#include <iostream>

#include <sstream>

#include "test\_helper.h"

#include "sha1.h"

TEST(MacAttackTests, playground) {

SHA1 sha1;

sha1.set\_iv("f4b645e89faaec2ff8e443c595009c16dbdfba4b");

std::string original = std::string( message\_1.begin(), message\_1.end() );

std::string extension = "P. S. Give Vaishnavi 100%";

uint64\_t message\_2\_bit\_size = extension.length() \* 8;

uint64\_t message\_1\_bit\_size = original.length() \* 8;

EXPECT\_EQ( 376, message\_1\_bit\_size );

uint64\_t key\_length = 128;

uint64\_t length\_size = 64;

uint64\_t message\_1\_padding = 1024 - (message\_1\_bit\_size + key\_length + length\_size);

uint64\_t message\_1\_full\_length = key\_length + message\_1\_bit\_size + message\_1\_padding + length\_size; // 376 + 128 + 456 + 64

uint64\_t length = message\_1\_full\_length + message\_2\_bit\_size;

sha1.update( extension );

std::cout << "Padding: " << message\_1\_padding << std::endl;

EXPECT\_EQ( 456 , padding(message\_1\_padding).length() );

EXPECT\_EQ( '1' , padding(message\_1\_padding)[0] );

EXPECT\_EQ( '0' , padding(message\_1\_padding)[1] );

std::ostringstream sever\_message\_base;

sever\_message\_base

<< string\_to\_binary\_string(original)

<< padding(message\_1\_padding + key\_length + message\_1\_bit\_size )

// I think this needs to be the ( original message || key ) in bits for size:

<< gen\_length( message\_1\_bit\_size + key\_length );

std::ostringstream server\_message;

server\_message

<< sever\_message\_base.str()

<< string\_to\_binary\_string(extension);

std::cout << "Using hacked length: " << length << std::endl;

std::string digest = sha1.final( length );

std::cout << "Hex: " << binary\_string\_to\_hex\_string( server\_message.str() ) << std::endl;

std::cout << digest << std::endl;

EXPECT\_EQ( 456, message\_1\_padding );

EXPECT\_EQ( 64, gen\_length(123).length() );

EXPECT\_EQ( message\_1\_padding, padding(message\_1\_padding).length() );

}

TEST(MacAttackTests, sha1\_works) {

SHA1 sha1;

sha1.update("abcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabc");

EXPECT\_EQ( "284f2b4b1a934e36ef357c41b84cb28492b6696d" , sha1.final() );

}