Secure Password Storage

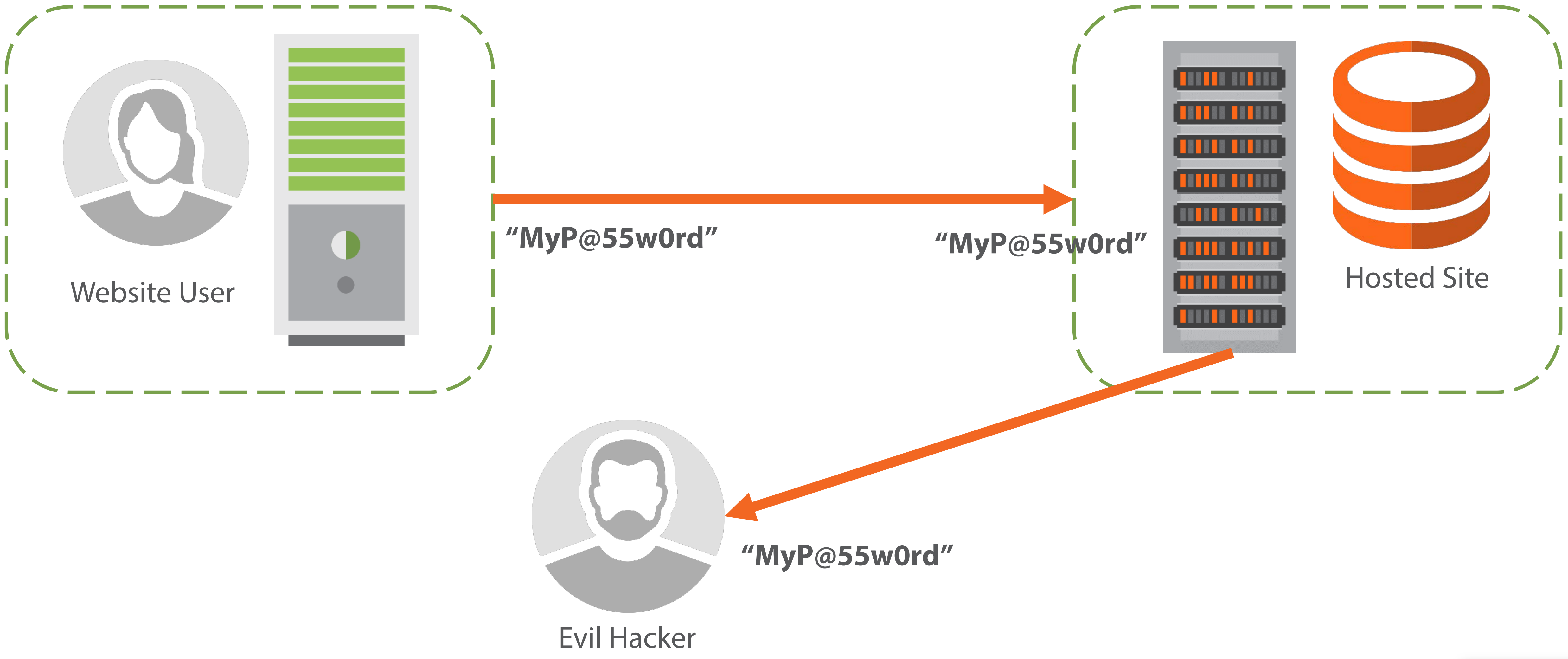
Stephen Haunts

@stephenhaunts | www.stephenhaunts.com

Overview

- Storing passwords in the clear
- Encrypting passwords
- Using hashes to store passwords
- Using salted hashes
- Using a password based key derivation function

Storing Passwords in the Clear

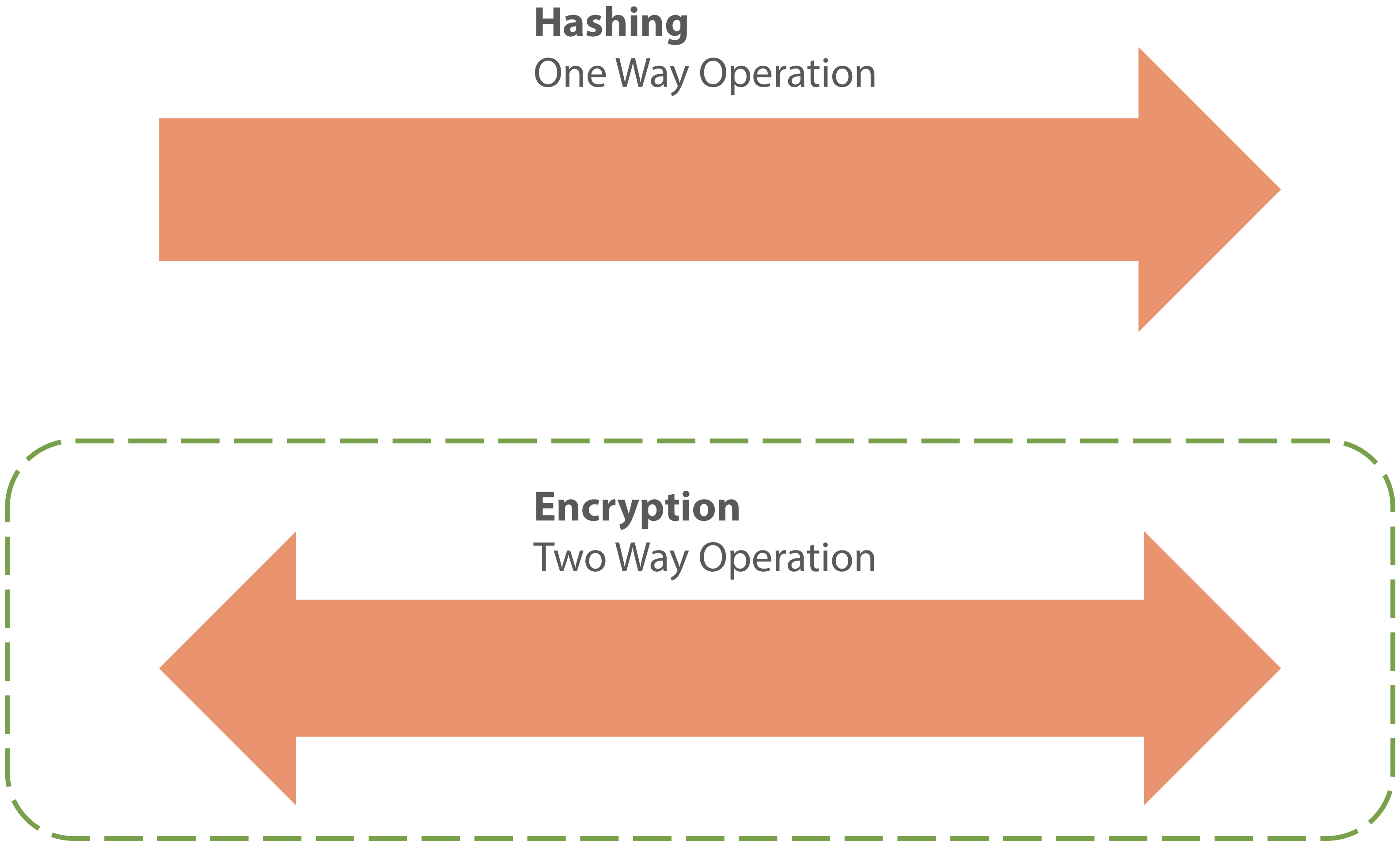


Storing Passwords in the Clear

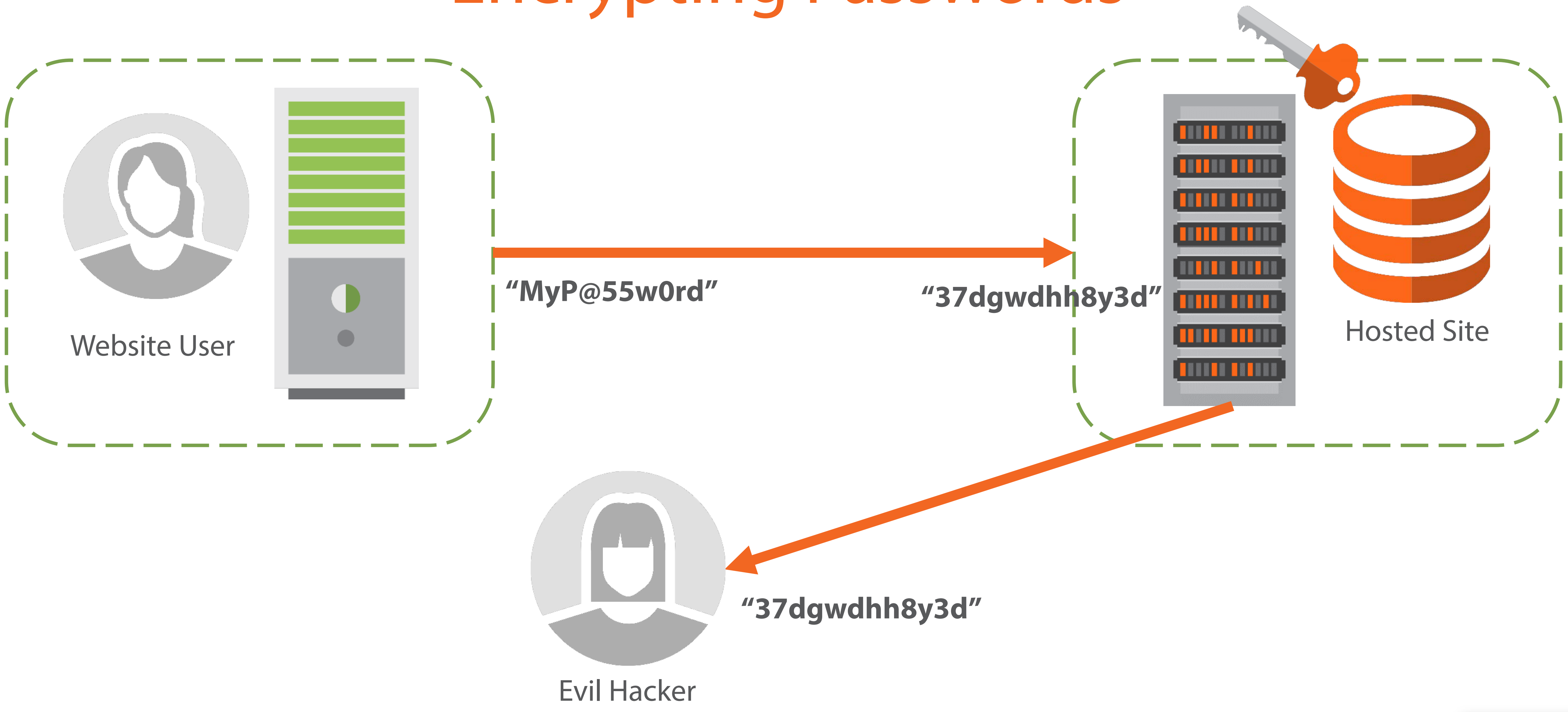
Evil Hacker

- Financial loss
- Reputational damage
- Legal action
- Loss of market share
- Regulatory fines

Encrypting Passwords



Encrypting Passwords

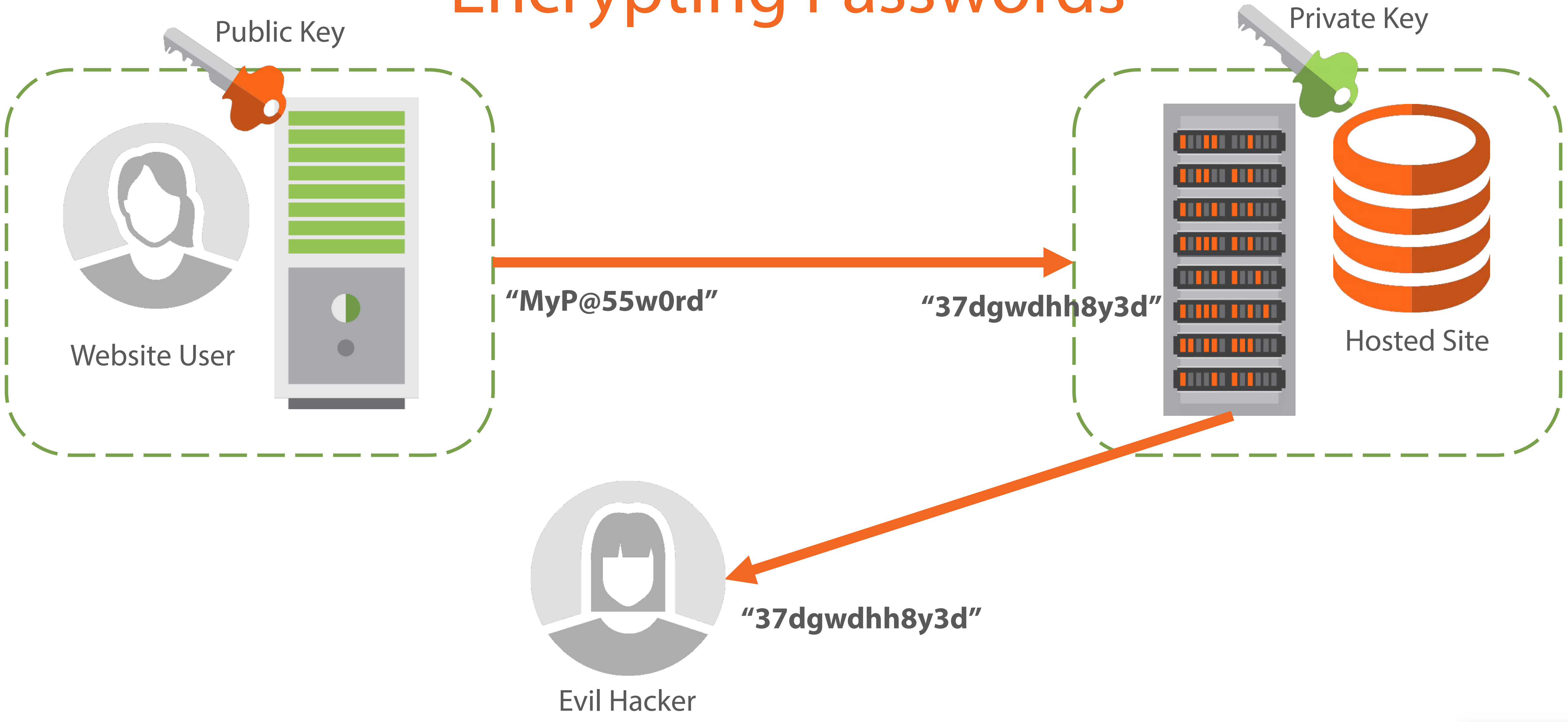


Encrypting Passwords

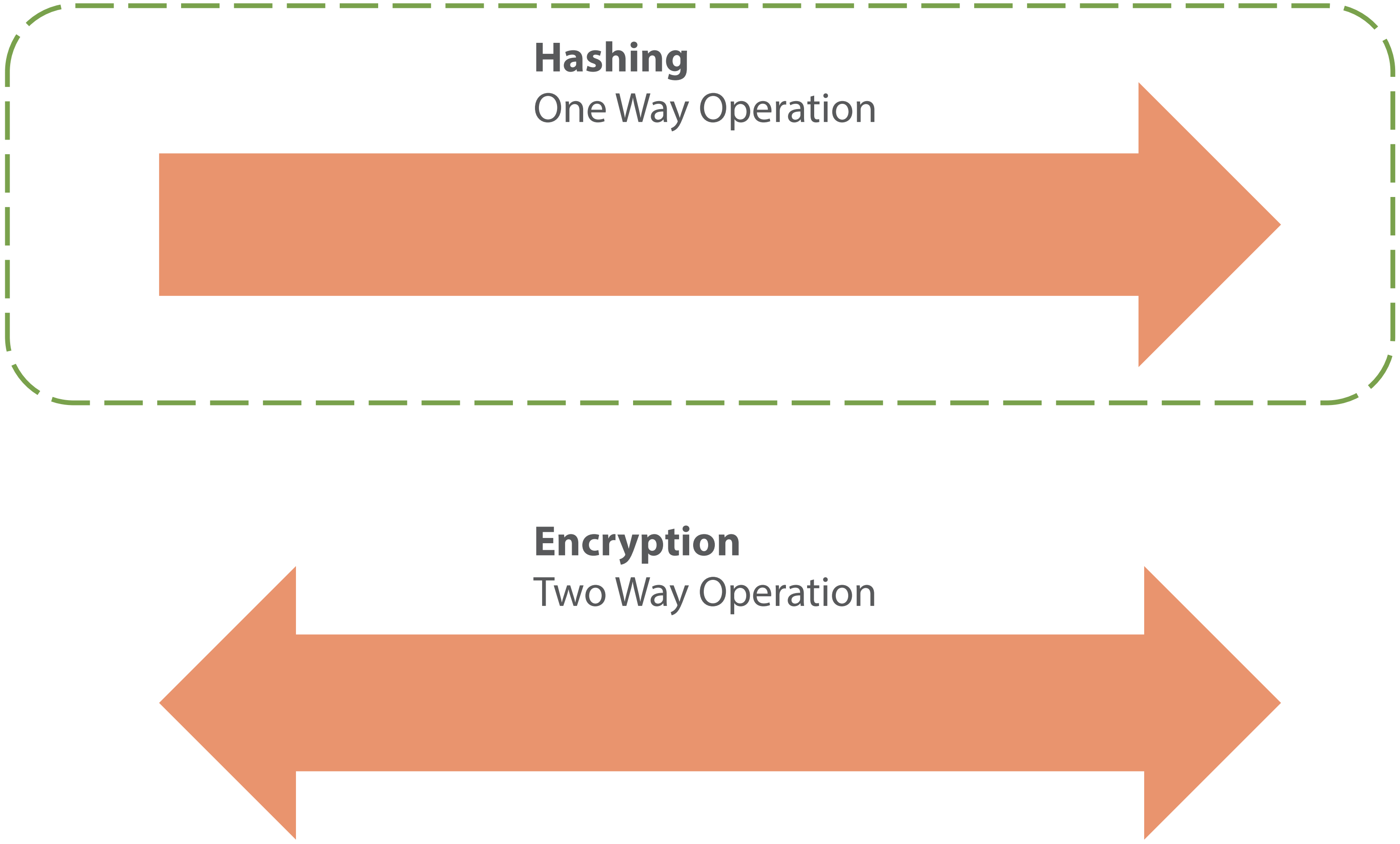
Evil Hacker

- Key management and storage
- Compromised keys

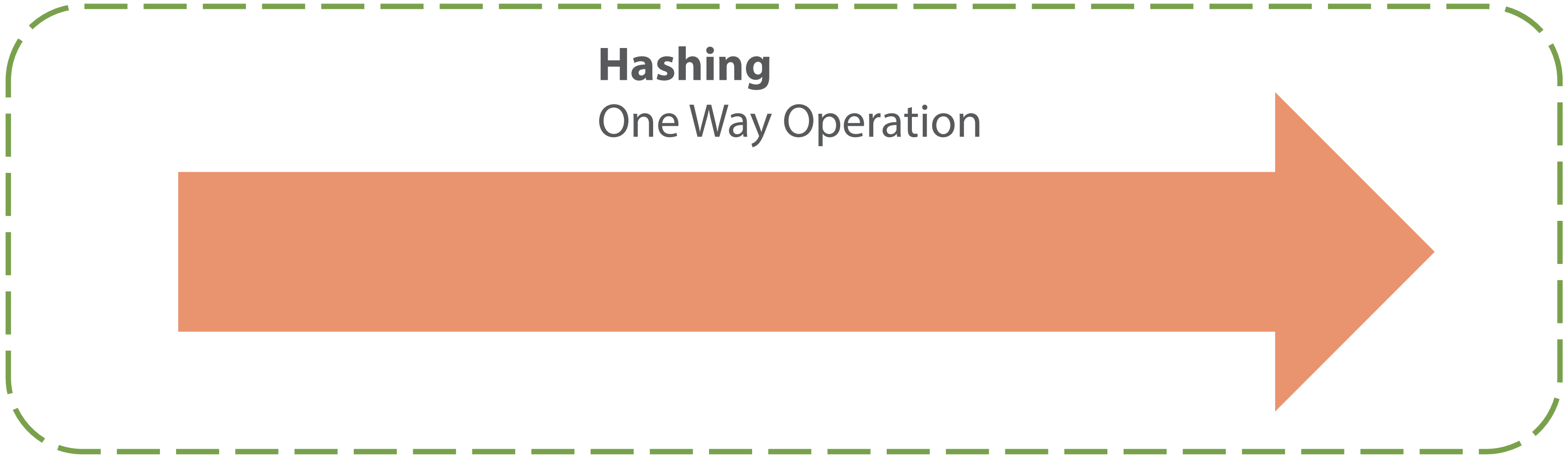
Encrypting Passwords



Using Hashes to Store Passwords



Using Hashes to Store Passwords



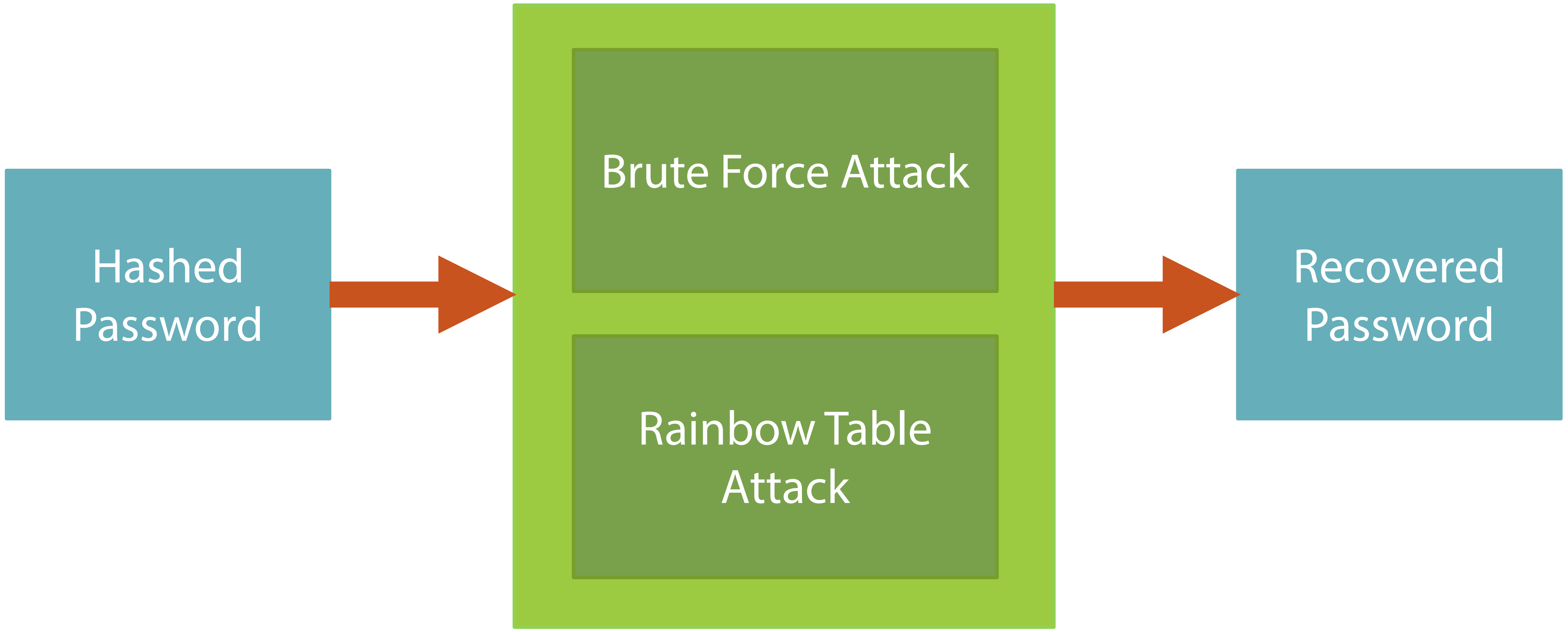
It is easy to compute the hash value for any given message

It is infeasible to generate a message that has a given hash

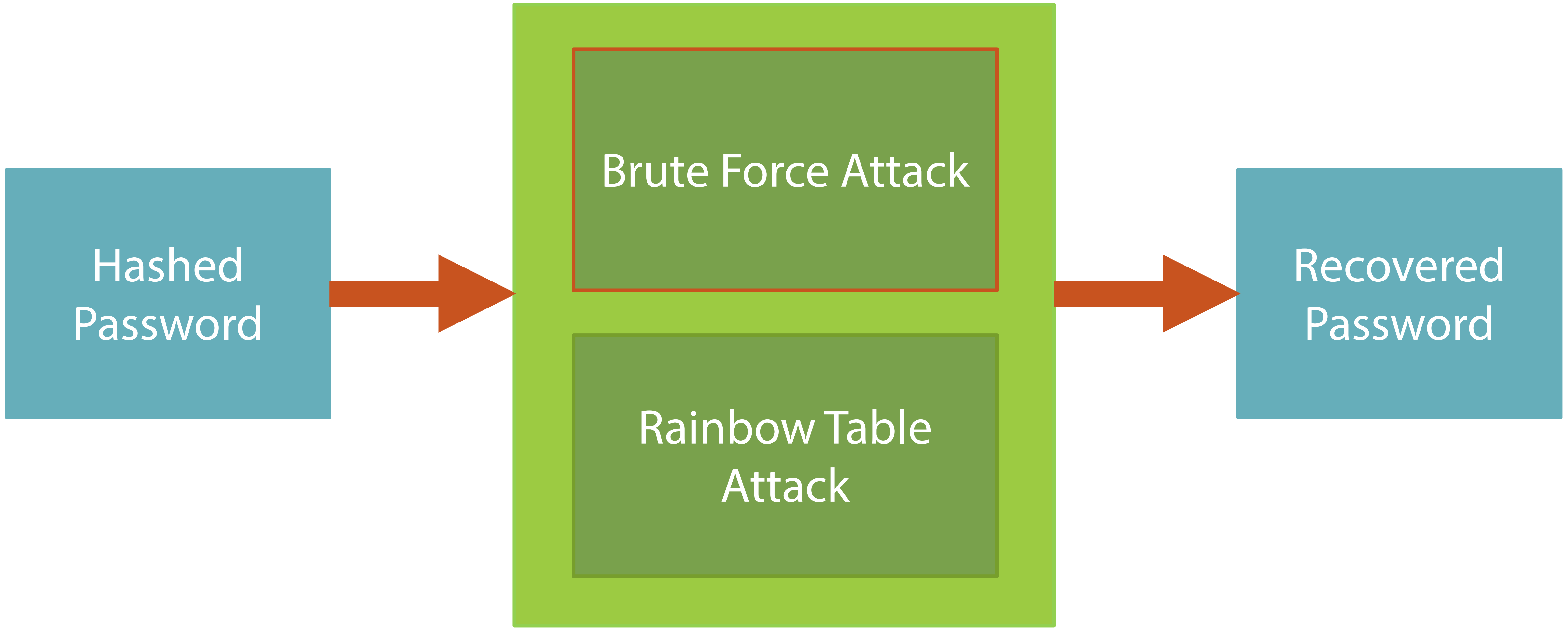
It is infeasible to modify a message without changing the hash

It is infeasible to find two different messages with the same hash

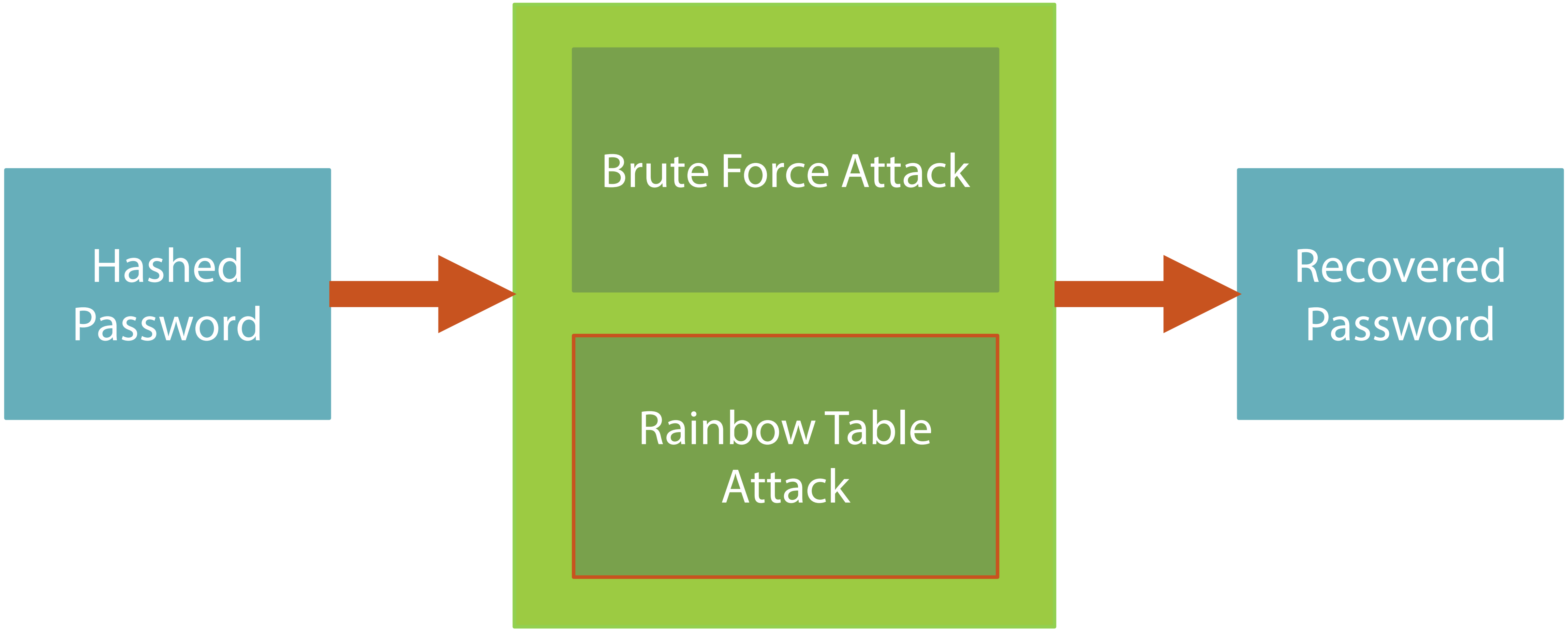
Using Hashes to Store Passwords



Using Hashes to Store Passwords



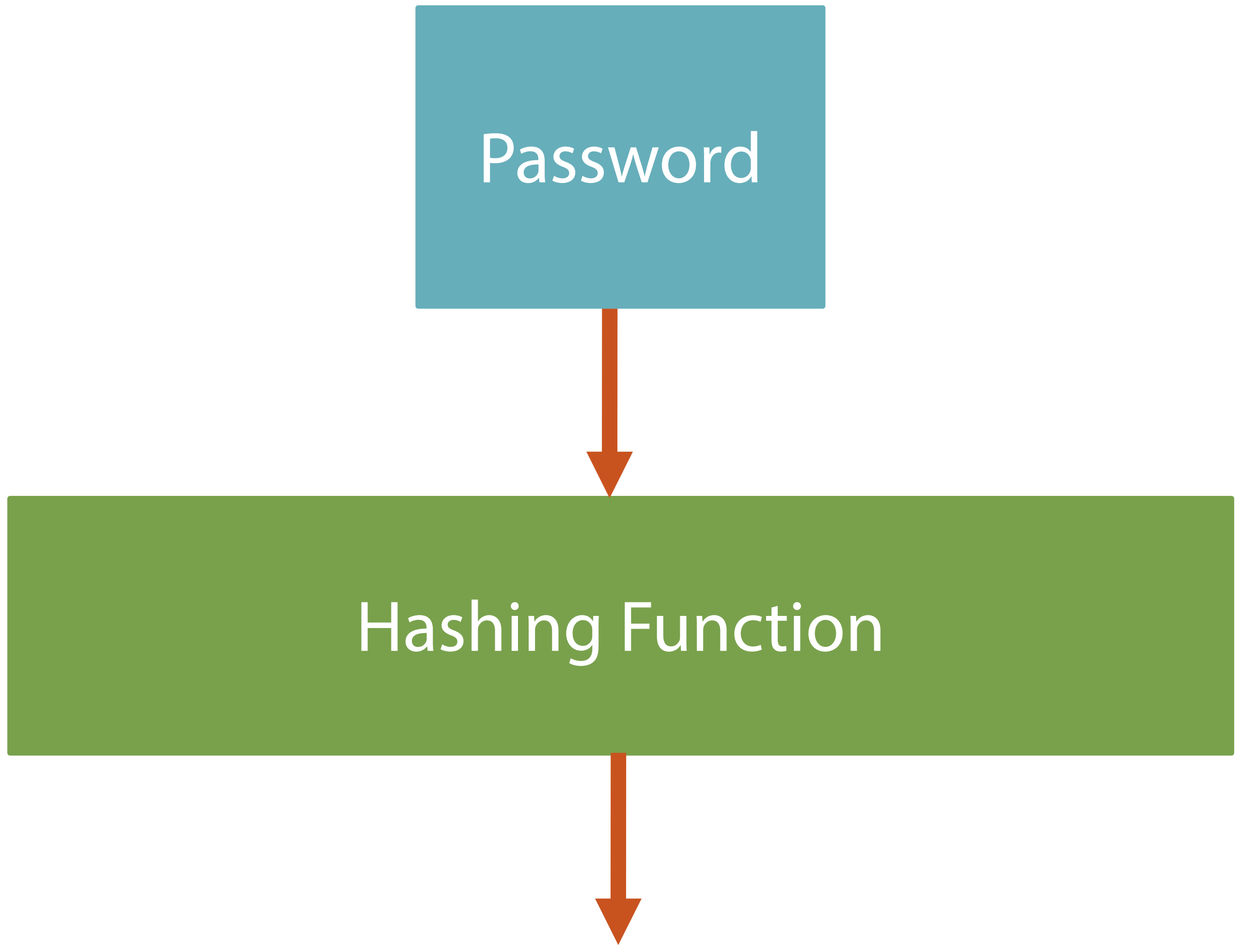
Using Hashes to Store Passwords



Demo

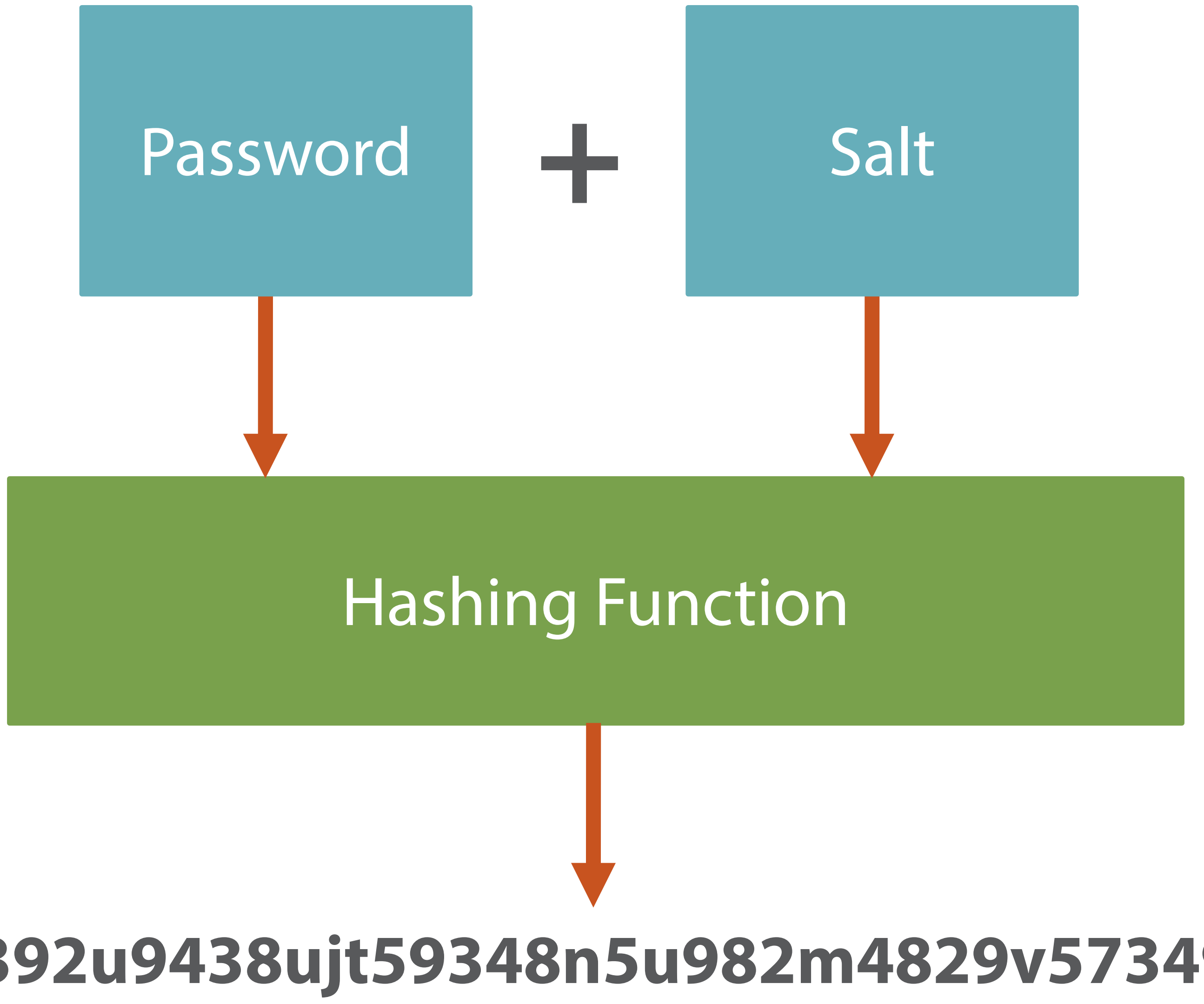
Using Rainbow Tables to Reverse Hashes

Using Salted Hashes to Store Passwords



32287fwyudvherryn238nyr4v5v7n8c4wr7w98n

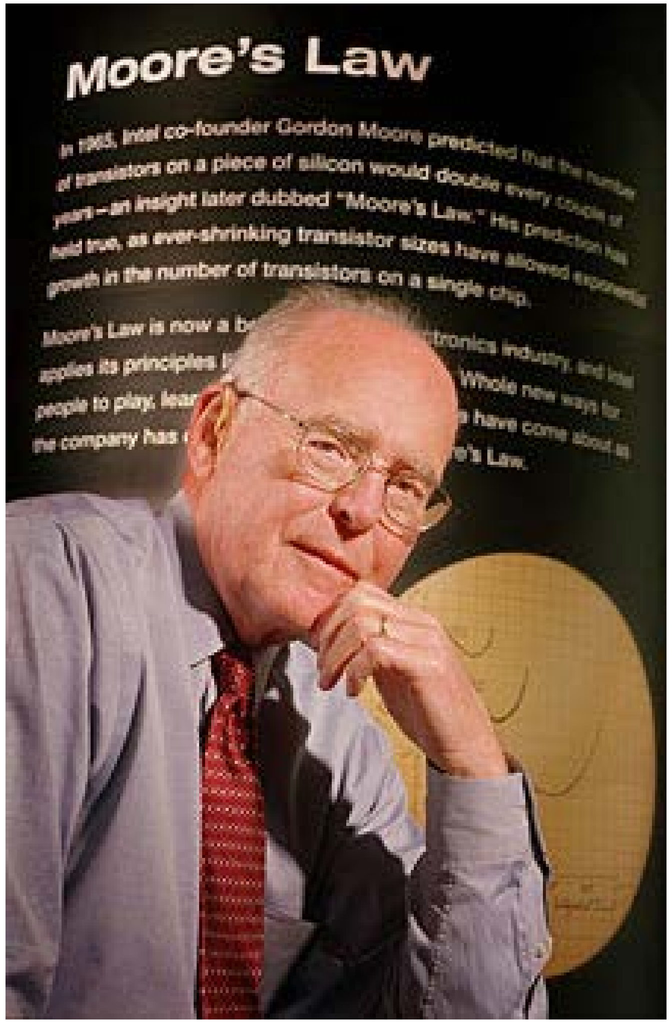
Using Salted Hashes to Store Passwords



Code Demo

Hashing Passwords with a Salt

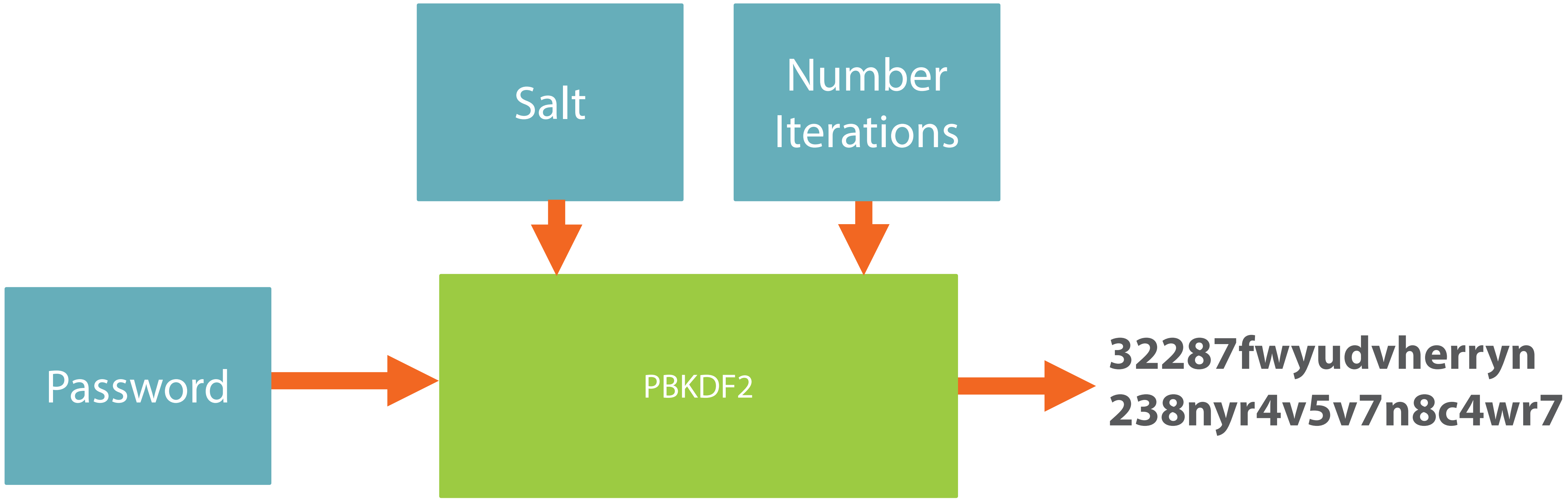
Password Based Key Derivation Functions


Moore's Law
In 1965, Intel co-founder Gordon Moore predicted that the number of transistors on a piece of silicon would double every couple of years—an insight later dubbed "Moore's Law." His prediction has held true, as ever-shrinking transistor sizes have allowed exponential growth in the number of transistors on a single chip.
Moore's Law is now a b
tronics industry, and Intel
applies its principles
Whole new ways for
people to play, lear
have come about as
the company has
e's Law.

Password Based Key Derivation Functions

- Password Based Key Derivation Function (PBKDF2)
- RSA Public Key Cryptographic Standards (PKCS #5 Version 2.0)
- Internet Engineering Task Force RFC 2898 Specification

Password Based Key Derivation Functions



Password Based Key Derivation Functions

- Good default is 50,000 iterations
- Balance number of iterations with acceptable performance
- Ideally double number of iterations every 2 years

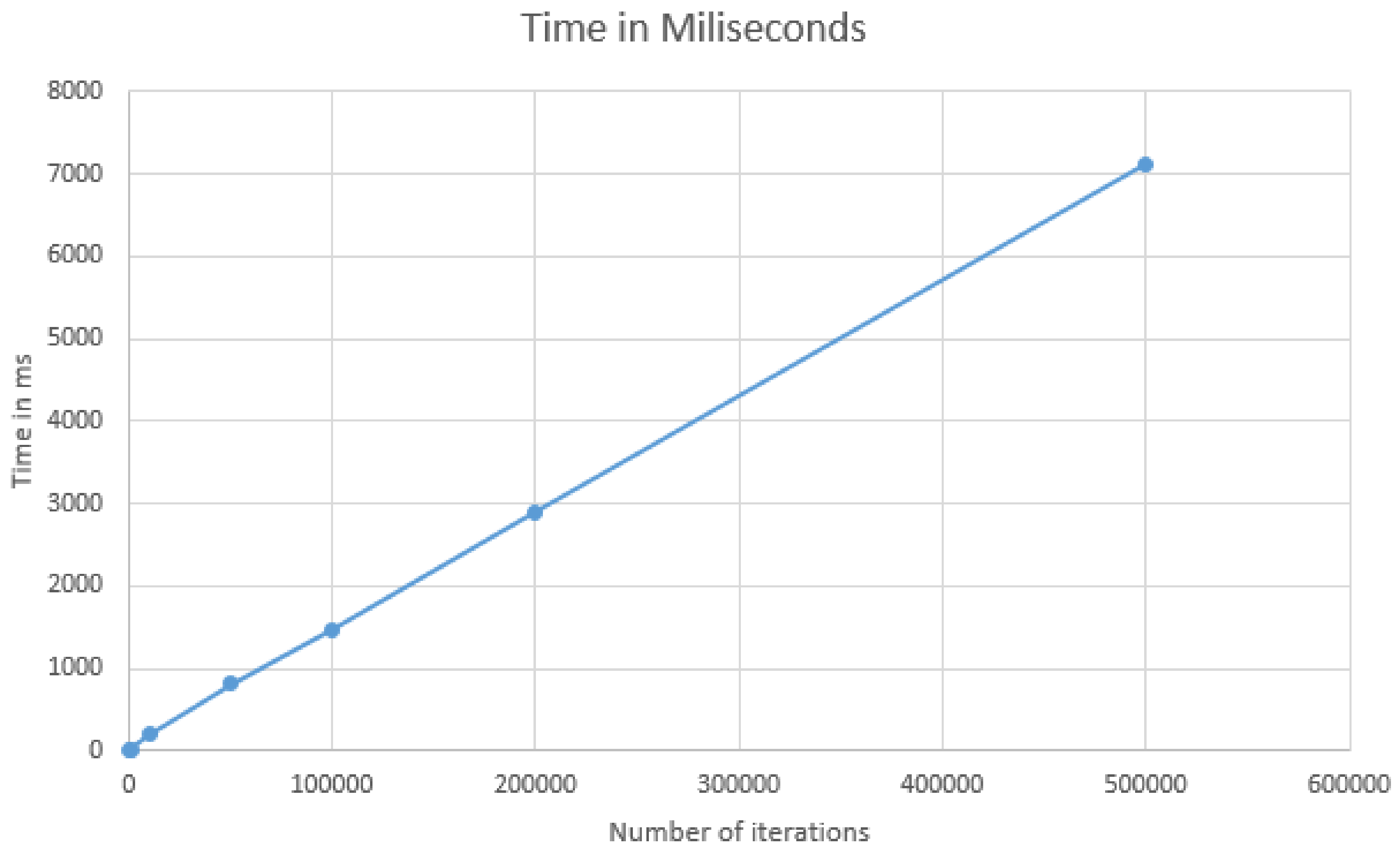
Password Based Key Derivation Functions

```
public static byte[] HashPassword(byte[] password, byte[] salt, int rounds)
{
    using (var rfc2898 = new Rfc2898DeriveBytes(password, salt, rounds))
    {
        return rfc2898.GetBytes(32);
    }
}
```

Code Demo

Password Based Key Derivation Functions

Password Based Key Derivation Functions



Module Summary

- Storing passwords in the clear
- Encrypting passwords
- Using hashes to store passwords
- Using salted hashes
- Using a password based key derivation function