MAC’s and Hash Functions

Prof. Zeph Grunschlag
Message Authentication Codes

AUTHENTICATION

PROBLEM: Alice (A) sends Bob (B) a message through insecure channel modifiable by Mallory (M)

GOAL: A encodes message in a way that enables B to detect any modification
Message Authentication Codes

**AUTHENTICATION**

SOLUTION: A appends **MAC** tag T which changes randomly whenever message is changed.

Use **hash function** to create the tag T from the message.

contradiction!!!
Desired Properties

Hash function $h$ should satisfy:

• Output $<$ Input

• Collision resistant:
  
  Finding different inputs with same output is computationally intractable

• One Way:
  
  Easy to compute, hard to find pre-images

• Over insecure channel: secret keys
Hash Function Family

A HASH FAMILY is a 4-tuple \((X, \mathcal{Y}, \mathcal{K}, \mathcal{H})\) satisfying
1. \(X\) is a (possibly infinite) set of MESSAGES
2. \(\mathcal{Y}\) is a finite set of possible TAGS (or digests)
3. \(\mathcal{K}\) is a finite set of possible KEYS
4. \(\mathcal{H}\) is a finite set of hash functions indexed by \(\mathcal{K}\)
   so for each \(K \in \mathcal{K}\) there is a function \(h_K : X \to \mathcal{Y}\)
Discrete Log Hash

Note: Picking specific values of message-size, tag-size, key-size. Could generalize to arbitrary but related sizes.

- 260 to 132 bit contractor:
  \[ X = [0, 2^{260} - 1], Y = [0, 2^{132} - 1] \]

- \( K \)-keys \( K = (p, q, \alpha, \beta) \) should satisfy:
  - \( p \) and \( q \) are prime with \( p = 2q + 1 \)
  - \( \alpha \neq \beta \) are primitive in \( \mathbb{Z}_p^* \)
  - Bit-lengths: \(|p| = 132, |q^2| = 261\)

- \( H \)-hash functions defined by
  \[ h_K(n) = \alpha \left\lfloor \frac{n}{q} \right\rfloor \beta^n \mod q \mod p \]
Primitive Elements

DEF: An element $g$ in a group $G$ is said to be **primitive** (or a **generator**) if every $g^i$ element in $G$ can be expressed in the form for some **exponent index** $i$. If $G$ contains a primitive element, $G$ is said to be by **cyclic**.

NOTE: Equivalently, $g$ is primitive if the first positive index for which $g^i = 1$ is $i = n = |G|$. 

THM: If $F$ is a finite field, then $F^*$ is cyclic.

COR: If $p$ is prime, $\mathbb{Z}_p^*$ is cyclic. Also, suppose $g$ is primitive in $\mathbb{Z}_p^*$. Then $g^i$ is primitive iff $i$ is relatively prime to $p - 1$. 
Index Calculus

Can figure out everything about how numbers *multiply* in \( \mathbb{Z}_p^* \) by seeing how their exponents (indices) *add* in \( \mathbb{Z}_\phi(p)^+ \). Generalization:

**THM:** If \( p \) is a prime number, then there is an isomorphism: \( \mathbb{Z}_p^* \approx \mathbb{Z}_\phi(p)^+ \).

**NOTE:** Isomorphism only easy to compute in (index) \( \rightarrow \) (number) direction. Other direction (number) \( \rightarrow \) (index) is DLog problem.
Discrete Logarithm Problem

DEF: Suppose that \( y = x^a \mod n \). Then \( a \) is said to be the **discrete logarithm** of \( y \) with base \( x \) modulo \( n \). Notation: \( a = D\log_x(y) \mod n \)

Discrete logarithm assumption: No BPP algorithm \( D(x,y,p) \) exists which successfully computes \( D\log_x(y) \mod p \) with “significant” probability given a random prime \( p \), a random primitive \( x \) in \( \mathbb{Z}_p^* \), and a random integer \( y \) in \( \mathbb{Z}_p^* \).

When all factors of \( p-1 \) are small, algorithms do exist: explains defining \( (p-1)/2 \) to be prime.
Computational Security of Logarithmic Hash


Note: Computational complexity definitions require considering infinite family of log hashes where allow arbitrarily large domains.

LEMMA: Collision resistance implies one-wayness when domain >> codomain.

COR: Discrete log hash “is” one way.
Collisions $\Rightarrow$ DLog

INPUT: $p$ - prime, $x,y \in \mathbb{Z}_p^*$ with $x$ primitive

OUTPUT: $D\log_x(y) \mod p$

EXTERNAL: FindCollision - assumed procedure for finding collisions in $h_K$

1. if $\frac{p - 1}{2}$ not prime, or $y$ not primitive “FAIL”

2. $q = \frac{p - 1}{2}, \alpha = x, \beta = y, K = (p, q, \alpha, \beta)$

3. $(a, b) = \text{FindCollision}(K)$

4. ... continued next page ...
Collisions $\Rightarrow$ DLog

$$i = \left\lfloor \frac{a}{q} \right\rfloor - \left\lfloor \frac{b}{q} \right\rfloor, \quad j = (b \mod q) - (a \mod q)$$

$$i = i \mod (p - 1), \quad j = j \mod (p - 1)$$

while( $i \mod 2 == 0$ ) {

    $$i = i / 2, \quad j = j / 2$$

    if ( $(i - j) \mod 2 \neq 0$ ) $j = j + (p - 1) / 2$

}

return $\left\lfloor i \cdot (j^{-1} \mod (p - 1)) \right\rfloor \mod (p - 1)$
Iterated Hashes

- A procedure for repeatedly applying a particular hash function, shrinking arbitrarily long messages to fixed length tags.

**EXAMPLE (Simple Merkle-Damgård):**

- Assume $h$ takes 260 bits to 132 bits and that $0^{132}$ is never an output. Discrete log hash (viewed on bitstring) satisfies these.

- Define buffer function $b$ - a 1-1 function from bitstrings of length < 132 to bitstrings of length exactly 132.
Simple Merkle-Damgård

INPUT: bitstring \( x = x_1 x_2 \ldots x_k \)

OUTPUT: bitstring \( y = y_1 y_2 \ldots y_{132} \)

EXTERNAL: compression function \( h \)

//Break up into 128-bit blocks:

\[
\text{for } i \in [1, \left\lfloor \frac{k}{128} \right\rfloor] \quad z_i = x_{128i+1} \ldots x_{128i+128}
\]

\[
z_{i+1} = b(x_{128i+1} \ldots x_k) \quad \text{// buffer}
\]

\[
n = 0^{132}
\]

// for each block \( z_i \)

\[
z = n || z_i \quad \text{// concatenate strings}
\]

\[
n = h(z) \quad \text{// view } z \text{ as a number}
\]

return \( n \)
Security of Simple Merkle-Damgård

THM: Any BPP algorithm to find collisions in the Simple Merkle-Damgård applied to a contraction function $h$, would imply a BPP algorithm for finding collisions in $h$.

COR: If the discrete logarithm assumption holds, Simple Merkle-Damgård with $h$ chosen from discrete log hash family is a secure hash family with arbitrary contraction.

NOTE: theoretically secure but impractical.