CS 380S - 0x1A Great Papers in Computer Security
Fall 2012

Homework #2

Due: 2pm CDT (in class), October 25, 2012

NO LATE SUBMISSIONS WILL BE ACCEPTED

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Sciences department code of conduct can be found at `http://www.cs.utexas.edu/academics/conduct/`.

Homework #2 (30 points)

Problem 1

Problem 1a (2 points)

Explain why enforcement of the same origin policy in modern browsers fundamentally relies on the integrity of the DNS (Domain Name System).

Problem 1b (2 points)

One solution for enforcing the same origin policy even when DNS may be compromised is to associate the server's public key with each Web object retrieved over HTTPS. When one Web object attempts to access another object, the browser checks whether the keys match.

Does this solution provide complete enforcement of the same origin policy, or can it still be subverted for certain Web objects by an attacker who controls DNS? Explain.

Problem 2

The Molvanîan Institute of Cryptology proposes the following variant of RSA encryption. Recall that an RSA public key is a pair (N, e). To encrypt some message m, first generate a fresh random value r of the same length as m. Use r as if it were a one-time pad to encrypt m (*i.e.*, let $s = m \oplus r$), and then encrypt r using plain RSA (*i.e.*, let $t = r^e \mod N$). The ciphertext is the (s, t) pair.

Problem 2a (2 points)

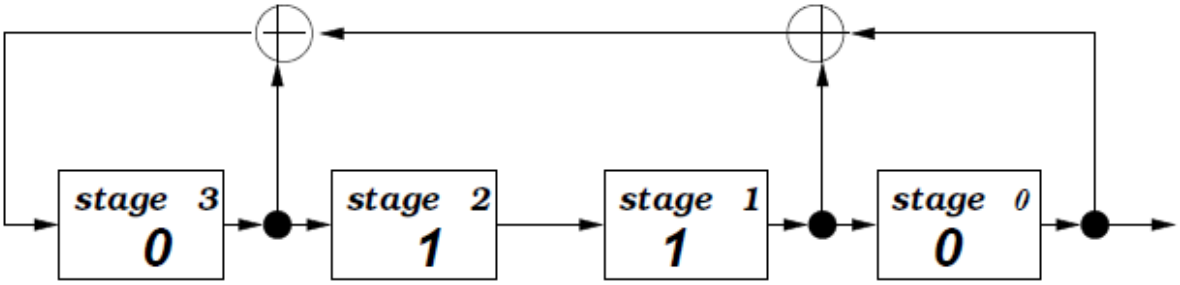
How does decryption work in this scheme?

Problem 2b (4 points)

Is this encryption scheme semantically secure? Explain.

Problem 3 (4 points)

Molvanîan Telecom encrypts voice-over-IP datagrams using a stream cipher based on the following pseudo-random generator:



The generator is initialized with the key, the first 4 bits of the keystream are discarded, and the rest is used for encryption.

Suppose you want to completely break this cipher (*i.e.*, recover the key) using a known-plaintext attack. How long should your plaintext be? How would the attack proceed (describe all details)?

Problem 4

The Yahalom protocol is a 3-party key establishment and authentication protocol between Alice (A), Bob (B) and a trusted Server (S). In the beginning of the protocol, A and S share the symmetric key K_{as}, B and S share the symmetric key K_{bs}. The objective of the protocol is to (i) create a fresh symmetric key K_{ab} between A and B, and (ii) authenticate A to B. The protocol proceeds as follows:

$$
\begin{array}{llcll}
1. & A & \rightarrow & B & A,\ N_a \\
2. & B & \rightarrow & S & B,\ \{A, N_b, N_a\}_{K_{bs}} \\
3. & S & \rightarrow & A & \{B, K_{ab}, N_a, N_b\}_{K_{as}},\ \{A, K_{ab}\}_{K_{bs}} \\
4. & A & \rightarrow & B & \{A, K_{ab}\}_{K_{bs}},\ \{N_b\}_{K_{ab}}
\end{array}
$$

A starts the protocol by generating a fresh random number (nonce) N_a, and sends N_a to B together with A's identity.

B generates his own nonce N_b, creates a message consisting of A's identity and the two nonces, encrypts it with the symmetric key K_{bs} and sends it to S together with his identity.

S decrypts the message and learns A's identity and the nonces. S then generates a fresh symmetric key K_{ab} for A and B, and sends a two-part message to A. In the first part, the new key K_{ab} is encrypted together with B's identity and the nonces by the symmetric key K_{as}, which is shared between S and A. In the second part, K_{ab} is encrypted together with A's identity by the symmetric key K_{bs}, which is shared between S and B (but not A).

Finally, A decrypts the first part of the message from S and learns K_{ab}. A then forwards the second part (which he cannot decrypt) to B. To prove that A successfully decrypted the first part and learned the new key K_{ab}, A also sends B's own nonce back to B encrypted with K_{ab}.

Problem 4a (10 points)

The first objective of the Yahalom protocol is *secrecy*. No party other than A, B, or S should be able to learn the new key K_{ab}.

Write a Murφ model of the Yahalom protocol, specify the secrecy property for K_{ab} as a state invariant, and verify whether it holds for every reachable state of your finite-state model.

Hint: consider more than one concurrent session of the protocol, and investigate intruder rules that ignore message formats (*e.g.*, the attacker may put two nonces in a message even if only one nonce is expected by the recipient).

Problem 4b (6 points)

The second objective of the Yahalom protocol is *authentication*. If B thinks that he is talking to A after the protocol has completed, then it should always be the case that he is indeed talking to A.

Specify the authentication property as a state invariant, add it to the model you implemented for Problem 4a, and verify whether it holds for the Yahalom protocol.

Downloading and running Murφ

Download Murφ from `http://mclab.di.uniroma1.it/software_cmurphi.html`
Murφ tutorial and user manual are in the `doc` directory. The Murφ model of the Needham-Schroeder public-key protocol is in `ex/secur/ns-old.m`

Tips:

- Always run your Murφ model with the `-ndl` option to make sure verification does not stop once a deadlock state is reached (typically, deadlock is not an issue for security protocols).

- Scalarsets are enumerated types. They are similar to `enum` types in C++. For example, type declaration `AgentId:scalarset(5)` says that variables of type `AgentId` can take one of 5 (unnamed) values. Scalarset types are simply finite, unordered sets of values.

  If you write `ruleset i: ScalarType` where `ScalarType` is a Scalarset type, then the enclosed rule will be executed once for every possible value of `ScalarType`. For each execution, the current value can be accessed as `i`.

- Multisets are data structures. A multiset is a set in which the same element may be included more than once. Multisets are similar to arrays, except that they are unordered.

  If you write `multisetcount(m:mset,P)` where `mset` is declared as a multiset and `P` is a predicate (i.e., a function returning true or false), then `P` will be applied to every element of the multiset `mset`, and the return value of `multisetcount` will be the number of elements of `mset` on which `P` evaluated to true. Within the body of `P`, the current element (i.e., the element to which the predicate `P` is being applied) can be accessed as `mset[m]`.

Submission instructions

Combine your Murφ models and text descriptions of the results (*e.g.*, text files containing Murφ printouts of attack traces) into a single `tar` file, and send it as an email attachment to `mgeorgiev@utexas.edu` with *CS 380S hw#2* in the subject line.