*

$$\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^n \mid h(y) \overset{\mathsf{def}}{=} h \cdot y, \quad \textit{where} \ \ y \in GF(2^n),\ \vec{0} \neq h \in GF(2^n)\ \} \ ,$$

let $\mathsf{trunc} : \{0,1\}^n \to \{0,1\}^{n-1}$ be a truncating function that outputs the first $n-1$ bits of input. Then, we have that

$$\mathcal{G} \overset{\mathsf{def}}{=} \{\ (\mathsf{trunc} \circ h \circ f\ ) : \{0,1\}^n \to \{0,1\}^{n-1} \mid h \in \mathcal{H}\ \}$$

is a family of $(t - n^{O(1)}, \varepsilon)$-universal one-way hash functions with 1 bit of shrinkage.

Proof. Suppose for contradiction that there exists a $\mathcal{G}$-collision finder A of running time $t - n^{O(1)}$ that on input (x, h), breaks the TCR2 with probability greater than ε, i.e.,

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n, h \xleftarrow{\$} \mathcal{H},\ x' \leftarrow \mathsf{A}(x,h)} [\ x \neq x' \wedge h(f(x))_{1,\ldots,n-1} = h(f(x'))_{1,\ldots,n-1}\ ] \ > \ \varepsilon$$

where $z_{1,\ldots,n-1}$ denotes the first $n-1$ bits of z. We define algorithm $\mathsf{Inv}^{\mathsf{A}}$ (that inverts f on input $y^* \in \{0,1\}^n$ by invoking A) as in Algorithm 1. We denote by $\mathcal{E}$ the event that $f(x) \neq y^*$.

Algorithm 1 $\mathsf{Inv}^{\mathsf{A}}$ that inverts f on input y^* using random coins (x, v).

```
Input: $y^* \xleftarrow{\$} \{0,1\}^n$ (note: $f(U_n)$ is identical to $U_n$ as $f$ is a permutation)
  Sample $x \xleftarrow{\$} \{0,1\}^n$
  if $f(x) = y^*$ then
    Output $x$ and terminate.
  end if
  let $h := (f(x) - y^*)^{-1} \cdot v$, where $v = \overbrace{0\ldots0}^{n-1}1$
  {note: The above implies $h(f(x))_{1,\ldots,n-1} = h(y^*)_{1,\ldots,n-1}$ due to the algebraic structure of $GF(2^n)$. }
  $x' \leftarrow \mathsf{A}(x, h)$
  if $f(x') = y^*$ then
    Output $x'$
  else
    Output $\perp$
  end if
  Terminate
```

By Claim 1.1, conditioned on $\mathcal{E}$ it is equivalent to consider that $\mathsf{Inv}^{\mathsf{A}}$ samples (x,h) from $\{0,1\}^n \times \mathcal{H}$ uniformly and independently, and then determine the value of y^*. Further, if $f(x) \neq y^*$, $x \neq x'$ (which implies $f(x) \neq f(x')$) and $h(f(x))_{1,\ldots,n-1} = h(f(x'))_{1,\ldots,n-1} = h(y^*)_{1,\ldots,n-1}$ then we must have $f(x') = y^*$ since $\mathsf{trunc} \circ h$ is strictly a 2-to-1 function. Therefore, $\mathsf{Inv}^{\mathsf{A}}$ inverts f with the following probability

$$\begin{aligned}
& \Pr_{y^* \leftarrow f(U_n),\ x \xleftarrow{\$} \{0,1\}^n} [\ f(\mathsf{Inv}^{\mathsf{A}}(y^*)) = y^*\ ] \\
\geq\ & \Pr_{x \xleftarrow{\$} \{0,1\}^n, y^* \xleftarrow{\$} \{0,1\}^n} [\ f(x) = y^*\ ] \\
& + \ \Pr[\mathcal{E}] \cdot \Pr_{x \xleftarrow{\$} \{0,1\}^n, h \xleftarrow{\$} \mathcal{H}, x' \leftarrow \mathsf{A}(x,h)} [\ h(f(x))_{1,\ldots,n-1} = h(f(x'))_{1,\ldots,n-1} \mid \mathcal{E}\ ] \\
\geq\ & 2^{-n} \ + \ (1 - 2^{-n})\varepsilon \ = \ \varepsilon + 2^{-n}(1-\varepsilon) > \varepsilon \quad ,
\end{aligned}$$

which completes the proof by reaching a contradiction. Note the inversion algorithm takes time t_{A} (the running time of A, which is $t - n^{O(1)}$) plus some light computation (which is accounted for by $n^{O(1)}$). □

Claim 1.1 (equivalent sampling)**.** *Let the values h, v, x, y^* be sampled as in Algorithm 1, then conditioned on the event $\mathcal{E}$ (i.e., $y^* \neq f(x)$), it is equivalent to sample $(x,\ h) \xleftarrow{\$} \{0,1\}^n \times \mathcal{H}$ uniformly and independently and then determine $y^* := f(x) - v \cdot h^{-1}$.*

Proof. We know that x is uniformly sampled from $\{0,1\}^n$ by definition, and thus it suffices to show that "fix any x, and conditioned on $\mathcal{E}$ (i.e., Y^* is uniform over $\{0,1\}^n \setminus \{f(x)\}$), it holds that h is uniform over $\mathcal{H}$". As $v \neq \vec{0}$, it follows that $h = (f(x) - Y^*)^{-1} \cdot v$ is uniform over $\{0,1\}^n \setminus \{\vec{0}\}$ which is exactly $\mathcal{H}$. Finally, for any given (x, h, v), one efficiently determines the value $y^* = f(x) - v \cdot h^{-1}$. □

3 Merkle-Damgård Domain Extension for CRHFs and UOWHFs

Similar to PRGs, once we get a PRG with small stretch, we can extend the range by sequential composition. Now we do the domain extension in a symmetric way for CRHFs and UOWHFs.

Lemma 2 (Merkle-Damgård Domain Extension for CRHFs). *Let*

$$\begin{aligned} & \mathcal{G} \subseteq \{g : \{0,1\}^n \to \{0,1\}^{n-s}\} \\ \textit{for each } g \qquad & (x_i, r_i) \mapsto (x_{i+1}) \text{ , where } x_i, x_{i+1} \in \{0,1\}^{n-s},\ r_i \in \{0,1\}^s \end{aligned}$$

be a (t,ε)-secure CRHF, and for any $q \in \mathbb{N}$ *define* $\mathcal{G}^q \subseteq \{g^q : \{0,1\}^{n+(q-1)\cdot s} \to \{0,1\}^{n-s}, g \in \mathcal{G}\}$ *where*

$$g^q(x_1, r_1, r_2, \ldots, r_q) = g(\ldots g(g(x_1, r_1), r_2), \ldots, r_q)$$

$x_i \in \{0,1\}^{n-s}$ *and* $r_i \in \{0,1\}^s$. *Then, we have that* $\mathcal{G}^q$ *is a* $(t - q \cdot t_g,\ \varepsilon)$*-secure CRHF, where* t_g *is the running time for computing function* g.

Proof. Suppose there exists A of running time $t - qt_g$ such that

$$\Pr_{g \xleftarrow{\$} \mathcal{G};\ (y,y') \leftarrow \mathsf{A}(1^n, g)} [\ y \neq y' \wedge g^q(y) = g^q(y')\ ] \ > \ \varepsilon \quad .$$

We define A' on input g as below:

- Invoke $(y, y') \leftarrow \mathsf{A}(g)$, where $y = (x_1, r_1, r_2, \ldots, r_q)$ and $y' = (x'_1, r'_1, r'_2, \ldots, r'_q)$.
- If $(x_1, r_1, r_2, \ldots, r_q) \neq (x'_1, r'_1, r'_2, \ldots, r'_q)$ and

$$g(\ldots g(g(x_1, r_1), r_2), \ldots, r_q) \ = \ g(\ldots g(g(x'_1, r'_1), r'_2), \ldots, r'_q)$$

  Then, there must exist $i \in [q]$[2] such that $(x_i, r_i) \neq (x'_i, r'_i)$ and $g(x_i, r_i) = g(x'_i, r'_i)$ where

$$x_i = g(\ldots g(g(x_1, r_1), r_2), \ldots, r_{i-1}) \quad \text{and } x'_i = g(\ldots g(g(x'_1, r'_1), r'_2), \ldots, r'_{i-1})$$

- Output (x_i, r_i) and (x'_i, r'_i).

It is easy to see that A' finds collision for g with the same probability (i.e., $> \varepsilon$) as A does for g^q, which completes the proof. □

[2] $[q] \stackrel{\text{def}}{=} \{1, \ldots, q\}$.

Lemma 3 (Merkle-Damgård Domain Extension for UOWHFs)**.** *Let*

$$\begin{array}{rl} & \mathcal{G} \subseteq \{g : \{0,1\}^n \to \{0,1\}^{n-s}\} \\ \textit{for each } g & (x_i, r_i) \mapsto (x_{i+1}) \ , \textit{where } x_i, x_{i+1} \in \{0,1\}^{n-s},\ r_i \in \{0,1\}^s \end{array}$$

be a (t,ε)-secure UOWHF, and for any $q \in \mathbb{N}$ define $\mathcal{G}^q \subseteq \{g^q : \{0,1\}^{n+(q-1)\cdot s} \to \{0,1\}^{n-s}, g_1, \ldots, g_q \in \mathcal{G}\}$ where

$$g^q(x_1, r_1, r_2, \ldots, r_q) = g_q(\ldots g_2(g_1(x_1, r_1), r_2), \ldots, r_q)$$

$x_i \in \{0,1\}^{n-s}$ and $r_i \in \{0,1\}^s$. Then, we have that $\mathcal{G}^q$ is a $(t - q \cdot t_g,\ q\varepsilon)$-secure UOWHF, where t_g is the running time for computing function g.
That is, in case of UOWHFs, we cannot[3] *just compose a single g with itself many times, but have to use independent g_i at every iteration.*

Proof. Now we use TCR (instead of TCR2) for our convenience. Suppose there exists some $y = (x_1, r_1, r_2, \ldots, r_q)$ and A of running time $t - qt_g$ such that

$$\Pr_{g_1 \xleftarrow{\$} \mathcal{G}, \ldots, g_q \xleftarrow{\$} \mathcal{G};\ y' \leftarrow \mathsf{A}(1^n, y, g_1, \ldots, g_q)} [\ y \neq y' \wedge g^q(y) = g^q(y')\ ] \ > \ q\varepsilon \quad .$$

where $y' = (x'_1, r'_1, r'_2, \ldots, r'_q)$. Now sample $i^* \xleftarrow{\$} [q]$, $g_1 \xleftarrow{\$} \mathcal{G}, \ldots, g_{i-1} \xleftarrow{\$} \mathcal{G}$,

$$x_{i^*} = g_{i^*-1}(\ldots g_2(g_1(x_1, r_1), r_2), \ldots, r_{i^*-1})$$

Now we define algorithm A' below that on $g \xleftarrow{\$} \mathcal{G}$ finds a collision for (x_{i^*}, r_{i^*}).

- Let i^*, $g_1, \ldots, g_{i^*-1}$ be sampled as above, sample $g_{i^*+1}, \ldots, g_q \xleftarrow{\$} \mathcal{G}$, let $g_{i^*} = g$.
- Invoke $y' \leftarrow \mathsf{A}(y, g_1, \ldots, g_q)$, where $y = (x_1, r_1, r_2, \ldots, r_q)$ and $y' = (x'_1, r'_1, r'_2, \ldots, r'_q)$.
- If $(x_1, r_1, r_2, \ldots, r_q) \neq (x'_1, r'_1, r'_2, \ldots, r'_q)$ and

  $$g_q(\ldots g_2(g_1(x_1, r_1), r_2), \ldots, r_q) \ = \ g_q(\ldots g_2(g_1(x'_1, r'_1), r'_2), \ldots, r'_q)$$

  Then, there must exist $i \in [q]$ such that $(x_i, r_i) \neq (x'_i, r'_i)$ and $g_i(x_i, r_i) = g_i(x'_i, r'_i)$ where

  $$x_i = g_{i-1}(\ldots g_2(g_1(x_1, r_1), r_2), \ldots, r_{i-1}) \quad \text{and } x'_i = g_{i-1}(\ldots g_2(g_1(x'_1, r'_1), r'_2), \ldots, r'_{i-1})$$

- If at the same time we have $i^* = i$ (which occurs with probability $1/q$), we get a collision for $g_{i^*} = g$, i.e., $(x_{i^*}, r_{i^*}) \neq (x'_{i^*}, r'_{i^*})$ and $g(x_{i^*}, r_{i^*}) = g(x'_{i^*}, r'_{i^*})$. Produce (x'_{i^*}, r'_{i^*}) as output.

Therefore,

$$\begin{aligned} & \Pr_{i^* \xleftarrow{\$} [q], g_1 \xleftarrow{\$} \mathcal{G}, \ldots,\ g_{i^*-1} \xleftarrow{\$} \mathcal{G};\ g \xleftarrow{\$} \mathcal{G}\ ,\ (x'_{i^*}, r'_{i^*}) \leftarrow \mathsf{A}'((x_{i^*}, r_{i^*}), g)} [\ (x_{i^*}, r_{i^*}) \ \neq \ (x'_{i^*}, r'_{i^*}) \wedge g(x_{i^*}, r_{i^*}) = g(x'_{i^*}, r'_{i^*})\ ] \\ \geq \ & \Pr_{g_1 \xleftarrow{\$} \mathcal{G}, \ldots, g_q \xleftarrow{\$} \mathcal{G};\ y' \leftarrow \mathsf{A}(1^n, y, g_1, \ldots, g_q)} [\ y \neq y' \wedge g(y) = g(y')\ ] \cdot \Pr[i^* = i] \\ > \ & q\varepsilon \cdot (1/q) = \varepsilon \quad . \end{aligned}$$

[3] By "cannot" I mean people don't have a proof for that, but let me know if you do.

It follows by an averaging argument that there exist some fixed values for $i^*, g_1, \ldots,\ g_{i^*-1}$, (and thus a fixed value for x_{i^*}, r_{i^*}), such that

$$\Pr_{g \xleftarrow{\$} \mathcal{G}\ ,\ (x'_{i^*}, r'_{i^*}) \leftarrow \mathsf{A}'((x_{i^*}, r_{i^*}), g)} [\ (x_{i^*}, r_{i^*})\ \neq\ (x'_{i^*}, r'_{i^*}) \wedge g(x_{i^*}, r_{i^*}) = g(x'_{i^*}, r'_{i^*})\ ]\ >\ \varepsilon$$

which completes the proof by reaching a contradiction. Note that $g_{i^*+1}, \ldots, g_q$ are the internal random coins of A', so we don't need to write them explicitly in presence of A'. $\square$

More Efficient Domain Extensions - the Merkle-Damgård Tree We can use the above technique to transform CRHFs/UOWHFs with small shrinkage into ones with large shrinkage. Once we get a length-halving CRHF/UOWHF $\mathcal{G} = \{g : \{0,1\}^{2m} \to \{0,1\}^m\}$, we can do domain extension more efficiently using the binary tree structure (symmetric to the GGM construction of PRF from PRG). See Section 4.6.4 from the KL book for the case of CRHFs (the case of UOWHFs are quite similar but have to use independent g at each level of the tree).

Applications of Cryptographic Hash Functions Hash functions have a wide variety of applications, such as digital signatures (another topic of modern cryptography), secure fingerprinting, and key derivation function, etc.

Homework 6.

Exercise 1. Reprove the generalized version of Theorem 1, where the shrinkage is s bits instead of a single bit. That is, let $f : \{0,1\}^n \to \{0,1\}^n$ be any $(t,\ \varepsilon)$-one-way permutation, let $\mathcal{H}$ be a family of universal hash permutations over $\{0,1\}^n$, i.e.,

$$\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^n \mid h(y) \overset{\mathsf{def}}{=} h \cdot y, \ \text{ where }\ y \in GF(2^n),\ \vec{0} \neq h \in GF(2^n)\ \}\ ,$$

let $\mathsf{trunc} : \{0,1\}^n \to \{0,1\}^{n-s}$ be a truncating function that outputs the first $n-s$ bits of input. Then, show that for small value of s,

$$\mathcal{G} \overset{\mathsf{def}}{=} \{\ (\mathsf{trunc} \circ h \circ f\ ) : \{0,1\}^n \to \{0,1\}^{n-s} \mid h \in \mathcal{H}\ \}$$

is a family of $(t',\ \varepsilon')$-universal one-way hash functions with s bits of shrinkage. Give the proof quantitatively (t' and ε' should be functions of t, ε, s.).

We show that $\mathcal{G}$ is a $(t - n^{O(1)},\ 2^s \cdot \varepsilon)$-UOWHF (where for small value of s the term $2^s\varepsilon$ is negligible if ε is) as below.

Proof. Suppose for contradiction that there exists a $\mathcal{G}$-collision finder A of running time $t - n^{O(1)}$ that on input (x, h), breaks the TCR2 with probability greater than ε, i.e.,

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n, h \xleftarrow{\$} \mathcal{H},\ x' \leftarrow \mathsf{A}(x,h)} [\ x \neq x' \wedge h(f(x))_{1,\ldots,n-s} = h(f(x'))_{1,\ldots,n-s}\ ]\ >\ 2^s \cdot \varepsilon$$

where $z_{1,\ldots,n-s}$ denotes the first $n-s$ bits of z. We define algorithm $\mathsf{Inv}^{\mathsf{A}}$ (that inverts f on input $y^* \in \{0,1\}^n$ by invoking A) as in Algorithm 2. We denote by $\mathcal{E}$ the event that $f(x) \neq y^*$.

By Claim 3.1, conditioned on $\mathcal{E}$ it is equivalent to consider that $\mathsf{Inv}^{\mathsf{A}}$ samples (x,h,v) from $\{0,1\}^n \times \mathcal{H} \times \mathcal{V}$ uniformly and independently (see the definition of $\mathcal{V}$ in Algorithm 2), and then determine the value of y^*. Note that A takes as input only (x, h) (i.e., independent of v). Therefore, conditioned on $\mathcal{E}$ and and that A finds a collision $x \neq x' \wedge h(f(x))_{1,\ldots,n-s} = h(f(x'))_{1,\ldots,n-s}$, we

Algorithm 2 $\mathsf{Inv}^{\mathsf{A}}$ that inverts f on input y^* using random coins (x, v).

```
Input: $y^* \xleftarrow{\$} \{0,1\}^n$ (note: $f(U_n)$ is identical to $U_n$ as $f$ is a permutation)
  Sample $x \xleftarrow{\$} \{0,1\}^n$
  if $f(x) = y^*$ then
    Output $x$ and terminate.
  end if
  let $h := (f(x) - y^*)^{-1} \cdot v$, where $v \xleftarrow{\$} \mathcal{V} \overset{\text{def}}{=} \{v \in \{0,1\}^n : v_{1,\ldots,n-s} = 0^{n-s} \wedge v \neq 0^n\}$
  {note: The above implies $h(f(x))_{1,\ldots,n-s} = h(y^*)_{1,\ldots,n-s}$ due to the algebraic structure of $GF(2^n)$. }
  $x' \leftarrow \mathsf{A}(x, h)$
  if $f(x') = y^*$ then
    Output $x'$
  else
    Output $\perp$
  end if
  Terminate
```

have that y^* is uniform over set $\{y^* := f(x) - v \cdot h^{-1}, v \in \mathcal{V}\}$ of size $|\mathcal{V}| = 2^s - 1$. Recall that $f(x')$ is also a member of the set, and thus $y^* = f(x')$ occurs with probability $1/(2^s - 1)$.

Therefore, $\mathsf{Inv}^{\mathsf{A}}$ inverts f with the following probability

$$\begin{aligned}
& \Pr_{y^* \leftarrow f(U_n),\ x \xleftarrow{\$} \{0,1\}^n} [\ f(\mathsf{Inv}^{\mathsf{A}}(y^*)) = y^*\ ] \\
\geq\ & \Pr_{x \xleftarrow{\$} \{0,1\}^n, y^* \xleftarrow{\$} \{0,1\}^n} [\ f(x) = y^*\ ] \\
& + \Pr[\mathcal{E}] \cdot \Pr_{x \xleftarrow{\$} \{0,1\}^n, h \xleftarrow{\$} \mathcal{H}, x' \leftarrow \mathsf{A}(x,h)} [\ h(f(x))_{1,\ldots,n-s} = h(f(x'))_{1,\ldots,n-s} \wedge y^* = f(x') \mid \mathcal{E}\ ] \\
\geq\ & 2^{-n} + (1 - 2^{-n}) \cdot (2^s \cdot \varepsilon) \cdot \frac{1}{2^s - 1} > \varepsilon \cdot \frac{1 - 2^{-n}}{1 - 2^{-s}} \geq \varepsilon \quad ,
\end{aligned}$$

which completes the proof by reaching a contradiction. Note the inversion algorithm takes time t_{A} (the running time of A, which is $t - n^{O(1)}$) plus some light computation (which is accounted for by $n^{O(1)}$). □

Claim 3.1 (equivalent sampling)**.** *Let the values h, v, x, y^* be sampled as in Algorithm 2, then conditioned on the event $\mathcal{E}$ (i.e., $y^* \neq f(x)$), it is equivalent to sample $(x, h, v) \xleftarrow{\$} \{0,1\}^n \times \mathcal{H} \times \mathcal{V}$ uniformly and independently and then determine $y^* := f(x) - v \cdot h^{-1}$.*

Proof. We know that (x, v) is uniformly sampled from $\{0,1\}^n \times \mathcal{V}$ by definition, and thus it suffices to show that "fix any (x, v), and conditioned on $\mathcal{E}$ (i.e., Y^* is uniform over $\{0,1\}^n \setminus \{f(x)\}$), it holds that h is uniform over $\mathcal{H}$". As $v \neq \vec{0}$, it follows that $h = (f(x) - Y^*)^{-1} \cdot v$ is uniform over $\{0,1\}^n \setminus \{\vec{0}\}$ which is exactly $\mathcal{H}$. Finally, for any given (x, h, v), one efficiently determines the value $y^* = f(x) - v \cdot h^{-1}$. □

An alternative proof. We can also give a simpler proof based on the statement of Theorem 1.

That is, suppose for contradiction we have

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n, h \xleftarrow{\$} \mathcal{H},\ x' \leftarrow \mathsf{A}(x,h)} [\ x \neq x' \wedge h(f(x))_{1,\ldots,n-s} = h(f(x'))_{1,\ldots,n-s}\ ] \ > \ 2^s \cdot \varepsilon$$

Then condition on $x \neq x' \wedge h(f(x))_{1,\ldots,n-s} = h(f(x'))_{1,\ldots,n-s}$ the probability that $h(f(x))_{1,\ldots,n-1} = h(f(x'))_{1,\ldots,n-1}$ is $1/2^{s-1}$. Therefore,

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n, h \xleftarrow{\$} \mathcal{H},\ x' \leftarrow \mathsf{A}(x,h)} [\ x \neq x' \wedge h(f(x))_{1,\ldots,n-1} = h(f(x'))_{1,\ldots,n-1}\ ] \ > \ 2^s \cdot \varepsilon / 2^{s-1} \ \geq \ \varepsilon$$

which is a contradiction to Theorem 1 and thus completes the proof.