How cryptography is used to secure web services

Josh Benaloh
Cryptographer
Microsoft Research

Transfer of Confidential Data

You (client) Merchant (server)

I want to make a purchase.

What is your Credit Card Number?

My Credit Card is 6543 2345 6789 8765.

Transfer of Confidential Data

♦ But the Internet provides no privacy.
♦ Is there any way to protect my data from prying eyes at intermediate nodes?

Symmetric Encryption

♦ If the user has a pre-existing relationship with the merchant, they may have a shared secret key $K$ – known only to the two parties.
♦ User encrypts private data with key $K$.
♦ Merchant decrypts data with key $K$.

Asymmetric Encryption

♦ What if the user and merchant have no prior relationship?
♦ Asymmetric encryption allows me to encrypt a message for a recipient without knowledge of the recipient’s decryption key.

The Fundamental Equation

$E = mc^2$
The Fundamental Equation

\[ Z = Y^X \mod N \]

When \( Z \) is unknown, it can be efficiently computed.

When \( X \) is unknown, the problem is known as the \textit{discrete logarithm} and is generally believed to be hard to solve.

When \( Y \) is unknown, the problem is known as \textit{discrete root finding} and is generally believed to be hard to solve … without the factorization of \( N \).

The problem is not well-studied for the case when \( N \) is unknown.

How to compute \( Y^X \mod N \)

- Compute \( Y^X \) and then reduce \( \mod N \).
- If \( X, Y, \) and \( N \) each are 1,000-bit integers, \( Y^X \) consists of \( \sim 2^{1010} \) bits.
- Since there are roughly \( 2^{250} \) particles in the universe, storage is a problem.
How to compute $Y^X \mod N$

- Repeatedly multiplying by $Y$ (followed each time by a reduction modulo $N$) $X$ times solves the storage problem.

- However, we would need to perform $\sim 2^{900}$ 32-bit multiplications per second to complete the computation before the sun burns out.

Multiplication by Repeated Doubling

To compute $X \times Y$, compute $Y, 2Y, 4Y, 8Y, 16Y, \ldots$ and sum up those values dictated by the binary representation of $X$.

Example: $26Y = 2Y + 8Y + 16Y$.

Exponentiation by Repeated Squaring

To compute $Y^X$, compute $Y, Y^2, Y^4, Y^8, Y^{16}, \ldots$ and multiply those values dictated by the binary representation of $X$.

Example: $Y^{26} = Y^2 \cdot Y^8 \cdot Y^{16}$.

We can now perform a 1,000-bit modular exponentiation using $\sim 1,500$ 1,000-bit modular multiplications.

- 1,000 squarings: $y, y^2, y^4, \ldots, y^{2^{1000}}$
- $\sim 500$ “ordinary” multiplications

The Fundamental Equation

$Z = Y^X \mod N$

When $Y$ is unknown, the problem is known as discrete root finding and is generally believed to be hard to solve … without the factorization of $N$.

RSA Encryption/Decryption

- Select two large primes $p$ and $q$.
- Publish the product $N = pq$.
- The exponent $X$ is typically fixed at 65537.

- Encrypt message $Y$ as $E(Y) = Y^X \mod N$.
- Decrypt ciphertext $Z$ as $D(Z) = Z^{1/X} \mod N$.

- Note $D(E(Y)) = (Y^X)^{1/X} \mod N = Y$. 

Microsoft Confidential 3
RSA Signatures and Verification

- Not only is \( D(E(Y)) = (Y^{X})^{1/X} \mod N = Y \), but also \( E(D(Y)) = (Y^{1/X})^{X} \mod N = Y \).

- To form a signature of message \( Y \), create \( S = D(Y) = Y^{1/X} \mod N \).

- To verify the signature, check that \( E(S) = S^{X} \mod N \) matches \( Y \).

Transfer of Confidential Data

**You (client)**
- I want to make a purchase.
  - Here is my RSA public key \( E \).
  - My Credit Card is \( E(6543\ 2345\ 6789\ 8765) \).

**Merchant (server)**
- What is your Credit Card Number?
- My Credit Card is 6543 2345 6789 8765.

Intermediary Attack

**You (client)**
- I want to make a purchase.
  - My public key is \( E \).

**Intermediary**
- My public key is \( E \).

**Merchant (server)**
- I want to make a purchase.
  - \( E(CC#) \)
  - \( E(CC#) \)

Digital Certificates

“Alice’s public modulus is \( N_A = 331490324840 \ldots \)"

-- signed … someone you trust.
Replay Attack

You (client)  
I want to make a purchase.

Here is my RSA public key $E$ and a cert.

My Credit Card is $E(6543 \ 2345 \ 6789 \ 8765)$.

Later …

Eavesdropper  
I want to make a different purchase.

Here is my RSA public key $E$ and a cert.

My Credit Card is $E(6543 \ 2345 \ 6789 \ 8765)$.

Transfer of Confidential Data

You (client)  
I want to make a purchase.

Here is my RSA public key $E$ and a cert and a “nonce”.

My Credit Card and your nonce are $E(6543 \ 2345 \ 6789 \ 8765)$, nonce.

SSL/PCT/TLS History

- 1994: Secure Sockets Layer (SSL) V2.0
- 1995: Private Communication Technology (PCT) V1.0
- 1996: Secure Sockets Layer (SSL) V3.0
- 1997: Private Communication Technology (PCT) V4.0
- 1999: Transport Layer Security (TLS) V1.0

SSL/PCT/TLS

You (client)  
Let’s talk securely.

Here are the protocols and ciphers I understand.

I choose this protocol and these ciphers.

Here is my public key, a cert, a nonce, etc.

Using your public key, I’ve encrypted a random symmetric key (and your nonce).

Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

- Block ciphers
- Stream ciphers

SSL/TLS

All subsequent secure messages are sent using the symmetric key and a keyed hash for message authentication.
Block Ciphers

DES, AES, RC2, RC5, etc.

Usually 8 or 16 bytes.

Block Cipher Modes

Electronic Code Book (ECB) Encryption:

Block Cipher

Plaintext

Ciphertext

Block Cipher

Block Cipher

Block Cipher

Block Cipher

Ciphertext

Electronic Code Book (ECB) Decryption:

Inverse Cipher

Inverse Cipher

Inverse Cipher

Inverse Cipher

Plaintext

Ciphertext

Cipher Block Chaining (CBC) Encryption:

IV

Block Cipher

Block Cipher

Block Cipher

Block Cipher

Plaintext

Ciphertext

Cipher Block Chaining (CBC) Decryption:

IV

Inverse Cipher

Inverse Cipher

Inverse Cipher

Inverse Cipher

Plaintext

Ciphertext

Block Cipher Integrity

With ECB mode, identical blocks will have identical encryptions.

This can enable replay attacks as well as reorderings of data. Even a passive observer may obtain statistical data.
Stream Ciphers

RC4, SEAL, etc.

- Use the key as a seed to a pseudo-random number-generator (PRNG).
- Take the stream of output bits from the PRNG and XOR it with the plaintext to form the ciphertext.

Stream Cipher Encryption

Plaintext: ****************************
PRNG(seed): ****************************
Ciphertext: ****************************

Stream Cipher Integrity

- It is easy for an adversary (even one who can’t decrypt the ciphertext) to alter the plaintext in a known way.
Bob to Bob’s Bank:
  Please transfer $0,000,002.00 to the account of my good friend Alice.

One-Way Hash Functions

The idea of a check sum is great, but it is designed to prevent accidental changes in a message.
For cryptographic integrity, we need an integrity check that is resilient against a smart and determined adversary.

MD4, MD5, SHA-1, SHA-256, etc.

A one-way hash function is a function

\[ H : \{0,1\}^* \rightarrow \{0,1\}^k \]

(typically \( k \) is 128 or 160)
such that, given an input value \( x \), one can’t find \( x' \neq x \) such that \( H(x) = H(x') \).
One-Way Hash Functions

There are many measures for one-way hashes.

- Non-invertability: given \( y \), it’s difficult to find any \( x \) such that \( H(x) = y \).

- Collision-intractability: one cannot find a pair of values \( x \neq x' \) such that \( H(x) = H(x') \).

One-Way Hash Functions

- When using a stream cipher, a hash of the message can be appended to ensure integrity. [Message Authentication Code]

- When forming a digital signature, the signature need only be applied to a hash of the message. [Message Digest]