Ph 20 Assignment 7

Alexander Zlokapa

1 Threat Model

A sample scenario is as follows: a terrorist organization seeks to disrupt flight schedules at a major airport, creating air traffic that messes up people's lives in general, ultimately causing economic and emotional distress. (These terrorists, although still detrimental to society, are relatively benign terrorists as far as terrorists go.) Since the airport obtains flight information from airlines, there are a few potential sites of attack:

1. Change flight data at the point of release: this would have maximal overall impact since it affects all airports, it would be targeted at a single airline.

2. Change flight data at the point of receipt: this would affect an entire airport, but it could knock out a major city rather than having a diffused effect across a country.

Obviously, the terrorists do not have legal authority for such an operation, since they are corrupting data without assent of the data provider or recipient.

2 RSA Overview

RSA uses the computational difficulty of solving modular exponentiation for large numbers in order to keep data secure. The overall procedure is as follows, in a hypothetical situation where Bob wants to send encrypted information to Alice:

1. Alice and Bob both generate private and public keys for themselves using two large prime numbers

2. Alice sends her public key (made of the integer pair (n, e) as defined by RSA) to Bob over a public communication channel

3. Bob applies an agreed-upon padding scheme on his plaintext message to yield m, which he then encrypts using Alice's public key to produce the cipher $c = m^e \mod n$. He transmits c to Alice.

4. Alice decrypts the message with her private key, recovering $m = c^d \mod n$. She then reverses the padding scheme to see the original plaintext message.

Given sufficient computational power or insufficiently large public/private keys, RSA becomes trivial to decrypt via a Montgomery reduction algorithm. Hence, it is necessary to use large keys that are definitely primes in order to prevent successful attacks.

Additionally, if no padding scheme is used, RSA is not semantically secure, i.e. given two plaintexts and two ciphers, an attacker would be able determine which cipher corresponds to which plaintext.

Thus, an attacker may be able to exploit the system by encrypting his own plaintexts with the public key to try to learn how to decrypt them, ultimately using his knowledge on the actual cipher he wishes to break. Consequently, padding schemes such as PKCS7 have been creating, making the cipher semantically secure.

3 RSA Demonstration

Implementing the above algorithm (without a padding scheme), Alice generated a random public key ($n = 194081, e = 17$) and a corresponding private key ($n = 194081, p = 2273$). Bob encrypted his message ($m = 137$) using her public key, yielding the cipher ($c = 1374$). Eve then tried every integer from 0 to 1,000,000 with a collision and a preimage attack, trying to find which original message would yield the same ciphertext when encoded with Alice's public key. She found that the numbers $(137, 194218, 388299, 582380, 776461, 970542)$ all produced the same cipher. However, due to the cyclical nature of $x \mod n$, the difference between possible plaintext messages is exactly equal to Alice's n in her public key. Consequently, Eve concluded that the lowest of these numbers, 137, must be Bob's original message. In the scenario mentioned in Section 1, Eve could use this information to discover, for instance, the password of an airline's main database, and then use the information to alter database records.

4 GnuPG Overview

GNU Privacy Guard implements symmetric-key cryptography and public-key cryptography (including RSA and DSA) to encrypt and sign data and communications. Due to its generic nature, it is a very versatile tool that can be used in many situations, such as in transferring flight data in the above sample scenario. However, GnuPG is a command-line system, and thus is difficult to incorporate into other software, making it less likely to be the actual tool used by an airport to communicate and protect flight data. Because GnuPG attempts to implement algorithms in highly efficient ways, security can be compromised. For instance, the sliding window method for exponentiation formerly used by GnuPG created the leakage of exponent bits, allowing full key recovery for all 1024-bit RSA keys and even an eighth of 2048-bit keys. Additionally, one month ago (October 2017), the ROCA vulnerability — a weakness that can reveal the private key by taking advantage of faulty key generation methods that use only a subset of sufficiently large prime numbers — affected many keys, making existing encryptions susceptible to attack.

5 Proof of Work

To demonstrate proof of work with gpg, we simply generated a fingerprint for the public key "mgrudich" by entering `gpg -fingerprint "mgrudich"`, yielding the following output:

```
pub rsa4096 2017-11-03 [SC] [expires:  2021-11-03]
DACF 5DB9 6175 1BF6 34EA 096B 8EC0 0E6B AE88 D5DC
uid [ unknown] Michael Grudić <mgrudich@caltech.edu>
sub rsa4096 2017-11-03 [E] [expires:  2021-11-03]
```

6 Authentication Mechanism

A secure authentication mechanism must have a private key — whether a text password or an RSA private key — that is kept secret. If used for communication, the authentication mechanism must have a protocol agreed upon by more than one party, allowing information to be transferred. In such cases, the existence of a public key would allow outside users to encrypt information, leaving it impossible for anyone except holders of the private key to decrypt the message. Even so, one easy way to compromise the cipher despite not knowing the private key is a preimage attack, where a person encrypts multiple random messages with the public key until they find a plaintext message that produces the same cipher as the one they wish to break. Other attack schemes exist, depending on the encryption system, including collision attacks (finding two messages that produce the same hashed cipher) and brute force attacks (exhaustively trying different possible private keys).

7 Conclusion

Although mathematically impossible to break with current computational capabilities, cryptosystems such as RSA still suffer from vulnerabilities caused by practical considerations, such as the inability to generate truly random prime numbers and the existence of information leakage, whether through measuring the time to decrypt a given cipher (and thus form conclusions about the private key) or reading extra bits during modular exponentiation. Ultimately, these practical flaws make truly impervious encryption systems difficult to achieve, allowing hackers in the above scenario to find points of exploitation in the airline/airport systems even if RSA or a different mathematically unbreakable cryptosystem is used. Thus, even with theoretically secure cryptography protocols in place, we find that vulnerabilities remain, creating a potential risk for damage in scenarios such as the one posed above.